COMAD 2014

Proceedings of the
20th International Conference on Management of Data

December 17-19, 2014
Hyderabad, India

Editors

Srikanta Bedathur
IBM Research, India

Divesh Srivastava
AT&T Labs-Research

Satyanarayana R Valluri
EPFL, Switzerland

Preface

For over two decades, the International Conference on Management of Data (COMAD), modeled along the lines of ACM SIGMOD, has been the premier international database conference hosted in India by Division II of Computer Society of India, CSI. The first COMAD was held in Hyderabad in 1989, and it is wonderful that in its 25th year it has returned to Hyderabad. The 20th edition in the COMAD series is held at the campus of International Institute of Information Technology (IIIT) Hyderabad, from December 17-19, 2014.

COMAD seeks to provide the community of researchers, practitioners, developers and users of data management technologies, a forum to present and discuss problems, solutions, innovations, experiences and emerging trends. Keeping with the fast changing landscape of data management and analytics, the scope of COMAD 2014 has evolved to include emerging topics in Big Data Analytics, Web, Information Retrieval, Data Mining and Machine Learning in addition to the traditional topics in data management.

This year's call for papers attracted 63 research submissions from across the world. Each research paper was rigorously reviewed by at least three members of the program committee, which featured 26 data management experts from academia and industry from 4 different continents. After in-depth discussions, we selected 6 high-quality research papers for presentation at the conference, 2 industry research papers, 6 poster presentations and 3 demonstrations.

COMAD 2014 features three keynote talks by Prof. S. Muthukrishnan (Rutgers University and Microsoft Research), Prof. Renée Miller (University of Toronto, Canada), and Srini V. Srinivasan (Founder and VP of Engg. and Operations, Aerospike Inc.). The keynotes focus on very different aspects of "Big-Data" challenge - algorithms, curation and engineering. The program also hosts 3 tutorials from leading experts covering entity extraction and disambiguation, data mining over large-scale software repositories and how it can help software engineering, and mining massive-scale web repositories. We also continued the tradition started by COMAD in 2010 to invite Indian authors of papers published in premier international conferences to present their work at COMAD. This year features 2 papers from SIGMOD, and one paper each from PVLDB, KDD and ICDE from this year.

This time, COMAD 2014, also has the opportunity to have a special invited session with Prof. Jayant Haritsa (IISc) who was awarded the prestigious Infosys Prize this year, adding to an already long list of his honors.

To ensure visibility of COMAD beyond this conference, these proceedings will also be available through ACM SIGMOD and DBLP.

We would like to thank all the members of the COMAD Organizing Committee and the COMAD Program Committee for their generous support, enabling us to put together such a high-quality program. We are also grateful for the support and generosity of our sponsors. Without our silver sponsors Microsoft, Google, Infosys and Honeywell as well as our bronze sponsor Progress, this conference would not be possible. We also thank IIIT-Hyderabad for providing a campus for the conference. Finally, we acknowledge the sustained cooperation and assistance extended by the Computer Society of India in organizing this event.

In closing, we welcome you to the COMAD 2014 conference in Hyderabad and hope you will have a fruitful and stimulating experience.

Kamal Karlapalem

IIIT-Hyderabad, Hyderabad, India

(General Chair)

Divesh Srivastava

AT&T Labs Research, USA

Srikanta Bedathur

IBM Research, India

(Program Co-Chairs)

Satyanarayana R Valluri

École Polytechnique Fédérale de Lausanne (EPFL), Switzerland

(Proceedings Chair)

Organizing Committee

GENERAL CHAIR	Kamal Karlapalem, IIIT Hyderabad
PROGRAM CHAIRS	Srikanta Bedathur, IBM Research, India Divesh Srivastava, AT&T Labs-Research
INDUSTRY CHAIR	Srinivasan Seshadri, Zettata
TUTORIALS & PANELS CHAIR	Sameep Mehta, IBM Research, India
POSTER & DEMO CHAIR	Manish Gupta, Microsoft India
PROGRAMMING CHALLENGE CHAIR	Arnab Bhattacharya, IIT Kanpur
WEB & PROCEEDINGS CHAIR	Satyanarayana R Valluri, EPFL, Switzerland
LOCAL ARRANGEMENTS CHAIR	P. Radhakrishna, Infosys, India

Program Committee

Srikanta Bedathur	IBM Research
Arnab Bhattacharya	Indian Institute of Technology, Kanpur
Indrajit Bhattacharya	IBM India Research Lab
Gautam Das	University of Texas at Arlington, USA
Mahashweta Das	HP Labs, Palo Alto
Prasad Deshpande	IBM Research - India
Lipika Dey	TCS Innovation Lab Delhi
Niloy Ganguly	Indian Institute of Technology Kharagpur
Vikram Goyal	IIIT-Delhi
Manish Gupta	Microsoft
Jayant Haritsa	Indian Institute of Science, Bangalore
Katja Hose	Aalborg University
Kalapriya Kannan	IBM
Gjergji Kasneci	Hasso-Plattner-Institute
Sameep Mehta	IBM Research
Karin Murthy	IBM Research
Aditya Parameshwaran	UIUC
Dhaval Patel	IIT Ropar
Krishna Reddy Polepalli	IIIT-H
Vikram Pudi	IIIT-H
Maya Ramanath	IIT Delhi
Sayan Ranu	IIT Madras
Ralf Schenkel	Universitaet Passau
Seshadri Srinivasan	Zettata
Divesh Srivastava	AT&T Labs-Research
S. Sudarshan	IIT Bombay

Table of Contents

Keynotes

Tutorials

Research Papers

Industry Papers

Poster Presentations

Demos

KEYNOTES

The Sublinear Approach to Big Data Problems

(Keynote)
Prof. S. Muthukrishnan
Department of Computer Science
Rutgers University
muthu@cs.rutgers.edu

ABSTRACT

We will discuss approaches to solving Big Data problems that use sublinear resources such as storage, communication, time, processors etc. We will also discuss potential models of computing that arise from this perspective. Finally, we will discuss new Big Data problems that arise from social network analysis, including ranking, scoring and others.

Biography

Muthu is a Professor in Rutgers Univ. and on leave. His research focus is on algorithms and databases. His recent research is on analyzing massive data streams and on Economics and optimization problems in online ad systems.

The 20th International Conference on Management of Data (COMAD),
17th-19th Dec 2014 at Hyderabad, India.

Big Data Curation

(Keynote)
Prof. Renée Miller
Department of Computer Science
University of Toronto
miller@cs.toronto.edu

ABSTRACT

A new mode of inquiry, problem solving, and decision making has become pervasive in our society, consisting of applying computational, mathematical, and statistical models to infer actionable information from large quantities of data. This paradigm, often called Big Data Analytics or simply Big Data, requires new forms of data management to deal with the volume, variety, and velocity of Big Data. Many of these data management problems can be described as data curation. Data curation includes all the processes needed for principled and controlled data creation, maintenance, and management, together with the capacity to add value to data. In this talk, I describe our experience in curating some open data sets. I overview how we have adapted some of the traditional solutions for aligning data and creating semantics to account for (and take advantage of) Big Data.

Biography

Prof. Rene Miller received BS degrees in Mathematics and in Cognitive Science from the Massachusetts Institute of Technology. She received her MS and PhD degrees in Computer Science from the University of Wisconsin in Madison, WI. She is a Fellow of the Royal Society of Canada (Canada's National Academy) and the Bell Canada Chair of Information Systems at the University of Toronto. She received the US Presidential Early Career Award for Scientists and Engineers (PECASE) , the highest honor bestowed by the United States government on outstanding scientists and engineers beginning their careers and the National Science Foundation Career Award. She is a Fellow of the ACM, a former President of the VLDB Endowment, and was the Program Chair for ACM SIGMOD 2011 in Athens, Greece. Her work has focused on the long-standing open problem of data integration and has achieved the goal of building practical data integration systems. She was a co-recipient of the ICDT Test-of-Time Award for an influential 2003 paper establishing the foundations of data exchange.

The 20th International Conference on Management of Data (COMAD),
17th-19th Dec 2014 at Hyderabad, India.

Lessons Learned in Building Real-time Big Data Systems

(Keynote)

Srini V. Srinivasan
Founder and VP Engineering & Operations
Aerospike
srini@aerospike.com

In the Age of the Customer, enterprises must modernize their application infrastructure to use real-time big data to attract, engage and retain consumers across devices, media and channels. Processing massive amounts of data in real-time creates a competitive advantage that has an enormously positive impact on business.

It has been clear now for a long time that lower latency means higher sales for Internet enterprises. In fact, Internet sites routinely lose users to other sites that support lower latency. E.g., Amazon found every 100ms of latency cost them 1% in sales. Google found an extra .5 seconds in search page generation time dropped traffic by 20%[1].

Therefore, predictable low latency is a sure fire way to win in the marketplace. Nowhere has this been more apparent than in the growth of Real-Time Bidding (RTB) systems for delivering digital advertising.

RTB has been effectively used to monetize "long tail" (remnant) inventory and target users across websites and mobile apps, anywhere they might be on the Internet. In fact, RTB has been the key factor driving the enormous growth in digital advertising worldwide. Low latency is a lynchpin of the RTB system, where the entire process from click to view must complete in under 150 milliseconds.

Platform companies realized the critical nature of keeping this contract[2]. At the center of such a business is fast access to data. Note that the user data in an RTB system is changing constantly since the choice of actions at every user visit needs to take into account past behavior of that user. So, such RTB applications need databases that provide predictable sub-millisecond latency for reads in the presence of heavy write load.

Clearly traditional systems are not sufficient for this. It has been known for a while that Database Systems need a complete rewrite[3]. Even most of the first generation NoSQL systems are inadequate. Some of the RTB majors have used custom systems they developed on their own on top of other inadequate systems. In fact, building a fast in-memory system on top of a slow Database could be a "fate worse than death"[4]. The most successful companies use ultra-fast clustered systems[5] or single node systems[6]. These systems work quite well on bare metal[7] or in the cloud[8].

System developers and operators face several issues while deciding to use such a new Database system for their applications:

- From the application point of view, the system needs to be able to deliver extremely low latency for reads in the presence of heavy write load. This is an especially hard problem to solve for traditional databases. In addition, the system must provide support for queries in addition to simple (and fast) key value access.
- It is important that applications work in both cloud based virtual deployments as well as on bare metal data center deployments. Specifically, it is critical that applications work on commodity hardware with no special purpose setup needed for launch.
- As more and more mainstream enterprises move to low latency applications, it is important to avoid sacrificing consistency at the altar of availability[9]. The best systems are those that make judicious choices and provide availability and consistency with high performance in a wide variety of useful scenarios[10]. For example, minimizing network partitions considerably reduces the negative effects of the CAP theorem and it is hard but not impossible to provide ACID support.
- Parallelism is quite powerful both within a node as well as across nodes. Harnessing the best performance and scaling up on one node and scaling out are both important. For example, using a hybrid in-memory system using both DRAM and SSD (Flash), one can run a 14-node cluster using a DRAM/SSD configuration instead of a 186-node cluster using a pure DRAM system. Such a cluster will still provide sub-millisecond latency, but do so at a ten times lower cost than pure DRAM systems.
- Operational excellence is necessary to ensure that a service runs 24X7. All code should be written so that it can run as a service. Extremely high performance (e.g., 1 million TPS per node) provides sufficient headroom for making sure that failures can be handled seamlessly. Additional capacity can also be used to provide better consistency in the presence of failures.
- High performance in a system can be achieved by ensuring that software takes maximum advantage of the performance of hardware. Techniques that are useful: using multiple threads, reference counts to avoid data copies, efficient memory usage (e.g., restricting the index entries to 64 bytes, the same as a cache line), real-time prioritization algorithms to keep the system running smoothly, etc.

The 20th International Conference on Management of Data (COMAD),
17th-19th Dec, 2014 at Hyderabad, India.

To conclude, by making appropriate choices, predictable low latency can co-exist with enough consistency in the vast majority of big data systems. This will enable enterprises to build real-time applications that add to the top line of every Internet enterprise.

Author:

Srini V. Srinivasan, founder and VP Engineering & Operations

Srini brings 20-plus years of experience in designing, developing and operating Web-scale infrastructures, and he holds over a dozen patents in database, Internet, mobile, and distributed system technologies. Srini co-founded Aerospike to solve the scaling problems he experienced with relational databases at Yahoo! where, as senior director of engineering, he had global responsibility for the development, deployment and 24×7 operations of Yahoo!'s mobile products, in use by tens of millions of users. Srini also was chief architect of IBM's DB2 Internet products, and he served as senior architect of digital TV products at Liberate Technologies. Srini has a B.Tech in Computer Science from IIT Madras and a M.S. and PhD in Databases from University of Wisconsin-Madison.

1. REFERENCES

[1] http://glinden.blogspot.com/2006/11/marissa-mayer-at-web-20.html

[2] http://www.adexchanger.com/online-advertising/equinix-seeks-to-speed-rtb-bidding/

[3] Michael Stonebraker, Samuel Madden, Daniel J. Abadi, Stavros Harizopoulos, Nabil Hachem, and Pat Helland. The end of an architectural era: (it's time for a complete rewrite). In *Proceedings of the 33rd International Conference on Very Large DataBases* (VLDB). 2007.

[4] https://gigaom.com/2011/07/07/facebook-trapped-in-mysql-fate-worse-than-death/

[5] http://highscalability.com/blog/2014/5/6/the-quest-for-database-scale-the-1-m-tps-challenge-three-des.html

[6] http://highscalability.com/blog/2014/8/27/the-12m-opssec-redis-cloud-cluster-single-server-unbenchmark.html

[7] http://www.aerospike.com/wp-content/uploads/2013/01/Ultra-High-Performance-NoSQL-Benchmarking.pdf

[8] http://highscalability.com/blog/2014/8/18/1-aerospike-server-x-1-amazon-ec2-instance-1-million-tps-for.html

[9] Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall, and Werner Vogels. Dynamo: amazon's highly available key-value store. In *Proceedings of 21st ACM SIGOPS Symposium on Operating Systems Principles* (SOSP). 2007.

[10] V. Srinivasan and Brian Bulkowski. Citrusleaf: A Real-Time NoSQL DB which Preserves ACID. *Proc. VLDB Endow. (PVLDB) Vol.4(12), 1340-1350.* 2011.

TUTORIALS

Entity Linking: Detecting Entities within Text

Deepak P[1] Sayan Ranu[2]
[1]*IBM Research – India, Bangalore, India*
[2]*Dept. of CS&E, IIT Madras, Chennai, India*
`deepak.s.p@in.ibm.com` `sayan@cse.iitm.ac.in`

1. MOTIVATION AND SUMMARY

With unstructured text on the web and social media increasing at a furious pace, it is all the more important to develop techniques that can ease semantic understanding of text data for humans. One of the key tasks in this process is that of entity linking; identifying *mentions* of entities in text. Consider the line that reads *"The Prime Minister came under harsh criticism over the Immigration Act 2014"* Without any additional context, it is not obvious to humans as to who is being talked about. An entity linking technique that has the entity database at its disposal, however, can easily figure out that the mention *Prime Minister* refers to the *Prime Minister of UK* since the mention of *Immigration Act 2014* in the same sentence narrows down the search space from the set of all countries that have Prime Ministers to just UK. Such linking of text documents to entities enables easier understanding for the reader, as well as improved accuracy in automated tasks such as text document clustering, classification and information retrieval.

With the advent of social media, the set of entities that have a presence on the web has increased from just famous places, objects and people, to everyone that has a social media presence, which is to say, virtually the vast majority of human beings. Availability of such a heterogeneous set of entities ranging from those in domain-specific ontologies to social media profiles provides fresh challenges and opportunities for entity linking. In this tutorial, we will cover the set of entity linking techniques that have been proposed in literature over the years, and provide a systematic survey of them with classifications along various dimensions. We will also explore the applicability of entity linking on noisy and short texts, such as those generated in microblogging platforms (ex. Twitter), and elaborate on the new challenges for entity linking that have not quite received enough attention from the scholarly community.

2. TUTORIAL ORGRANIZATION

We propose to organize this as a 1.5 hour tutorial. A brief outline of the tutorial content is as follows:

- **Introduction** (10 minutes)
  - In this segment, we will introduce the task of entity linking with examples as well as technical formalisms. We will motivate the problem and illustrate how entity linking can help in improving traditional learning tasks such as classification and clustering. We will also outline how entity linking differs from closely related tasks such as information extraction and named-entity detection.
- **Considerations in Entity Linking** (25 minutes)
  - We will next introduce the three phases of entity linking, viz., *mention detection*, *candidate discovery* and *entity assignment*. Of these, we will particularly focus on the three criteria that are used in the last phase of entity assignment, i.e., *entity popularity*, *entity-mention similarity* and *document-level coherence*. We will outline the measures that are often used in quantifying each of these notions; for example, entity popularity is often quantified using anchor texts [3], whereas entity-mention similarity is estimated using text similarity metrics [1]. Document-level coherence of entities, on the other hand, is a set-level property and is estimated using graph-mining techniques such as in AIDA [8].
- **Classification of Entity-Linking Techniques** (15 minutes)
  - Entity Linking methods may be classified based on various attributes; in this section, we will analyze entity linking techniques with respect to two major attributes, those pertaining to *usage of supervision* and *document length*. Along the first dimension, we will outline the usage of supervision in techniques such as those in [5] and [7] and the approaches followed by the more popular paradigm of unsupervised entity linking [4, 3]. Most entity linking techniques focus on document-type articles; in this context, we will also delve into techniques that deal with short texts [2] and tweets [6].
- **Evaluation of Entity Linking** (10 minutes)
  - Entity Linking techniques are evaluated using common IR-based metrics such as precision, MAP, MRR and NDCG when ranked lists are output by the techniques [1]. On the other hand, if the entities are returned as sets, set-based evaluation metrics such as recall and F-measure are used. We will introduce these metrics and provide intuitions on which metrics are suitable for various scenarios.

The 20th International Conference on Management of Data (COMAD), 17th-19th Dec 2014 at Hyderabad, India.

---

[1] http://nlp.stanford.edu/IR-book/html/htmledition/evaluation-of-ranked-retrieval-results-1.html

- **Resources for Entity Linking** (10 minutes)
  - Towards motivating the audience to consider entity linking as a field of study and/or exploration, we will outline the various resources that are readily available on the web. These include entity repositories such as Wikipedia[2], Yago[3] as well as numerous text collections. We will also include pointers to entity linking systems that can be accessed on the web.
- **Challenges in Entity Linking** (10 minutes)
  - In this segment, we will systematically explore challenges that have received limited attention from the scholarly community. These include tasks pertaining to entity linking on new entity datasets (e.g., social media profiles) as well as new kinds of document datasets (e.g., scholarly articles, web search queries etc.). Additionally, we will also spend some time discussing methods by which entity linking techniques can enhance general Information Retrieval.
- **Conclusions and Discussion** (10 minutes)

3. TARGETED AUDIENCE & EXPECTATIONS

This tutorial is targeted towards computer scientists interested in the field of data analytics, which includes graduate students and faculty members from academia as well as industry professionals. The tutorial is organized in a self-contained way and does not assume any particular expertise from the audience. By the end of the tutorial, the goal is to expose the audience to the diverse set of problems arising in entity linking, demonstrate how these problems translate to real life applications, and finally, equip attendees with technical insights on how these problems can be solved.

The tutorial is of interest to the COMAD audience since entity linking from text data is a vibrant and active research area due to the omniprescence of social networks in human lives. The tutorial will survey techniques from top publication venues while maintaining a striking balance between the theoretical concepts and their practical importance.

4. BRIEF BIOGRAPHY

Deepak P: Deepak is a researcher in the Information Management Group at IBM Research - India, Bangalore. He obtained his B.Tech degree from Cochin University, India followed by M.Tech and PhD degrees from IIT Madras, India, all in Computer Science. His current research interests include Similarity Search, Spatio-temporal Data Analytics, Graph Mining, Information Retrieval and Machine Learning. He is a senior member of the ACM and IEEE.

Sayan Ranu: Sayan is an Assistant Professor at IIT Madras. Prior to joining IIT Madras, he was a researcher in the Information Management group at IBM Research - India, Bangalore. He obtained his PhD from University of California, Santa Barbara. His current research interests include spatio-temporal data analytics, graph indexing and mining, and bioinformatics.

5. REFERENCES

[1] J. Dalton and L. Dietz. A neighborhood relevance model for entity linking. In *Proceedings of the 10th Conference on Open Research Areas in Information Retrieval*, pages 149–156. LE CENTRE DE HAUTES ETUDES INTERNATIONALES D'INFORMATIQUE DOCUMENTAIRE, 2013.

[2] P. Ferragina and U. Scaiella. Tagme: on-the-fly annotation of short text fragments (by wikipedia entities). In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 1625–1628. ACM, 2010.

[3] J. Hoffart, M. A. Yosef, I. Bordino, H. Fürstenau, M. Pinkal, M. Spaniol, B. Taneva, S. Thater, and G. Weikum. Robust disambiguation of named entities in text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 782–792. Association for Computational Linguistics, 2011.

[4] S. Kulkarni, A. Singh, G. Ramakrishnan, and S. Chakrabarti. Collective annotation of wikipedia entities in web text. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 457–466. ACM, 2009.

[5] Y. Li, C. Wang, F. Han, J. Han, D. Roth, and X. Yan. Mining evidences for named entity disambiguation. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1070–1078. ACM, 2013.

[6] E. Meij, W. Weerkamp, and M. de Rijke. Adding semantics to microblog posts. In *Proceedings of the fifth ACM international conference on Web search and data mining*, pages 563–572. ACM, 2012.

[7] A. Pilz and G. Paaß. Collective search for concept disambiguation. 2012.

[8] M. A. Yosef, J. Hoffart, I. Bordino, M. Spaniol, and G. Weikum. Aida: An online tool for accurate disambiguation of named entities in text and tables. *Proceedings of the VLDB Endowment*, 4(12):1450–1453, 2011.

---

[2] http://en.wikipedia.org

[3] http://www.mpi-inf.mpg.de/yago-naga/yago

Kashvi: A Framework for Software Process Intelligence

Ashish Sureka
IIIT-Delhi, India
ashish@iiitd.ac.in

Atul Kumar
Siemens Research, India
kumar.atul@siemens.com

Girish Maskeri Rama
Infosys Labs, India
Girish_Rama@infosys.com

ABSTRACT

Software Process Intelligence (SPI) is an emerging and evolving discipline involving mining and analysis of software processes. This is modeled on the lines of Business Process Intelligence (BPI), but with the focus on software processes and its applicability in software systems. Process mining consists of mining event log and process trace data for the purpose of process discovery (run-time process model), process verification or compliance checking (comparison between design-time and run-time process model), process enhancement and recommendation. Software Process Mining or Intelligence is a new and emerging discipline which falls at the intersection of Software Process & Mining, and Software & Process Mining. Software Process Mining is integral to discovering and verifying the processes in a software system.

Software Process Mining is a three word phrase which can be viewed from two perspectives: Software + Process Mining and Software Process + Mining. Software development and evolution involves usage of several workflow management and information systems and tools such as Issue Tracking Systems (ITS), Version Control Systems (VCS), Peer Code Review Systems (PCR) and Continuous Integration Tools (CIT). Such information systems log data consisting of events, activities, time-stamp, user or actor and context specific data. Such events or trace data generated by information systems used during software construction (as part of the software development process) contains valuable information which can be mined for gaining useful insights and actionable information. In this paper, we present *Kashvi*: A Framework for Software Process Intelligence

Categories and Subject Descriptors

H.2.8 [**Database Applications**]: Data Mining

Keywords

Automated Software Engineering, Business Process Intelligence (BPI), Mining Software Repositories, Process Mining, Software Process Intelligence

The 20th International Conference on Management of Data (COMAD), 17th-19th Dec 2014 at Hyderabad, India.

1. PROCESS MINING

Process mining is an area at the intersection of business process intelligence and data mining consisting of mining event logs from process aware information systems for the purpose of process discovery, process performance analysis, conformance verification, process improvement and organizational analysis. The approaches and algorithms within process mining enables information extraction from event logs or traces generated as a result of execution of a business process [7][8]. An audit trails of a workflow management system within a health-care organization (Hospital Information Management System) can be used to discover models describing processes and organizations. Similarly, the transaction logs of an enterprise resource planning system within a manufacturing unit can be used to discover models describing processes which can be used for process conformance and verification [7][8]. The event logs consists of several events. Each event in the event-log refers to an activity which is a well-defined step within the business process. Each event also refers to a case or trace (i.e., a process instance). Each event can have a performer also referred to as originator (the actor executing or initiating the activity) and events have a timestamp. The events in the event-logs are totally ordered [7][8].

ProM[1] (an abbreviation for Process Mining framework) is a Free and Open Source tool as well as framework for process mining algorithms. ProM provides a usable and scalable platform to process analysts and developers of the process mining algorithms. The architecture of ProM is such that it is easy to extend using plug-ins. ProM consists of several types of plug-ins. Mining plug-ins which implement mining algorithm to construct a Petri-Net based on an event log. Import and Export plug-ins, Analysis plug-ins and conversion plug-ins (which implement conversions between different data formats, e.g., from EPCs to Petri-Nets)

2. MINING SOFTWARE REPOSITORIES

Large and complex software projects use defect tracking systems for managing the workflow of bug reporting, archiving, triaging and tracking. Version control or source code control systems are used to manage changes to project files and documents. Peer code review systems are used to manage peer review of source code before committing the

---

[1] http://www.processmining.org/prom/start

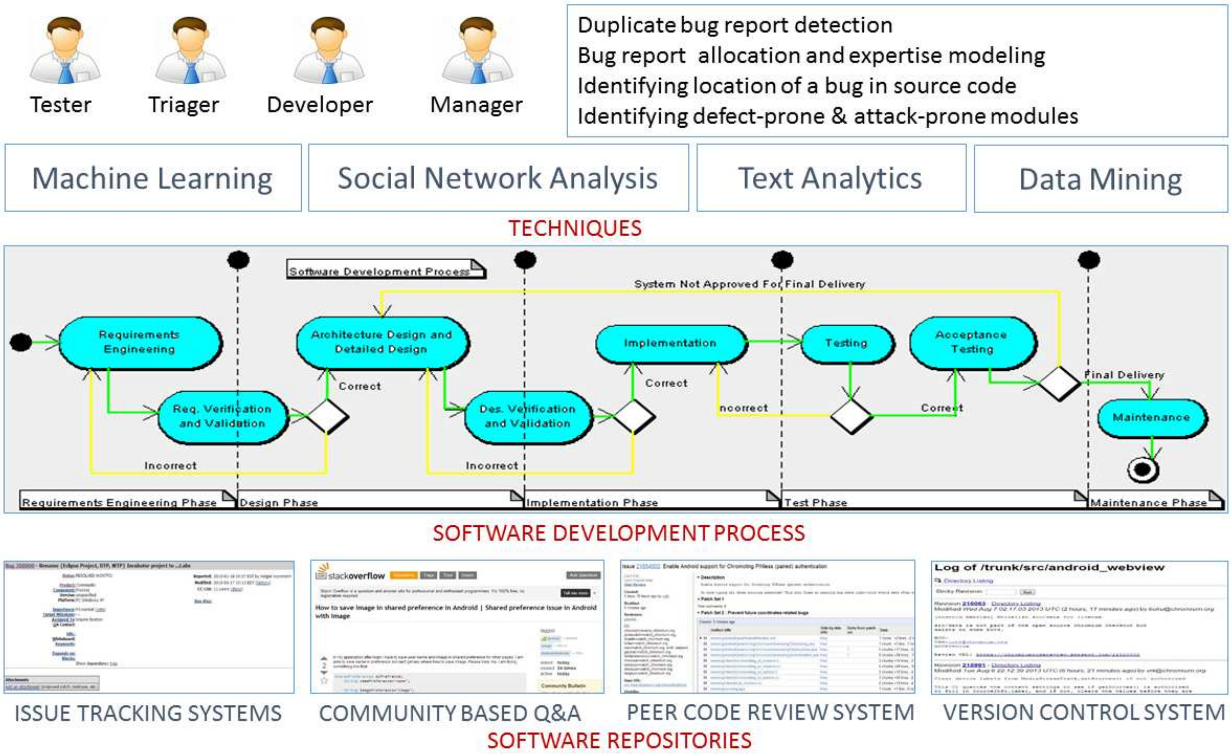

Tester
Triager
Developer
Manager
Duplicate bug report detection
Bug report allocation and expertise modeling
Identifying location of a bug in source code
Identifying defect-prone & attack-prone modules
Machine Learning
Social Network Analysis
Text Analytics
Data Mining
TECHNIQUES
Software Development Process
System Not Approved For Final Delivery
Requirements Engineering
Architecture Design and Detailed Design
Implementation
Testing
Acceptance Testing
Req. Verification and Validation
Correct
Incorrect
Des. Verification and Validation
Correct
Incorrect
Incorrect
Correct
Final Delivery
Maintenance
Requirements Engineering Phase
Design Phase
Implementation Phase
Test Phase
Maintenance Phase
SOFTWARE DEVELOPMENT PROCESS
stackoverflow
How to save image in shared preference in Android | Shared preference Issue in Android with Image
Log of /trunk/src/android_webview
ISSUE TRACKING SYSTEMS
COMMUNITY BASED Q&A
PEER CODE REVIEW SYSTEM
VERSION CONTROL SYSTEM
SOFTWARE REPOSITORIES

Figure 1: Kashvi: A Framework for Software Process Intelligence. Figure showing the Software Repositories, Data getting generated during Construction of Software, Mining Techniques, Practitioners and Problems Encountered by Practitioners

source code to identify defects though inspection. Community based Q&A websites for programmers and online forums are widely used by developers for asking questions and sharing knowledge. Bug databases, version archives, source code repository, peer code review system, community based Q&A websites, mailing lists and online forums for programmers are software repositories containing large volumes of valuable structured data and unstructured data (free-form text) entered by developers during the software development process. For example, a bug report typically contains information describing the problem, application environment, steps to reproduce and stack trace. A source control system contains information regarding the files that were revised, the changes that were made, developer who made the change, developer comments and time-stamp.

These repositories have been primarily serving the purpose of archiving information or recording keeping. Mining Software Repositories (MSR) researchers have investigated social network analysis, data mining, machine learning and information retrieval based approaches to analyze software repositories to uncover interesting patterns and knowledge which can be used to support developers in the process of software maintenance. The work on Mining Software Repositories is based on the premise that historical data present in software repositories can be mined to derive actionable information resulting in increased productivity and effectiveness of developers [1][9]. Researchers have also conducted field studies and survey of practitioners to understand problems encountered by them and developed mining software repositories based solutions to address the problems encountered by developers and project teams [1][9]. Some of the general themes[2] within MSR are: analysis of software ecosystems and mining of repositories across multiple projects, models for social and development processes that occur in large software projects, prediction of future software qualities via analysis of software repositories, models of software project evolution based on historical repository data, characterization, classification, and prediction of software defects based on analysis of software repositories, techniques to model reliability and defect occurrences, search-driven software development, including search techniques to assist developers in finding suitable components and code fragments for reuse, and software search engines, analysis of change patterns and trends to assist in future development and Visualization techniques and models of mined data [1][9].

3. SOFTWARE PROCESS INTELLIGENCE

Software Process Intelligence (SPI) is an emerging and evolving discipline involving mining and analysis of software processes. This is modeled on the lines of application of Business Intelligence techniques to business processes (Business Process Intelligence (BPI)), but with the focus on software processes and its applicability in software engineering and information technology systems. Software Process Mining or falls at the intersection of Software Process & Mining, and Software & Process Mining. It is a three word phrase which can be viewed from two perspectives: Software + Process Mining and Software Process + Mining. Software

---

[2] http://2015.msrconf.org/

development and evolution involves usage of several workflow management and information systems and tools such as Issue Tracking Systems (ITS), Version Control Systems (VCS), Peer Code Review Systems (PCR) and Continuous Integration Tools (CIT). Such information systems log data consisting of events, activities, time-stamp, user or actor and context specific data. Such events or trace data generated by information systems used during software construction (as part of the software development process) contains valuable information which can be mined for gaining useful insights and actionable information [5][6].

Figure 1 illustrates the broad framework for Software Process Intelligence. As shown in Figure, the framework consists of software repositories (version control system, issue tracking system, peer code review system, community based Q&A websites, source code repositories and developer mailing lists) containing data generated as part of constructing a software. Figure 1 shows the complete software development process: requirements engineering, design, implementation, test and maintenance. Software Process Intelligence consists of applying machine learning, information retrieval, social network analysis, text analytics and data mining based techniques on the software engineering data to extract actionable information aimed at solving problems encountered by practitioners. Figure 1 shows the practitioners (tester, triager, developer, project manager, quality assurance manager, requirements engineer) and some of the technical problems (defect prediction, identifying fault-prone entities, bug localization, automatic bug triaging, bug report allocation and expertise modeling)

Software Process Intelligence has diverse applications and is an area that has recently attracted several researcher's attention due to availability of vast data generated during software development. Some of the business applications of process mining software repositories are: uncovering runtime process model, discovering process inefficiencies and inconsistencies, observing project key indicators and computing correlation between product and process metrics, extracting general visual process patterns for effort estimation and analyzing problem resolution activities [2][3][5][6]. Some of the themes within Software Process Intelligence are: Big-Data and scalability issues in software process intelligence, Integration of agile development methods and process mining, Metrics for software process intelligence, Predictive analysis using process mining results, Privacy and confidentiality aspects in software process intelligence, Process mining for software process assessment and improvement, Program workflow mining, Relationship between effect of software process intelligence and organizational performance, Software process intelligence tool support, Software process intelligence in small and medium scale enterprises, Software quality and use of software process intelligence, Techniques to monitor software processes, Visualization in software processes, Visualization of software process mining and/or conformance results.

Mittal et al. present an approach for mining the process data (process mining) from software repositories archiving data generated as a result of constructing software by student teams in an educational setting [4]. They present an application of mining three software repositories: team wiki (used during requirement engineering), version control system (development and maintenance) and issue tracking system (corrective and adaptive maintenance) in the context of an undergraduate Software Engineering course [4]. Gupta et al. present an application of process mining three software repositories (ITS, PCR and VCS) from control flow and organizational perspective for effective process management [3]. They discover runtime process model for bug resolution process spanning three repositories using process mining tool, Disco, and conduct process performance and efficiency analysis. They identify bottlenecks, define and detect basic and composite anti-patterns. In addition to control flow analysis, they mine event log to perform organizational analysis and discover metrics such as handover of work, subcontracting, joint cases and joint activities [3]. Gupta et al. apply business process mining tools and techniques to analyze the event log data (bug report history) generated by an issue tracking system with the objective of discovering runtime process maps, inefficiencies and inconsistencies. They conduct a case-study on data extracted from Bugzilla issue tracking system of the popular open-source Firefox browser project [2].

4. REFERENCES

[1] Msr 2014: Proceedings of the 11th working conference on mining software repositories. 2014.

[2] M. Gupta and A. Sureka. Nirikshan: Mining bug report history for discovering process maps, inefficiencies and inconsistencies. In *Proceedings of the 7th India Software Engineering Conference*, ISEC '14, pages 1:1–1:10, 2014.

[3] M. Gupta, A. Sureka, and S. Padmanabhuni. Process mining multiple repositories for software defect resolution from control and organizational perspective. In *Proceedings of the 11th Working Conference on Mining Software Repositories*, MSR 2014, pages 122–131, 2014.

[4] M. Mittal and A. Sureka. Process mining software repositories from student projects in an undergraduate software engineering course. In *Companion Proceedings of the 36th International Conference on Software Engineering*, ICSE Companion 2014, pages 344–353, 2014.

[5] W. Poncin, A. Serebrenik, and M. van den Brand. Process mining software repositories. In *Software Maintenance and Reengineering (CSMR), 2011 15th European Conference on*, pages 5–14, March 2011.

[6] V. Rubin, C. W. Günther, W. M. P. Van Der Aalst, E. Kindler, B. F. Van Dongen, and W. Schäfer. Process mining framework for software processes. In *Proceedings of the 2007 International Conference on Software Process*, ICSP'07, pages 169–181, 2007.

[7] W. van der Aalst, T. Weijters, and L. Maruster. Workflow mining: Discovering process models from event logs. *IEEE Trans. on Knowl. and Data Eng.*, 16(9):1128–1142, Sept. 2004.

[8] W. M. P. van der Aalst. *Process Mining: Discovery, Conformance and Enhancement of Business Processes.* Springer Publishing Company, Incorporated, 1st edition, 2011.

[9] T. Zimmermann, M. D. Penta, and S. Kim. Proceedings of the 10th working conference on mining software repositories, msr '13, san francisco, ca, usa, may 18-19, 2013. 2013.

Exploration and Mining of Web Repositories

Gautam Das
Computer Science and Engineering
University of Texas at Arlington
gdas@uta.edu

ABSTRACT

With the proliferation of very large data repositories hidden behind web interfaces, e.g., keyword search, form-like search and hierarchical/graph-based browsing interfaces for Amazon.com, eBay.com, etc., efficient ways of searching, exploring and/or mining such web data are of increasing importance. There are two key challenges facing these tasks: how to properly understand web interfaces, and how to bypass the interface restrictions. In this tutorial, we start with a general overview of web search and data mining, including various exciting applications enabled by the effective search, exploration, and mining of web repositories. Then, we focus on the fundamental developments in the field, including web interface understanding, sampling, and data analytics over web repositories with various types of interfaces. We also discuss the potential changes required for query processing, data mining and machine learning algorithms to be applied to web data. Our goal is two-fold: one is to promote the awareness of existing web data search/exploration/mining techniques among all web researchers who are interested in leveraging web data, and the other is to encourage researchers, especially those who have not previously worked in web search and mining before, to initiate their own research in these exciting areas.

Biography

Gautam Das is a Full Professor in the Computer Science and Engineering Department of the University of Texas at Arlington. Prior to UTA, Dr. Das has held positions at Microsoft Research, Compaq Corporation and the University of Memphis, as well as visiting positions at IBM Research. He graduated with a BTech in computer science from IIT Kanpur, India, and with a PhD in computer science from the University of Wisconsin- Madison. Dr. Das's research interests span social computing, data mining, information retrieval, databases, graph and network algorithms, and computational geometry. His research has resulted in over 150 papers, many of which have appeared in premier conferences and journals. He is the recipient of the IEEE ICDE 2012 Influential Paper Award. His research has been supported by grants from federal and state agencies such as US National Science Foundation, US Office of Naval Research, US Department of Education, Texas Higher Education Coordinating Board, Qatar National Research Fund, as well as industry such as Cadence, Nokia, Apollo, and Microsoft.

The 20th International Conference on Management of Data (COMAD), 17th-19th Dec 2014 at Hyderabad, India.

RESEARCH PAPERS

A Model Independent and User-Friendly Querying System for Indoor Spaces

Amrutha H.
Dept. of Computer Science and Engineering,
Amrita School of Engineering, Coimbatore
Amrita Vishwa Vidyapeetham (University)
amrutha.hari12@gmail.com

Vidhya Balasubramanian
Dept. of Computer Science and Engineering,
Amrita School of Engineering, Coimbatore
Amrita Vishwa Vidyapeetham (University)
b_vidhya@cb.amrita.edu

ABSTRACT

Querying indoor information has become important with increasing demand for indoor pervasive applications in vogue. A number of applications have been developed like indoor navigation, localization etc., which work on the modeled indoor data. Different models like geometric, spatial and topological models exist for the indoor space. Existing query languages are model specific, and not user friendly. We propose a querying system which will work irrespective of the underlying model by hiding the complex details of the indoor model from the user. A querying framework is developed which abstracts out basic entities and primitive operators from multiple models. A text-based query language for the indoor space is built on this framework. A visual querying interface is developed which further simplifies the task of querying.

Index Terms- indoor information modeling, querying framework, visual querying

1. INTRODUCTION

Indoor information modeling and management has gained significance, with a large number of applications like indoor navigation, localization, asset management etc operating on the indoor space. To support these applications an effective querying framework over indoor space is necessary. Existing querying systems over indoor space have been developed based on the underlying indoor models like geometric, spatial and topology based models [7]. Models constructed for the indoor space, represent its entities like rooms, doors etc, relations between the entities and a set of constraints. Each model deals with different aspects of an indoor space. Spatial models represent the spatial attributes of entities and relations, topology based models represent the space as a set of entities connected by a set of relations[11] etc. Each model is stored in suitable databases and querying is done using the general purpose query languages supported by them.

The current query languages which support querying over indoor models (e.g. SQL which supports spatial models [16], Cypher Query Language that supports topology based model [1] and BIMQL for Building Information Models that supports a semantic model [12]) have a syntax that is difficult for use by non-professional users. These languages use complex terminologies, and are tightly coupled with the underlying modeling framework. The user needs to be familiar with the specific terminologies associated with a framework, and the way in which the space is modeled, each time he queries the data model stored. In current systems, to query an indoor space, a naive user has to either directly query the underlying database using the associated general purpose query language or use an existing model specific language making the querying complicated. This necessitates development of a generalized query language which can work above multiple indoor data models.

The next challenge is that existing query languages over indoor space are complex i.e, though they use SQL like syntax, the queries are long and complicated. For instance for finding a path between two points in the indoor space, a function has to be written in the underlying language. There are no simple and direct constructs that can help users specify such queries easily. While such constructs have been developed for outdoor spatial applications, it has not been developed in the indoor domain to the best of our knowledge. Also, to ease the querying process further, effective visual querying systems are needed, as there are no known visual query interfaces for indoor spaces. Compared to text-based querying, visual querying mechanisms simplify the task of querying and provide an increased level of comprehension [13]. The user friendliness of querying can hence be improved by adopting a visual querying interface.

In this paper, we address the above issues by developing a model independent querying framework for the indoor space. This querying system can be used in different application scenarios irrespective of the underlying data models. Along with providing a model independent querying system, the work aims to enhance the user's querying experience, by defining an indoor query language that can help construct indoor queries easily both using SQL like syntax and a visual query interface.

To achieve these goals, we develop a querying framework which abstracts out the basic entities and operators which are common to multiple models. Based on this querying framework, SQL type text-based query operators are developed. An SQL type query language is developed as SQL

---

[0] This work has been funded in part by DST(India) grant DyNo. 100/IFD/2764/2012-2013

The 20th International Conference on Management of Data (COMAD),
17th-19st Dec, 2014 at Hyderabad, India.

syntax shares similarities with most of the existing query languages. A visual querying component is added above this language to help the user construct queries with much ease and improved comprehension. For using the querying system above multiple data models, translation modules are designed to translate the input queries to the general purpose languages supported by the models.

The rest of the paper is organized as follows : Section II and III present the related work and illustrate the architecture of the proposed indoor querying system. Section IV deals with design of the model independent querying framework along with its evaluation. The query translation and its evaluation are discussed in Section V. Conclusion and future directions are given in Section VI.

2. RELATED WORK

Our goal in this paper is to design a querying framework that is model independent and is user friendly. In this section, we detail the existing spatial querying approaches for both indoor and outdoor space, and motivate the need for our work.

One of the primary problems in querying spatial data is the complex syntax of the spatial functions. To address this, one of the earlier approaches to make querying over spatial data more easier is to use Structured Query Language (SQL) extensions. Works based on this approach, add functionalities to SQL for supporting spatial queries like shortest path and nearest neighbor queries. One such query language developed for spatial databases is Spatial SQL [8]. This provides support for spatial data types like lines and polygon, operators like intersects, disjoint etc., and predicates over SQL. Some systems, additionally use an interface that allows for spatial objects used in the queries to be picked from the screen. Another work with a similar approach is GEOQL (GEOgraphic Query Language) [14] that defines a similar set of spatial predicates for geographical data. In [3], a spatial query language for building information models is designed by adding extensions to SQL. It defines a set of geometric operators between objects in a 3D space by designing a 9IM (9 Intersection model). The operators defined are '*contain*', '*disjoint*', '*equal*', '*overlap*', '*touches*' and '*within*' between the geometries. In this language however, the specific terminologies in terms of IFC (Industry Foundation Classes) standard like IfcSpace, IfcDoors, etc. are used in the queries, making it difficult to use.

Another approach is to define a new language for a particular domain. A domain specific query language captures the semantics of the domain better than a general purpose query language. BIMQL (Building Information Model Query Language) [12] is an open source spatial query language developed for the spatial analysis of building information models. This is an improvement to the previously mentioned work that extended SQL for building information models. Building Information Modelling(BIM) is the standardization of IFC(Industry Foundation Classes) based models of buildings. The IFC specific terminologies like IfcDoor, IfcStandardWallCase etc., are replaced by natural language terms like 'doors', 'walls' etc. The language hides the complex terminologies involved in the IFC based modeling but does not reduce the complexities of query syntax. In addition the language is still tightly bound to the underlying model.

An indexing for the trajectories and a query language for finding the indoor objects is proposed in [10]. It uses two R-Tree based structures to represent the user trajectories. The queries defined are of the format, $Q(E_s, E_t, P)$ where E_s is an indoor space partition, E_t is the temporal extent and P is the topological predicate. This primarily is designed to support trajectory based querying and is not extensible to general indoor querying.

In order to support the heterogeneity in GIS data, a VirGIS mediation system is proposed in [4] for the outdoor space. There exist different data sources for GIS data (e.g. topographic maps, satellite images etc.). The system proposed in this work provides a unified model for supporting data from different data sources. A global schema is developed which represents a set of abstract features like roads, bridges etc. in the outdoor space. Mappings from the global schema to the underlying local data sources are done using one to one mappings. The queries issued to the global schema are converted using the corresponding local schema.

To improve user friendliness of queries several approaches have been proposed, one of which is to use natural language. One such system [17] adopts a controlled natural language interface for GIS(Geographic Information Systems). Since the introduction of natural language interfaces can lead to vague inputs from the user, the work proposes a controlled language interface. A semantic representation of the GIS queries called Lambda SQL is defined which serves as an intermediate representation to the interface. The natural language query is converted to the intermediate format which is then converted to the SQL query with spatial support. This language works only for outdoor queries and high level queries describing a building and is not generalizable to any model.

Another approach to increase ease of querying, is to use a menu based natural language interface as is proposed in (MBNLI) [18]. It uses a completion based menu interface where each word selected by the user is parsed and another set of words are suggested to construct the query. This helps overcome issues in natural language queries and prevents the user from writing vague queries. An extension to MBNLI is introduced in [5] to support geospatial queries. Here support for spatial operators such as intersects, contains, touches, covers, disjoint etc. are added as defined in Oracle. The MBNLI query, termed as LingoLogic query is converted to the equivalent spatial query. The output is converted to KML(Keyhole Markup Language) and displayed in Google Earth. However such approaches are yet to be tested in a 3d space.

Visual querying is another suitable approach, which helps the user construct queries through visual interactions. Users need not learn the query syntax as in the text-based query languages. Visual querying on spatial databases is presented in [13], where a diagrammatic technique is used based on a data flow metaphor. The flow of data between the input and output elements through one or more filters visually represents a query. Spatial entities and spatial relations (e.g. disjoint, touches, crosses, in etc.) are defined, which interact in constructing spatial queries. Another work [2] presents a prototype implementation of Spatial-Query-By-Sketch which is a sketch based user interface to query GIS data. While the previous work involves using a set of icons for querying, this approach processes the sketch drawn by the user to convert it into a canonical form called digital sketch. This format identifies the entities, their topological and directional relations.

While several querying approaches are available as mentioned above, they are developed to suit a particular modelling framework. Also, visual querying which makes the task of querying the most simpler is not implemented for indoor information. We develop a generic query language to support the various(spatial, geometric and topology based) models of the indoor space. Additionally a visual querying interface that helps enhance the user's experience of building the queries is introduced in the system.

3. ARCHITECTURE OF THE INDOOR QUERYING SYSTEM

We now explain the architecture of the proposed querying system which will work on multiple models of indoor space. To achieve this, the system works on a framework that abstracts out the details that are common to most used indoor models inorder to construct a generic representation. Text-based and visual querying languages are designed based on this framework. The working of the system starts with the user constructing a visual query, which is converted to the text-based query defined specifically for indoor spaces. This query is then converted to the corresponding query languages like SQL or cypher query language associated with the underlying database. Figure 1 presents the architecture diagram of the indoor querying system proposed in this work. The main modules of this system are explained as follows.

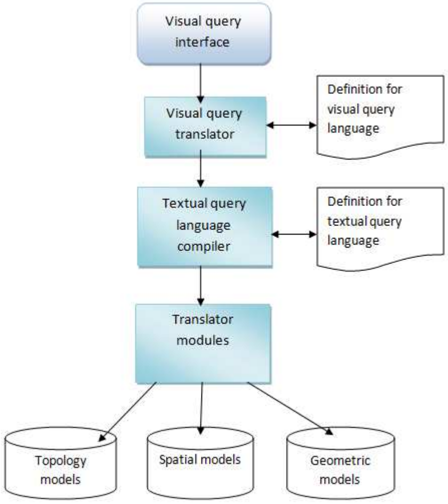


Figure 1: Architecture of the indoor querying system

- Visual query interface
  This provides a 3D visualization of the indoor space through which the user interacts to construct the visual queries. For each query, a set of visual interactions are defined like selecting the query type and giving the query parameters visually.

- Query compilers and translators
  These enable the translation of the input visual queries to a format which can be issued to the stored indoor data models. There are two query compiler modules defined in the system.

  - Visual query compiler
    The compiler processes the visual query input by the user to generate the query in corresponding text-based query language defined in the system. The relevant details like the query type and query parameters are extracted and substituted in the text query syntax.

  - Text-based query compiler
    This component parses the text-based query to validate the query syntax and aid its translation. The compiler on parsing each textual query generates an abstract syntax tree.

- Translator modules
  The parsed text query from the compiler module is fed to the translator to generate the queries in languages supported by each databases. Separate translation modules exist to generate queries in these general purpose query languages.(e.g. To SQL for PostGIS[16], to cypher queries for Neo4j [1] etc.)

- Databases
  Indoor information models are of different types like geometric, spatial and topological. Based on the data models, different databases are adopted (e.g. Topology based model best represented in a graph database like Neo4j, spatial models represented in PostGIS etc.).

The proposed querying framework works irrespective of the underlying models. The framework is formulated using the abstractions from different models. Based on this framework, a text-based and visual query languages are defined. The translation to the existing general purpose languages are done by the translator modules defined in the system. The next section will delve into the conceptual modeling of our querying framework.

4. MODEL INDEPENDENT QUERY FRAMEWORK

The primary purpose of this work as mentioned in previous sections is to generate an indoor querying system that is generic enough to support any indoor modeling framework. To achieve this, we propose the underlying framework that defines the basic indoor entities and primitive operators that operate in the indoor space. We identify the basic entities and operators in the main models of indoor space namely spatial, topological and geometric, and define a minimum common set that can map to entities and operators of these models in constant time. The identified entities and operators in the indoor space are as given below. While these entities are similar to the definitions of IndoorGML, they have been defined keeping in mind specifications in the most common indoor models.

- Space : This represents all the entities which semantically represent a space in a building's interior. These include rooms, corridors, sub-spaces of corridors/rooms etc. Space is created by a set of boundaries that determine its dimensions.

- Boundary: This corresponds to all the boundary structures which enclose the space entities in the indoor space. Boundaries can be classified as navigable boundaries and non-navigable boundaries. The non-navigable boundaries are the boundary components which block navigation, like the walls. The navigable boundaries are those which form the boundaries of the space entities and allow navigation, like the doors.
- Transition: Transitions are the entities in the indoor space which enable the movement from one space to another. They are the navigable connections which exist within any indoor environment. Examples of entities which belong to this class are exits, stairs and elevators.

To demonstrate the completeness of the chosen entities and operators, we show the correspondence between them and those in the existing modelling frameworks. The equivalence of entities in different models are shown in Table 1.

Basic Entities	PostGIS (spatial)	BIM (geometric)	IndoorGML (topology based)
Space	Polygons	IfcSpace	Abstract Space
Boundary	Polylines	IfcWall, IfcDoor, IfcWindow	Abstract Space Boundary
Transition	MultilineString	IfcStair	Transfer Space

Table 1: Entity equivalence

In the spatial model, space is characterised by its geometry which defines its extent and position in the space under consideration[11]. This model defines a set of geometry types like point, polygon, linestring, multilinestring etc, which is common to spatial database extensions like PostGIS. The indoor entities are represented as polygons, polylines or multilinestrings, providing a direct mapping to the spatial model. In the IndoorGML standard specification, which deals with the topology based modeling of indoor space, a set of classes are defined for entities and a set of relations between these entities to form the topologies. The classes defined for the entities are AbstractSpace to represent an indoor space(e.g. rooms), AbstractSpaceBoundary to represent boundaries of an indoor space(e.g. walls), TransferSpace to represent passages from one space to another(e.g.stairs) etc. [11]. Such mappings can also be provided to other specifications of the topological model. Semantic representation is used in the Building Information Modeling(BIM), which represents a building's design as a collection of objects. The underlying modeling of space is based on geometric modeling. The objects carry their geometry, attributes and relations [6]. Industry Foundation Classes(IFC) is the standard which allows representation and exchange of BIM data. Different classes exist which defines the indoor entity types like IfcSpace, IfcDoor, IfcWalls, IfcWIndows, IfcStairs etc. The space entity defined in our framework is an abstraction of IfcSpace, boundary is an abstraction of IfcWalls, IfcDoors etc., and transition is an abstraction of IfcStairs in BIM.

Next we define the set of primitive operators through which the basic entities interact with each other. These operators form a maximum subset such that they can be transformed to the operators in any model in constant time. Table 2 shows the primitive operators which are specified between each of the above defined entities.

	SPACE	BOUNDARY	TRANSITION
SPACE	Adjacent, Connected, Overlaps, Within, Intersects.	Boundedby, Linked	Linked, Connected
BOUNDARY	Bounds, Linked	Intersects, Touches	Intersects, Touches
TRANSITION	Linked, Connected	Intersects, Touches	Connected

Table 2: Primitive operators in proposed Indoor Query Framework

We define these operators based on relationships between entities in the indoor space in various contexts. A space is *'adjacent'* to another space when there is a common boundary between them. Two spaces are *'connected to'* each other when there is a navigable boundary like a door or a transition between them. A space is *'linked to'* a navigable boundary and a space is *'bounded by'* a non-navigable boundary. Other standard spatial relations [16] like *'intersects'*, *'touches'*, *'overlaps'* and *'within'*, existing between the entities are also given in Table 2. These primitive operators specified in this work and their definitions are given in Table 3. The notations are defined as follows: E refers to an entity, G_{Entity} to the geometry of an entity, S refers to space, NB to navigable boundary, NNB to non-navigable boundary and T refers to transition.

To demonstrate the model independence of these operators and for translating the proposed language to the existing models, there needs to be a correspondence between the operators defined in our framework and that of the existing frameworks. We show the corresponding operators in the existing modelling frameworks like PostGIS and BIM in Table 4. The correspondences, either direct or two-step correspondences are as shown in Table 4.

These operators have either direct or two level correspondence with the operators in existing query languages. The former include operators which have a direct correspondence with PostGIS and BIM operators like Intersect, Within, Touches and Contains. The latter represents the operators that are equivalent to a combination of operations in PostGIS and BIM. For instance the '*Connected*' operator is defined for spaces S_1 and S_2/ transition T_2 if both intersect the same navigable boundary in PostGIS. In BIM, IfcSpace_1 is connected to the IfcSpace_2/ IfcTransition_2 if both intersect a navigable boundary such as IfcDoor.

4.1 Proposed query language

Using the specification of entities and operators over these entities as given above, we now define the proposed indoor query language. The goal of this language is to cover the indoor domain specific queries irrespective of the underlying

Primitive operator	Format	Definition
Intersects	E_1 Intersects E_2	Returns true when the geometry of two entities intersect.
Touches	B_1/T_1 Touches B_2/T_2	Two entities(B/T) touch when their geometries have at least one common point but their interiors do not intersect.
Overlaps	S_1 Overlaps S_2	Two spaces overlap when their geometries have a common part but are not completely contained by each other.
Within	S_1 Within S_2	A space lies within another space when the geometry of the former lies completely inside that of the latter.
Boundedby	S Boundedby NNB	A space S is bounded by a non-navigable boundary NNB when their geometries intersect
Bounds	NNB Bounds S	A non navigable boundary bounds a space when they have intersecting geometries.
Linked	S Linked NB/T	A space has a linked relation to a navigable boundary or a transition T with which it intersects.
Adjacent	S_i Adjacent S_j	Two spaces are adjacent when they are linked or bounded by a common boundary.
Connected	S_i Connected S_j/ T_j	A space is connected to another space or transition when they intersect the same navigable boundary.

Table 3: Primitive operator definition

models, and to provide user friendly querying experience. It is designed so that a single query in the language can replace a set of multiple queries, in a model specific query language. For instance, to find all the entities which fall within a specific range around a given entity, the user has to write a function which consists of multiple sub-queries in existing query languages. In addition, the logic for range querying in 3d space has to be implemented by the user. This task can be simplified by a single range query syntax defined in the new language. This section explains the proposed query language and demonstrates how it achieves the above objectives.

Operator	PostGIS	BIM
One level correspondence		
Intersects	ST_Intersect (G_{E_1}, G_{E_2})	$IfcElement_1$ not *disjoint* $IfcElement_2$
Touches	ST_Touches (G_{B_1}/G_{T_1}, G_{B_2}/G_{T_2})	$IfcBuildingElement_1$ *touch* $IfcBuildingElement_2$
Overlaps	ST_Overlaps (G_{S_1}, G_{S_2})	$IfcSpace_1$ *overlap* $IfcSpace_2$
Within	ST_Within (G_{S_1}, G_{S_2})	$IfcSpace_1$ within $IfcSpace_2$
Two level correspondence		
Boundedby	ST_Intersect (G_S, G_{NNB})	IfcSpace *intersect* IfcWall/ IfcWindow
Bounds	ST_Intersect (G_{NNB}, G_S)	IfcWall/ IfcWindow *intersect* IfcSpace
Linked	ST_Intersect (G_S, G_{NB}) or ST_Intersect (G_S, G_T) and vice versa	IfcSpace *intersect* IfcDoor/ IfcStair and vice versa
Adjacent	ST_Intersect (G_{S_1}, G_{NNB}) && ST_Intersect(G_{NNB}, G_{S_2})	$IfcSpace_1$ intersect IfcStandardWallCase and IfcStandardWallCase intersect $IfcSpace_2$
Connected	ST_Intersect (G_{S_1}, G_{NB_1}) && ST_Intersect (G_{NB_1}, G_{S_2}/ G_{T_2})	$IfcSpace_1$ intersect IfcDoor intersect $IfcSpace_2$/ $IfcStair_2$

Table 4: Equivalence of primitive operators

The indoor query language, is defined to have an SQL like syntax, with clauses, predicates and expressions, since SQL like statements are easier to express.

Any query in this proposed query language has the format

`'Find indoor entity where conditions'.`

The *'Find'* clause contains the indoor domain specific entities or items to be selected. This is followed by an optional *'Where'* clause in which one or more conditions can be specified similar to an SQL query. The indoor queries can be broadly classified into the following types: attribute queries, spatial (e.g. adjacent, k-NN etc.) and geometric (e.g. finding area, volume).

Attribute queries select the indoor entities based on some operations on their attributes. An "attribute query" finds all entities E_i's of the specified type whose attributes satisfy the conditions given in the query. These queries can be specified as 'Find entity where *conditions*'. For example 'Find room where type='classroom'' is an attribute query.

Spatial queries deal with the spatial characteristics of the entities in the indoor space. Since indoor space is three dimensional, the spatial queries have to be modified to suit this space. They involve the use of spatial relations defined in the querying framework. For example, 'Find adjacent(SPACE)' , finds all the spaces which have a common boundary with this space. The common boundary can also be in the 'z' axis i.e., across floors. Similarly a range

Query category	Syntax	Definition	Description
Attribute query	Find E_i where E_i.Attr$_1$= value$_1$ && E_i.Attr$_2$ =value$_2$..	For every E_i , return E_i : { E_i.Attr$_1$= value$_1$ && E_i.Attr$_2$= value$_2$..}	Returns all entities of the specified type which satisfy the conditions specified on its attributes.
Adjacency query	Adjacent (S_i) where *conditions*	For every space S_j, return S_j, : { S_i adjacent S_j is true && *conditions* met}	Two SPACE entities are adjacent if they share a common boundary (navigable or non navigable) and all the conditions met.
Path query	Path (S_{start}, S_{end}/ T_{end}) [not] through E_i where *conditions*	Returns P ={ S_{start}, S_2, .. , S_{end}} such that for every S_i, S_i linked NB_i linked S_{i+1}/ T_{end} is true && E_i[not] in P && *conditions* met.	Returns the sequence of connected spaces or transitions between the start and end entities, with the conditions met.
Range query	Range (Type of entity, S_{origin}, *range value*) where *conditions*	For every E_i , return E_i :{ p= path(S_{origin} to E_i) exists && *conditions* met && length(p) $\leq$ *range value* }	An entity of the type specified selected if there is an accessible path which falls within the specified range meeting the conditions specified.
K-nearest neighbor query	Knn (Type of entity, S_{origin}, *k value*) where *conditions*	For every E_i , return E_i :{ p= path(S_{origin} to E_i) exists and E_i is not kth entity && *conditions* met}	Finds the first k entities of a given type at closest navigable distance from the queried entity and meet the conditions specified.
Volume	Find Volume(E_i)	Returns Volume(E_i)	Returns the volume of the specified entity calculated based on the geometry.

Table 5: Indoor query definitions

query aims to find all entities of a specified basic type (space, boundary or transition) that fall within a specific distance around a target entity. Here the range is based on navigable distance, rather than Euclidean distance due to obstacles in indoor space. The query format is ‘*Find range(entity type, entity, range value)*’. Similarly k-nearest neighbour queries return k entities, which are at the closest navigable distances from the origin entity.

Geometric queries deal with the geometric attributes of the indoor space entities, and they work based on the relationships between geometries. A query to find the volume of an entity also belongs to this category. The syntax for this query is defined as ‘Find volume(entity)’.

Navigation queries help find the path between two points in the indoor space. It can be specified by using the *‘Path’* keyword. The ‘*path query*’ finds a sequence of spaces { S_{start}, S_2, .. , S_{end}} where each consecutive pair of spaces in the sequence are connected to each other. A sequence of connected spaces from the origin space which lead to the end space are identified. In order to find the shortest path among these, the navigable distance between the spaces are given to A* or Dijkstra's based path finding modules [19]. These queries can include basic shortest path queries or constrained path queries, which specify constraints like paths without stairs etc. In addition, the query language can also specify queries which are a combination of above queries. The *‘Where’* clause can be used to combine different types of queries.

The query language syntax is defined based on the BNF grammar definitions of SQL(BNF Grammar for ISO/IEC 9075-2:2003- Database Language SQL(SQL-2003))[9]. The grammar defined for the query language is given in the following part. ANTLR parser generator[15] is used for constructing the compiler.

⟨*statement*⟩ ::=‘`Find`’ qstatement [wherestatement]

⟨*qstatement*⟩ ::= attributestmnt | adjstmnt | pathstmnt
| knnstmnt | rangestmnt | geomstmnt

⟨*attributestmnt*⟩ ::= space | boundary | transition

⟨*adjstmnt*⟩ ::= ‘`Adjacent`’ ‘(’space‘)’

⟨*pathstmnt*⟩ ::= ‘`Path`’ ‘(’entity‘`,`’entity‘`)`’ [passconstraint]

⟨*knnstmnt*⟩ ::= ‘`Knn`’ ‘(’etype‘`,`’entity‘`,`’kval‘`)`’

⟨*rangestmnt*⟩ ::= ‘`Range`’ ‘(’etype‘`,`’entity‘`,`’range‘`)`’

⟨*geostmnt*⟩ ::= ‘`Volume`’ ‘(’ entity ‘`)`’
| ‘`Area`’ ‘(’ entity ‘`)`’

⟨*passconstraint*⟩ ::= [‘`not`’] ‘`through`’ entity

⟨*wherestatemnt*⟩ ::= ‘`Where`’ ⟨*conditions*⟩

⟨*conditions*⟩ ::= orExpr

⟨*orExpr*⟩ ::= andexpr (‘`Or`’ andexpr)*

⟨*andexpr*⟩ ::= compexpr (‘`And`’ compexpr)*

⟨*compexpr*⟩ ::= atom (‘`Less`’ | ‘`Equal`’ | ‘`Grtr`’ atom)?

⟨*entity*⟩ ::= SPACE | BOUNDARY | TRANSITION

⟨*SPACE*⟩ ::= room | corridor

⟨*TRANSITION*⟩ ::= stair | elevator

⟨*BOUNDARY*⟩ ::= walls | windows | doors

⟨*etype*⟩ ::= ‘`space`’ | ‘`boundary`’ | ‘`transition`’

⟨*kval*⟩ ::= Num

⟨*range*⟩ ::= Num

⟨*atom*⟩ ::= (‘`a`’..‘`z`’ | ‘`A`’..‘`Z`’) | ‘`0`’..‘`9`’)+

⟨*Num*⟩ : ‘`0`’..‘`9`’+

The grammar defined for the language presents the syntax definitions for the specific query types which belong to the different indoor query categories. The syntax for each query has at least one basic entity in the indoor space among its parameters. The definition of each query made in terms of the basic entities and primitive operators that belong to the query framework developed in this work are as shown in Table 5.

As shown in the table, the language also supports additional constraints to be specified on some queries. For example, in path query, the user can specify constraints i.e., whether a path must or must not pass through an entity. This is specified through the ‘*passconstraint*’ part of the path query syntax. Additional conditions can be specified in the ‘where’ clause of each query. This helps provide completeness to the indoor query language. The query language as shown helps provide the user with a simple to use language where most indoor queries can be specified using simple constructs. To improve the user experience further, a visual query interface is proposed which is described next. The user inputs queries visually and these are converted to the above defined query syntax. The following subsection presents the visual composition of queries and the corresponding text-based queries generated in the language defined in this work.

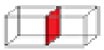
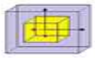
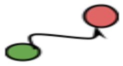
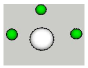
Visual operator	Querying mechanism	Equivalent text query
	Represents the adjacency of two spaces. This pointer is placed on the 3D entity for which adjacency is to be found.	Find adjacent (SPACE)
	This supports the range query. A resizable box is placed on the entity to be queried. Increasing/decreasing the pointer size sets the range.	Find range (Etype, SPACE, rangeval)
	This represents the path query. The user selects the start and end entities and a link is drawn between the entities for finding the path.	Find path (Estart,Eend)
	For the $k-NN$ query a sphere is first generated around the selected entity. Selecting the sphere generates one small sphere each time around the entity. The number of small spheres denotes the value of k.	Find knn(Etype, SPACE,k)

Table 6: Visual query primitives

4.2 Visual Query Language for Indoor Spaces

A visual querying interface helps improve the querying experience of a user, specially in the indoor space. The user makes visual interactions to construct the queries. This helps any user query the indoor space without need for an understanding of the underlying system or model. Additionally it reduces the chances of syntax errors being made when writing a text-based query.

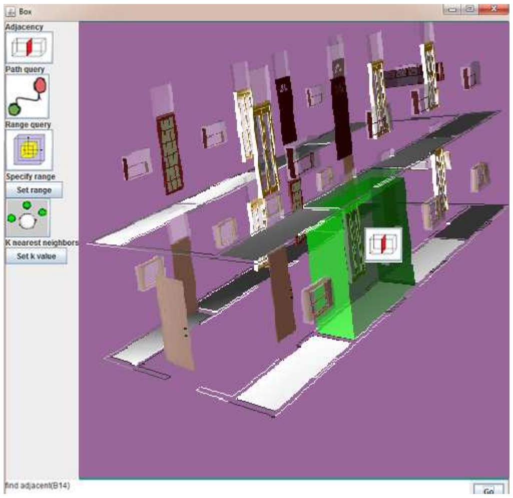
The visual querying module designed for the indoor querying framework is presented in Table 6. It constitutes 1) a 3D visualization of the indoor space, 2) visual primitives/ operators defined for a set of queries and 3) translation mechanisms to generate equivalent textual queries. The visual operators help allow the user to construct the queries in the indoor space. These operators include the basic operations like select, point etc, which are common to most visual query interfaces. In addition we define operators specific to the indoor space, and the Table 6 show these operators, their operation and their equivalent text command.

The visual operators are designed in such a way that they visually express their functionality in 3D indoor space. Most of them have been adapted from common visual operators used in current spatial systems. For instance, the operator for the k-nearest neighbor query is composed of a sphere surrounded by k-smaller spheres denoting the neighbors around an entity. Each operator visually explains its intended functionality. Queries like volume queries are defined using the simple select function and choosing options from the selected entity. For specifying constraints, menu buttons and text menus are provided.

Using these primitive visual operators a user can construct the desired indoor queries. To generate the equivalent text-based query, we need : 1) type of the query (e.g. adjacency, range query etc.), 2) entity on which the query is issued and 3) the associated parameters(e.g. value of range in range query, end entity in path query etc.). These are then substituted in the corresponding text query syntax. A visual query interface is designed, which allows the users to interact with the indoor space, and construct queries over it. Figure 2 shows a prototype visual query interface.

The interface consists of the operators panel, and the canvas showing the 3d visualization of the chosen building. The user selects the type of the query using the visual components (Table 6) presented in the top left side of the interface. Then an entity of interest is picked from the 3D visualization. The user now performs the interactions defined for each query (see Table 6).

The query type, entity chosen, and the constraints are then substituted in the textual syntax defined for each query. Since the visual query has a one-to-one correspondence with



```
find adjacent(B14)
```

Figure 2: Adjacency query constructed

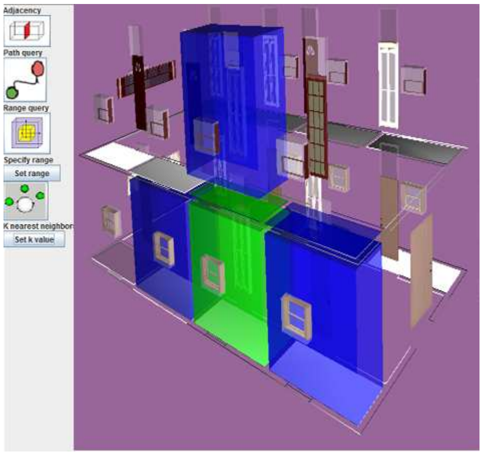


Figure 3: Adjacency query result visualized

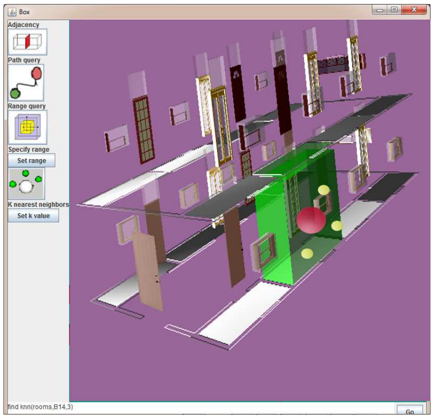
the textual query syntax, the conversion is done with simple substitutions of the details extracted from the visual interactions as mentioned previously.

Figure 2 shows the adjacency query constructed on a required indoor space. Selection on the visual primitives given on the left part of the interface is made to construct each query type. The corresponding text-based query is automatically generated and displayed in the text area at the bottom of the interface.

The resultant adjacent rooms of the given space computed from the indoor data models(spatial models stored in PostGIS, topology based models stored in Neo4j) are shown in the 3D visualization (Figure 3).

Figure 4 shows an example k-nearest neighbor query constructed. The user first selects the visual operator for the k nearest neighbor query. A sphere pointer is placed above the chosen entity, and as the user clicks the number of outer spheres increase, indicating the value of k. Figure 5 shows the rooms that are highlighted as a result of this operator.

We have described so far, a querying framework, a text-based query language and visual querying mechanism that



```
find knn(rooms,B14,3)
```

Figure 4: k-NN visual query constructed

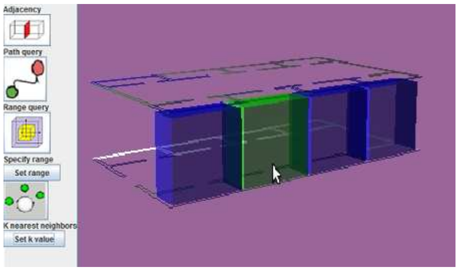


Figure 5: k-NN query result

have been developed based on this framework. The following sub-section deals with the evaluation of the proposed query framework.

4.3 Evaluating the query framework

The querying framework is evaluated based on a set of queries constructed in the proposed language and analysing the results obtained. This section explains the system environment and the use cases demonstrating the correctness and effectiveness of the proposed query framework.

4.3.1 System Environment

In order to evaluate the system, building's indoor models stored in PostGIS and Neo4j graph databases are used. The dataset consists of a simulated 20-storey building, where each floor has approximately 30 rooms.The database stores both the spatial and topological models of indoor space. The spatial model is represented as a set of tables storing the spatial and non spatial attributes of the indoor entities and is stored in the PostgreSQL database. The topological model consists of the following graphs namely adjacency, connectivity,and logical graphs with reference to the IndoorGML standard[11]. Each graph depicts the basic indoor entities as the nodes and a particular relation between them as the edges. The adjacency graph models the Adjacent_to relationships between the indoor entities, connectivity graph depicts the Connected_to relations and logical graphs model the access and temporal constraint information about the indoor entities. These graphs are generated as follows. Adjacency graph is generated by creating an Adjacent_to relation between the spaces that share a common boundary. Connectivity graph comprises of edges between nodes corresponding to the spaces or transitions which are linked to a common navigable boundary. Logical graphs are made by storing the user access types(normal or administrative) and temporal constraints(the open or closed status). These graphs are stored in the Neo4j graph database. The indoor querying system accesses both these databases.

4.3.2 Use cases

In order to evaluate the query language, we analyze the following a) model independence, b)correctness and c) completeness of the proposed query framework using use cases. Model independence is brought about when the queries can be specified irrespective of the underlying model, or language. Correctness is assured when the queries in the language yields results which match with the expected outcomes. Evaluation of completeness is to demonstrate that any query in the indoor domain can be issued using the language under consideration.

Now we present the set of use cases for performing evaluations in the above mentioned criteria. Table 7 presents use cases that evaluate the model independence of the querying framework. For each query in our language, equivalent queries exist in the SQL(for PostGIS) and cypher query language(Neo4j) to which it can be mapped. From the example queries in the tables we can see the following

- A query in PostGIS or Neo4j requires the user to understand the underlying database schema. In our language the user only has to enter the entity name, which is commonly used. The only constraint is that the name commonly used must be unique and an attribute in the database.

- Additionally a user should understand the spatial properties (property of r.geom) and how the geometrical operators like ST_Intersects work when querying the spatial model or the graph theoretical terminologies like nodes, and operators like "n-[:ADJACENT_TO]->m". In the proposed framework, the user just needs to understand the semantic aspects of indoor space and can be ignorant of the details of how it is represented.

These two aspects demonstrate the model independence of the query language.

In order to analyze the correctness of the query language, we have to assure that the structure of the query types is suitable for the underlying databases. This ensures that the query can retrieve the intended results by processing the indoor data which is available in the databases. The querying framework that we have defined in this work is based on the abstractions of different data models of indoor information. Each query is formulated based on these basic entities and operators which are abstracted out from the data models. Hence the queries will have a structure which is compatible to the data present in the databases. The queries also prevent the users from giving inputs that deviate from its defined structure or format. It does not allow the user to provide incorrect arguments to the queries and details which may not exist in the database. For instance, consider a query to find the adjacent rooms to a specific room and which are of type 'class room'. This query in textual form is represented as 'Find Adjacent(A19) where type='class room' '. The following sequence of interactions are accepted by our visual query interface to construct the query.

- The user picks the room on which the query is to be issued from the 3D visualization of the indoor space. This prevents the user from giving unavailable rooms or incorrect references to the rooms in the query.

- In order to state the condition that the adjacent rooms are to be of type 'class room', the user makes a selection from a drop down list which lists all the room types which are available in the database. This again prevents the user from giving a type which is unavailable.

Hence the query language prevents the user from constructing queries which are incorrect with respect to the indoor data stored in the databases.

The next set of use cases are provided to analyse the output of queries specified in the proposed Indoor query language. To analyze this, a set of queries (both simple and complex) in the indoor space are executed. The results obtained are compared with the ideal results. Tables 8 and 9 show the sample queries and their results, and corresponding expected results. Instances of attribute, adjacency, range, nearest neighbor and path queries are presented along with the results. These form basic set of queries which emerge in the indoor domain and hence are considered for the evaluation. For each query issued in the language, the expected and the obtained results are presented. The table shows that our queries retrieve the right results in various cases.

Finally we study the completeness of the proposed query framework. In this framework a set of basic queries have been defined. Any complex query can be formulated in terms of the basic query types defined. This ensures the completeness of the language. For instance, consider a query to find all rooms within 200 metres around a specific entity which are accessible to normal users and are open. This query can be constructed by enhancing the existing range query by adding conditions checking to the access types and the closed/ open status of the spaces. Consider another query for finding five nearest rooms which can accommodate at least 100 people, which may arise in the case of an educational institution. The above mentioned query can be written in our language as 'find knn (room, entity_id, 5) where capacity $\geq$ 100'. This indicates that new queries can be formulated and issued using the basic set of queries without changes to syntax.

5. QUERY TRANSLATIONS

We have proposed a model independent query framework and evaluated the query language in previous sections. One of the primary requirements in achieving this model independence is to be able to translate the query in the proposed language to any general purpose query language. Specifically, translations from visual to text-based and from text-based to the general purpose query languages supported by the modelling frameworks are required. This section details the translation mechanisms defined in the system.

The visual to text-based query translation involves simple substitutions. There exists a 1-1 correspondence between each text-based and visual query, as explained in the previous section, and hence the translation is as described

Query	PostGIS	Neo4j
Find Adjacent(A21)	SELECT r.roomname FROM bld1floor1rooms as s , bld1floor1walls as w, bld1floor1rooms as r, WHERE ST_Intersects (r.geom,w.geom) and ST_Intersects (w.geom,s.geom) and s.roomname='A21'	start n=node(*) match n-[:ADJACENT_TO]->m where n.roomname='A21' return n,m
Find range (rooms,B19, 200)	SELECT r.roomname, r.geom FROM bld1floor1rooms as s , bld1floor1doors as d, bld1floor1rooms as r, WHERE ST_Intersects (r.geom,d.geom) and ST_Intersects (d.geom,s.geom) and sum (ST_Distance (r.geom, d.geom), ST_Distance (d.geom, s.geom)) ≤ 200 and s.roomname= 'B19'	start n=node(*),m=node(*) match p= (n)-[r:CONNECTED_TO *..10]->(m) where n.roomname='B19' and sum (r.distance) ≤ 200 return m
Find rooms where type = 'conference halls' and floor=1	SELECT * from bld1floor1rooms where roomtype= 'conference halls'	Start n=node(*) match n.room n.type: 'conference hall', n.floor:'1' return n

Table 7: Evaluation of model independence

Query	Query in proposed language	Expected result	Obtained result
Find all restrooms in 2nd floor	Find rooms where type= 'restroom' and floor=2	(ID,name) (B5,Gents) (B8, Gents) (B15 , Ladies) (B29, Ladies)	(ID,name) (B5,Gents) (B8, Gents) (B15 , Ladies) (B29, Ladies)
To find all adjacent spaces of room P12	Find adjacent (P12)	(ID,name) (P6 , Room_6_6)	(ID,name) (P6 , Room_6_6)

Table 8: Evaluating querying framework's correctness

Query	Query in proposed language	Expected result	Obtained result
Find all adjacent class rooms to B4	Find adjacent (B19) where type='class room'	(ID,name) (B1 , IV IT A) (B3 , IV IT C)	(ID,name) (B1 , IV IT A) (B3 , IV IT C)
Finding path from B2 to B16	Find path (A21, B12)	{B2, CRB1, CRB2, CRB2, CRB2, B16 }	{B2, CRB1, CRB2, CRB2, CRB2, B16}
To find five nearest neighboring rooms of B4 of type 'classroom'	Find knn (rooms, B4, 2) where type='class room'	(ID,name) (B3 ,Room_3_3) (B5, Room_3_5) (B1, Room_3_1) (B12, Room_3_12) (B6, Room_3_6)	(ID,name) (B3 ,Room_3_3) (B5, Room_3_5) (B1, Room_3_1) (B12, Room_3_12) (B6, Room_3_6)
To find all rooms within 200m around A2	Find range (rooms, A2, 200)	(ID,name) (A5 ,CSE PI 2)	(ID,name) (A5 ,CSE PI 2)

Table 9: Evaluating querying framework's correctness (contd)

earlier. The translation of the text-based query to the existing general purpose languages is done by processing the abstract syntax tree (AST), which is generated while parsing the text-based indoor query. The nodes in the AST represent each construct in the input query. Parsing the input query involves a set of syntax rules matched from the syntax definition being invoked. Each rule invoked, triggers the generation of a subtree in the AST corresponding to the query. So the AST generation evolves through a sequence of syntax rule invocations.

In order to perform translation by processing the generated AST, the structure of the AST has to be defined and known. Each invoked syntax rule determines the structure of a part or a subtree of the query's AST. A syntax rule is made up of a set of constructs. In order to specify the structure of the AST's subtree generated by the rule, which construct forms the root node and which form the child nodes are defined. Consider a syntax rule with n constructs say $construct_1$, $construct_2$, .., $construct_n$. The structure definition format for the subtree to be generated is as given below.

$\wedge(construct_1\ construct_2\ ..\ construct_n)$

Here the first element after the $\wedge$ symbol indicates the root element. $construct_2$, .., $construct_n$ form its child nodes in the same order. For every syntax rule in the language's definition, a similar structure is defined. The evolution of an AST with the help of each syntax rule defined and the associated structure definition is presented below. Every query is made of a 'qstatement'(or query statement), and the 'wherestmnt' which indicates the 'where' clause in the query. The AST therefore has the 'Find' as root of the AST and the 'qstatement' and 'wherestmnt' become the children. The subtree with 'qstatement' as the root node, consists of the type(e.g. adjacent, range, knn etc.) as its children. Based on the query type, the subtree is chosen. Consider an input query type to be a 'range' query. The syntax for range query statement is given by

```
<rangestmnt> ::= 'Range' ( etype, querypoint, range)
```

The structure of the corresponding sub tree in the AST is specified as

```
^(Range  etype querypoint  rval)
```

Here 'Range' forms the root of the subtree and the parameters form the children. Here 'etype' indicating the type of the entities, 'querypoint' the entity on which the query is issued and 'rval' the range value form the children.

The input query may have a set of conditions given in a 'where' clause. The syntax of the 'where' clause in the language is given as follows.

```
<wherestmnt> ::= 'Where' attrcomp
```

Here the construct 'attrcomp' indicates one or more attribute comparisons of the form *'attribute comparison-operator value'* combined using 'and' or 'or' operators. The structure of the corresponding subtree in the AST is defined as follows

```
^(Where attrcomp)
```

Similarly for each rule in the language's syntax definition, a structure definition is provided in terms of its constructs.

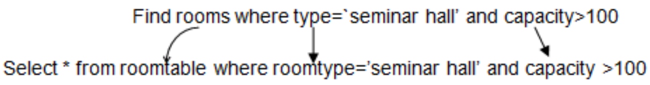
The AST generated is processed for performing the query translation. It is processed using a preorder depth first search traversal. We define an algorithm 1 which will specify the method of extracting the details from the abstract syntax tree. From any input query, the query type, the parameters and conditions given in the 'where' clause are extracted using this algorithm. Since the structure of the subtree corresponding to each syntax rule is defined, for each such subtree, we know which is the root node and in which order are the child nodes present. The algorithm presents, for each subtree of the AST defined, the method of extracting its details. For example, consider the method of processing the subtree corresponding to 'range' query type. Based on the structure defined, the first node is extracted as the type of entity, the second as the query origin and the third as the range value. All these details are then used to generate the general purpose queries namely SQL(for PostGIS) and cypher queries(for Neo4j). They are substituted to generate a single query as in attribute queries or used to invoke functions written in SQL or cypher query language in case of range, adjacent, k nearest neighbor and path queries.

An example of translating an attribute query from the proposed language to SQL is presented in figure 6. The entities and attributes specified are mapped to the entity and attribute names in the underlying schema to generate the SQL query. In some cases, converting these queries to the general purpose query languages involve invoking procedures corresponding to the query functions in the language.

Next we evaluate the designed translation mechanism by analysing the time incurred in a set of query translations.

```
Input: Node astRoot
Node N=astRoot;
if N.data='Root' then
    ProcessAST(N.getChild(0));
else if N.data='Find' then
    ProcessAST(N.getChild(0));
    ProcessAST(N.getChild(1));
else if N.data='where statement' then
    for Each child ch of N do
        Condition$_i$.operator=ch.data;
        Condition$_i$.attribute=ch.getChild(0).data;
        Condition$_i$.value=ch.getChild(1).data;
    end
    ConditionList.add(Condition_$i$);
else if N.data='Adjacent' then
    Entity= N.getChild(0).data;
else if N.data='Path' then
    startEntity= N.getChild(0).data;
    endEntity= N.getChild(1).data;
    if N.getChild(2) != null then
        passConstraint=N.getChild(2).data;
        passentity=N.getChild(2).getChild(0).data;
    end
else if N.data='Range' or N.data='knn' then
    EntityType= N.getChild(0).data;
    Entity= N.getChild(1).data;
    if N.data='Range' then
        rangevalue=N.getChild(2).data;
    else
        if N.data='Knn' then
            k=N.getChild(2).data;
        end
    end
else
    EntityType=N.getChild(0);
end
```

Algorithm 1: ProcessAST



```
Find rooms where type=`seminar hall' and capacity>100
Select * from roomtable where roomtype='seminar hall' and capacity >100
```

Figure 6: Attribute query generated for PostGIS

Table 10 presents the results of this analysis. Here t_1 represents translation time incurred in our language, t_2, the execution time of a query in our language, and t_3, the execution time of a query in PostGIS/Neo4j. It also shows for each query in the proposed language, the translated queries in the general purpose query languages. The translation proceeds as mentioned previously by traversing the abstract syntax tree constructed.

It can be observed that the translation times are considerably smaller and hence do not cause much overhead to the entire query execution. The extra translation part which exists in the system does not affect the total execution time of the query. To determine if this performance holds for a larger set of queries, we evaluated the translation time for 100 different queries. The average query translation time

Query	Translated query	Evaluation time (sec)
Find adjacent (A21)	SELECT r. roomname FROM bld1floor1rooms as s , bld1floor1walls as w, bld1floor1rooms as r, WHERE ST_Intersects (r.geom, w.geom) and ST_Intersects (w.geom, s.geom) and s.roomname =‘A21’	$t_1=$ 0.02248, $t_2=$ 0.09485, $t_3=$ 0.07237 (PostGIS)
Find range (rooms, B19, 200)	start n=node(*), m=node(*) match p= (n)-[r: CONNECTED _TO *..10]-(m) where n. roomname=‘B19’ and sum (r. distance) $\leq$ 200 return m	$t_1=$ 0.02276, $t_2=$ 0.93542, t_3 0.91261 (Neo4j)
Find path (B14,B2) where length $<$ 400	start n=node(*), m =node(*) match p= (n)- [r: CONNECTED_TO * .. 10] -(m) where n. roomname =‘B14’ and n. roomname =‘B2’ and sum (r. distance) $<$ 400	$t_1=$ 0.02851, $t_2=$ 0.93557, t_3 0.90704 (Neo4j)

Table 10: Translations and evaluations

was found to be 0.0207 seconds, with a standard deviation of 0.0046 seconds. This demonstrates that the translation component is efficient and does not degrade the querying system’s performance.

6. CONCLUSION AND FUTURE WORK

In this paper we propose a model independent querying system for indoor spaces. We have developed a querying framework which abstracts out and represents the common features of the underlying models. Based on this querying framework, a text-based and visual querying languages are developed. Visual querying enhances the ease of querying the indoor data models. Translation modules are defined for converting queries in the proposed query language to the general purpose query languages(SQL and Neo4j) supported by the models. This allows the system to be used above multiple models. Evaluations of the querying framework developed and the translation mechanisms demonstrate the completeness, correctness and model independence of this framework. The future work on this system include defining more queries in the visual querying system and adding support for other modelling frameworks like BIM and IndoorGML. Additionally user studies for evaluating the visual querying and improving it is part of ongoing work.

7. REFERENCES

[1] Neo4j graph database. http://www.neo4j.org/, November 2013.

[2] Blaser, A. D., and Egenhofer, M. J. A visual tool for querying geographic databases. In *Proceedings of the Working Conference on Advanced Visual Interfaces* (2000), AVI ’00, ACM, pp. 211–216.

[3] Borrmann, A., and Rank, E. Topological analysis of 3d building models using a spatial query language. *Adv. Eng. Inform 23* (2009), 370–385.

[4] Boucelma, O., Essid, M., Lacroix, Z., Vinel, J., Garinet, J.-Y., and Betari, A. Virgis: mediation for geographical information systems. In *Data Engineering, 2004. Proceedings. 20th International Conference on* (March 2004), pp. 855–.

[5] Chintaphally, V., Neumeier, K., McFarlane, J., Cothren, J., and Thompson, C. Extending a natural language interface with geospatial queries. *Internet Computing, IEEE 11* (2007), 82–85.

[6] Chuck Eastman, Paul Teicholz, R. s. A guide to building information modelling for owners, managers, designers, engineers and contractors, 2011.

[7] Claus Nagel, Thomas Becker, R. K. Requirements and space-event modelling for indoor navigation, 2010.

[8] Egenhofer, M. Constraint qualifications in maximization problems. *Knowledge and Data Engineering, IEEE Transactions on 6* (1994), 86–95.

[9] IEC, I. Bnf grammar for iso/iec 9075-2:2003. http://savage.net.au/SQL/sql-2003-2.bnf, December 2013.

[10] Jensen, C. S., Lu, H., and Yang, B. Indexing the trajectories of moving objects in symbolic indoor space. In *Proceedings of the 11th International Symposium on Advances in Spatial and Temporal Databases* (Berlin, Heidelberg, 2009), SSTD ’09, Springer-Verlag, pp. 208–227.

[11] Jiyeong Lee, Ki-Joune Li, S. Z. Open geospatial consortium inc. indoorgml draft,reference no: Ogc 13-nnnrx, 2013.

[12] Mazairac, W., and Beetz, J. Bimql - an open query language for building information models. *Adv. Eng. Inform 27* (2013), 444–456.

[13] Morris, A. J., Abdelmoty, A. I., and El-Geresy, B. A. A visual query language for large spatial databases. In *Proceedings of the Working Conference on Advanced Visual Interfaces* (New York, NY, USA, 2002), AVI ’02, ACM, pp. 359–360.

[14] Ooi, B.-C., Davis, R., and McDonnell, K. Extending a dbms for geographic applications. In *Data Engineering, 1989. Proceedings. Fifth International Conference on* (Feb 1989), pp. 590–597.

[15] Parr, T. Antlr. http://www.antlr.org/, December 2013.

[16] Research, R. Postgis. http://postgis.net/, September 2013.

[17] Sela Mador-Haim, Yoad Winter, A. B. Controlled language for geographical information system queries. In *Proc. of Inference in Computational Semantics.* 2006.

[18] Tennant, H. R., Ross, K. M., and Thompson, C. W. Usable natural language interfaces through menu-based natural language understanding. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 1983), CHI ’83, ACM, pp. 154–160.

[19] Thomas H. Cormenn, Charles E. Leiserson, R. L. R. C. S. *Introduction to Algorithms (3rd edition).* MIT Press, 2009.

Distributed Elastic Net Regularized Blind Compressive Sensing for Recommender System Design

Anupriya Gogna
IIIT-Delhi
Delhi, INDIA
anupriyag@iiitd.ac.in

Angshul Majumdar
IIIT-Delhi
Delhi, INDIA
angshul@iiitd.ac.in

ABSTRACT

Design of recommender system following the latent factor model is widely cast as a matrix factorization problem yielding a rating matrix, which is a product of a dense user and a dense item factor matrices. A dense user factor matrix is a credible assumption as all users are expected to have some degree of affinity towards all the latent factors. However, for items it's not a reasonable supposition as no item is expected to possess all the traits (factors). In this work, we propose a matrix factorization model which yields a dense user but a sparse item factor matrix; having equivalence to Blind Compressive Sensing (BCS) formulation. Basic BCS framework is augmented with an added elastic net regularization term. The addition helps in capturing correlation between different item latent factors. Despite the efficiency of matrix factorization approach, it's not feasible to apply the techniques for very large datasets (rating matrices). For this purpose, we employ Divide and Combine (DnC) approach – wherein proposed method is applied to distinct subsets of the rating matrix simultaneously and resulting estimates combined to yield the final result. The (randomized) DnC approach retains the convergence guarantees of matrix factorization. Experiments were conducted on real world Movielens dataset and our technique was compared against popular matrix factorization methods. The results indicate the superiority of our method in terms of both accuracy and speed.

Keywords

Blind compressive sensing, collaborative filtering, elastic net regularization, latent factor model.

1. INTRODUCTION

Information overload on the internet regards to the number of items, services, service providers and even reviews makes the job of finding the desired cumbersome for any customer. With the advent of Recommender Systems (RS), in 1990's [30], [36], this task is considerably eased. An efficient Recommender System helps both the customers – by providing relevant suggestions, as well as e-commerce portals (like Amazon, Flipkart) – by increasing their popularity and hence revenue. Task of a RS is to predict a user's choice based on his/her past history and make relevant recommendation of products and services for future.

Methods for design of RS can be classified on the basis of information utilized and technique adopted for rating prediction into – Content based, Collaborative filtering and hybrid techniques [1]. Content based methods [32] are based on finding similarity/match between user's choice profile and the content description of items. Collaborative filtering (CF) techniques [18], [14] rely on either implicit ratings – inferred from users past behavior such as browsing history, or on explicit ratings – ratings given by users on a small subset of items. They are the most widely used and efficient means of design of recommender systems. Unlike content based methods, they do not require any explicit characterization of items or users; which might not be always feasible. Hybrid schemes employ a combination of both [28].

CF methods can be further subdivided into memory based and model based approaches. Memory based methods [35], [36] are primarily neighborhood based strategies. They scan the entire rating matrix to find users with high similarity (measured based on ratings on commonly rated items) to the target user. Predicted rating for an item is just a linear combination of ratings given by similar users on the concerned item. The approach can be extended to work on item similarity rather than user similarity [22]. These methods are more intuitive, but lack the desired speed of computation; due to large size of the rating matrix. Also, the sparsity of rating matrix makes finding similar users difficult at times, which prohibits predicting ratings especially for a new user – the cold start problem [1].

Model based methods [39], [42] on the other hand, construct a model from existing dataset and subsequently use it for rating prediction. The lower dimension of the model viz-a-viz original database makes them suitable for faster online computations. Also, they are able to provide better coverage and prediction accuracy than their memory-based counterparts [1]. Several model based approaches have been studied such as Bayesian probabilistic modelling [37], cluster based methods [42], and latent factor models [19], [39].

Latent factor models have gained tremendous popularity over the past decade. These models rest on the premise that a user's choice of an item is governed by the traits possessed by the item and the user's affinity towards those characteristics. Every user can be profiled as vector of his/her affinity to certain characteristics or latent factors. Similarly, every item can also be profiled by a vector describing the extent to which it possesses those latent factors. User's rating on an item are a result of interaction between these latent factor vectors.

The 20th International Conference on Management of Data (COMAD),
17th-19th Dec, 2014 at Hyderabad, India.

Latent factor based approach has been conventionally cast as a matrix factorization problem – representing the rating matrix as a product of user and item latent factor matrices [19]. Recently, in some works [20], [38], the latent factor model has been cast as a (low rank) matrix completion task. It's a convex formulation unlike matrix factorization which is bilinear and hence non-convex. But, use of singular value decomposition makes these algorithms computationally too intensive to allow wide spread application in RS design.

Our proposed approach is based on matrix factorization formulation, which recovers the latent factor matrices for users and items. Existing works on matrix factorization aims to recover a dense user and item latent factor matrices [34]. It's realistic to expect a dense user latent factor matrix, as all users will have some degree of affinity towards all latent factors. However, such a scenario is not correct with respect to items. For example, consider the case of a Music recommender system. A user will have certain degree of interest towards all forms, be it Bollywood, Ghazals or Rap. Similarly, he/she may have a favourite singer, but will not be averse to listening to others. All this will translate into dense latent factor vectors for the users. However, a song cannot simultaneously belong to all genres or be sung by all singers. Thus the latent factor vector for any song, will have very few non-zero values – indicating possession of a few of the entire list of latent factors. Following this proposition, we formulate a matrix factorization approach that promotes recovery of the rating matrix as a product of a dense user latent factor matrix and a sparse item latent factor matrix. Our formulation shows equivalence to Blind Compressive Sensing (BCS) framework [11] in signal processing.

In addition to being sparse, item latent factors also exhibit some correlation amongst themselves. For example, an album by Lady Gaga will invariably be Pop. This correlation can be captured by an elastic net [45] type penalty term added to our base matrix factorization formulation. Thus our framework resembles a BCS formulation with an additional elastic net penalty.

The application of any matrix factorization algorithm to huge datasets (with rating matrix dimension exceeding tens of thousands) is not a feasible scenario. In [26] authors proposed a Divide and Conquer Approach which can be used to apply MF algorithms to huge datasets. We extend the approach to our formulation in order to generate predictions for very big datasets.

The results obtained using our algorithm are compared against those obtained using existing state of the art formulations. Our method yields better results than the techniques compared against with regards to both recovery accuracy and execution time – important aspects of RS design.

The rest of the paper is organized as follows. Section 2 provides brief description of work done in related areas. Our proposed formulation is elaborated in section 3. Section 4 includes the experimental setup and results. Conclusion and future work are presented in section 5.

2. LITERATURE SURVERY AND PRELIMINARIES

2.1 Latent Factor Model

Actual ratings available in the database are influenced by not just the liking of a user towards the traits possessed by an item, but are also impacted by certain biases embedded in both users and items. If we consider a user who is a movie enthusiast, he will probably be fervent about all movies and rate all of them generously. Such a user ends up having a positive user bias. Similarly, a movie which is a big Oscar Awardee will tend to get higher ratings by almost all users, inflicting it with a positive item bias.

The bias terms constitute the baseline estimate which can be modeled as [19]

$$baseline(u_a, i_b) = m_g + b_u(u_a) + b_i(i_b) \tag{1}$$

where, m_g is the global mean, $b_u(u_a)$ is the user bias for user '*a*' and $b_i(i_b)$ is the item bias for item '*b*'.

The interaction component of the actual ratings, i.e. excluding the baseline estimates, is what can be modelled as in terms of user's affinity to traits possessed by the item. The challenge in RS design is modelling the interaction component; baseline estimation is easy. Modelling this component using latent factor based design rests on the suggestion that that a users' rating of an item is a function of his/her affinity towards the traits (latent factors) possessed by the target item. For example, consider the case of a book recommendation system. Each user's choice of a book can be defined in terms of choice of category (fiction/nonfiction etc.), author and certain other related features. Similarly a book will have these characteristics (latent factors) to varying extent. The interaction between the user and item can be modelled as the interaction between these feature vectors for books and users as in (2)

$$interaction(u_a, i_b) = \left\langle f_{u_a}, f_{i_b} \right\rangle \tag{2}$$

where, f denotes the latent factor vector. Actual ratings can be considered as a combination of interaction and baseline measures.

$$R(u_a, i_b) = baseline(u_a, i_b) + interaction(u_a, i_b) \tag{3}$$

Extending (2) and (3) to entire rating matrix, we can model the rating matrix, R, as

$$R = m_g I + B_u + B_i + F_U \times F_I \tag{4}$$

where, B_u / B_i *and* F_U / F_I are the matrix counterpart of the user/item bias and latent factor vectors respectively.

The observed rating matrix Y given by

$$Y = M(R) \tag{5}$$

where, M is the masking/subsampling operator. Only a small percentage of total ratings are available in the database, i.e. Y is extremely sparse. The task in Collaborative filtering is to predict the missing ratings and fill in the rating matrix.

Most frequently used method of rating prediction using (5) is Matrix Factorization (MF) [19] - involving solving an optimization problem of the form

$$\begin{aligned} \min_{B_u, B_i, F_U, F_I} & \left\| Y - M(m_g I + B_u + B_i + F_U \times F_I) \right\|_F^2 \\ & + \lambda_b \left(\|B_i\|_F^2 + \|B_u\|_F^2 \right) + \lambda_i \left(\|F_U\|_F^2 + \|F_I\|_F^2 \right) \end{aligned} \tag{6}$$

where, λ_b and λ_i are the regularization parameters which aid in preventing over fitting of model to observed data. Equation (6) is a non-convex formulation, but with separable variables. This enables minimizing over each of the variables alternately using alternating least squares [3] or stochastic gradient descent algorithm [47].

Because of efficiency of MF approach, it has received lots of attention and several works [20], [29], [39] have proposed algorithms to solve the same.

A recent addition to solving latent factor model based formulations is Matrix Completion (MC) approach. MC formulation aims to recover the interaction model (W) directly, instead of its factored version $F_U \times F_I$ by solving expression of the form given below

$$Q = M(W) \tag{7}$$

where, Q is the interaction component of observed set of ratings. Given the subsampling nature of masking operator M, (7) is an underdetermined linear system of equation. However, we can aim for a unique solution if we place a constraint on W [12]. In case of RS, the overall interaction component is affected by only the latent factors (which are the independent variables). As the number of latent factors (generally around 40-50), is far less than dimension of rating matrix (even reaching hundreds of thousands), W has a significantly low rank structure. Thus, we can look for the lowest rank solution. Hence, MC problem can be cast as

$$\min_W \left\| Q - M(W) \right\|_F^2 + \lambda_n rank(W) \tag{8}$$

where, λ_n is the regularizing term penalizing any deviation in W from the low rank nature. However, rank minimization is a NP-hard problem [2http], and thus any algorithm for the same has complexity of a brute force algorithm. Hence, the rank constraint can be replaced by its convex hull, nuclear norm – sum of singular values – constraint as in (9) [5].

$$\min_W \left\| Q - M(W) \right\|_F^2 + \lambda_n \left\| W \right\|_* \tag{9}$$

Nuclear norm regularization term in (9), while maintaining convexity of the formulation, promotes recovery of a low rank solution [33]. Several solvers exist for (9) [4], [25], [41]. Although MC being convex has convergence guarantees, it is not widely used in RS design. This is so because most existing solvers use Singular value Decomposition (SVD) for solving (9), which because of its high complexity, is inefficient for very large rating datasets.

2.2 Compressed Sensing

Theory of Compressed Sensing (CS) focusses on recovery of a sparse signal from its lower dimensional projections [8]. If y is the observation vector, and x is the original signal, given the (linear) projection operator A, the three are related as

$$y = Ax \tag{10}$$

If A has a dimension of $m \times n; m \ll n$, then (10) being an under determined system of equation can have infinite solutions. According to CS theory, if the signal '*x*' is sparse or adequately compressible, a unique solution to (10) can be obtained [2] by looking for the sparest solution (minimizing l_0 norm), i.e. solving problem of the form

$$\min_x \|x\|_0 \text{ subject to } y = Ax \tag{11}$$

Most signals are often not themselves sparse, but sparse in some transform domain (for example images are sparse in wavelet domain). In such a case, we can modify (11) as follows.

$$\begin{aligned} &\min_x \|\beta\|_0 \text{ subject to } y = AD^T\beta \\ &\text{where, } x = D^T\beta \end{aligned} \tag{12}$$

where, D is the sparsifying dictionary.

However, (12) is NP-hard [31], with all algorithms to solve the same having complexity equal to brute force algorithms. But, non-convex l_0 norm can be approximated by its convex surrogate, the l_1 norm (13), which yields the same solution as (12) if certain conditions are met [6].

$$\begin{aligned} &\min_x \|\beta\|_1 \text{ subject to } y = AD^T\beta \\ &\text{where, } x = D^T\beta \end{aligned} \tag{13}$$

Equation (13) can be put as an unconstrained convex formulation

$$\min_\beta \left\| y - AD^T\beta \right\|_2^2 + \lambda \|\beta\|_1 \tag{14}$$

where, λ is the regularization parameter. It has been shown and if A and D are incoherent and *x* is sufficiently sparse, solution of (12) and (14) match.

2.3 Blind Compressed Sensing

CS theory assumes the either the signal is sparse as it is, or sparse in some known transform domain i.e. sparsifying dictionary D is known a priori. But, there could be cases where that is not the case. Such problems fall under the framework of Blind Compressed Sensing (BCS) [11].

BCS formulation attempts to simultaneously recover the sparse signal and the sparsifying basis from the under sampled signal measurements. However, a robust solution to this is possible only in case of Multiple Measurement vector (MMV) setting depicted in (15). Consider multiple observation vectors stacked in columns of Y and B being the sparse coefficients matrix, then

$$Y = AD^T B \tag{15}$$

where, $Y = [y_1 \mid y_2 \mid .... \mid y_N]$ and $B = [\beta_1 \mid \beta_2 \mid .... \mid \beta_N]$

Even though, BCS shows similarity to dictionary learning, latter is an offline technique, i.e. it cannot be used for signal recovery/ reconstruction.

To obtain a unique solution, it's necessary to impose some constraint on the sparsifying basis also, alongside the sparsity constraint on the transformed signal coefficients [11]. One of the possible formulation for BCS, used in [23] for dynamic MRI reconstruction, is given in (16). It imposes a constraint on the Frobenius norm of the dictionary.

$$\min_{\beta, D}\left[\sum_{i=1}^{l}\left\|A_i(\beta D)-y_i\right\|_2^2\right]+\lambda\|\beta\|_1 \qquad (16)$$
$$\text{subject to } \|D\|_F^2 \le const$$

2.4 Elastic-net Regularization

Classical regression model can be written as

$$y = Ax + \eta,\ \ \eta \sim N(0,\sigma^2) \qquad (17)$$

where, *y* is a observed data, *A* is a matrix of explanatory variables and *x* is the unknown weight vector which explains the observation in terms of explanatory variables.

The most basic and straight forward regularization is the ridge regression [15] which solves the problem of the form

$$x_r = \min_x \|y - Ax\|_2^2 + \lambda\|x\|_2^2 \qquad (18)$$

Where, λ is the regularization parameter. Solving (18) promotes recovery of a dense solution because of l_2 norm penalty term.

However, in cases where only a few explanatory variables explain the entire formulation, ridge regression fails to capture the structure correctly. In such a case, we want a sparse solution (weight vector) *x* so that only a few explanatory variables participate in the describing the observed variable. Here comes the LASSO (Least Angle Shrinkage and Selection Operator) regularization [40].

LASSO solves (17) with the l_2 norm penalty term replaced by l_1 regularization (19)

$$x_l = \min_x \|y - Ax\|_2^2 + \lambda\|x\|_1 \qquad (19)$$

Use of l_1 penalty promotes recovery of a sparse weight vector.

But, in certain cases even with the desired weight vector being sparse, LASSO fails to yield the correct structure. For example, consider a case where the explanatory variables are interdependent or highly correlated. In this scenario, the correlated explanatory variables should occur together, i.e. if one of them is selected, others in the group should also be a part of selection. However, LASSO fails to capture or promote this group structure.

This is where, the role of elastic net regularization [27], [45] comes into play. It includes an additional penalty term (l_2 norm) on the weight vector into the LASSO framework. This quadratic penalty aims at selecting all the correlated variables together. Equation (20) shows elastic net regularization.

$$x_{enet} = \min_x \|y - Ax\|_2^2 + \lambda_1\|x\|_1 + \lambda_2\|x\|_2^2 \qquad (20)$$

2.5 Divide and Conquer

Most e-commerce sites have very large database of users and items; generating a huge but extremely sparse rating matrix. Applying any algorithm to such huge matrices is almost computationally impossible because of its enormous time and space complexity.

Research has been undertaken to solve this scalability problem and several approaches have been suggested. In [46] authors proposed a parallel ALS algorithm. In this, various “labs” in parallel MATLAB work on a subset of columns of the rating matrix and the resulting estimates are shared with other “labs”. [24] proposed an approach for performing nonnegative matrix factorization as a series of map and reduce steps.

Authors in [26] proposed a distributed approach for performing MF on large datasets on a distributed architecture. It follows three steps

1. Divide: Divide the observed rating matrix (Y) into sub matrices - $M_1, M_2, ..., M_N$, by splitting Y along the longer dimension.
2. Factor: Perform independent MF on each of the sub matrices, to yield partial estimates corresponding to each sub matrix - $\hat{M}_1, \hat{M}_2, ...\hat{M}_N$.
3. Combine: A technique using randomized column projection method suggested in [10] is used. One of the estimated sub matrix is selected at random, and all sub matrix estimates are projected on to its column space to yield a (low rank) estimate for the entire rating matrix. Same procedure is implemented for all the sub matrices and an average of all resulting estimates yields the final matrix factors.

This methodology is implemented on a distributed platform, with split and factorization algorithms running simultaneously on all sub matrices in parallel. It achieves a huge decrease in run time and also reduces net computational complexity.

3. PROPOSED APPROACH

3.1 Proposed Formulation

In this section, we present our novel proposition for latent factor model based design of an efficient recommender system.

For our model, we first estimate the baseline offline. Baseline estimation is done using stochastic gradient descent algorithm for solving the formulation in (21).

$$\min_{b_u, b_i} \sum_{u,i\in\Omega}\left(y_{u,i} - \left(b_u + b_i + m_g\right)\right)^2 + \lambda_b\left(b_u{}^2 + b_i{}^2\right) \qquad (21)$$

where, Ω is the set of observed ratings.

For our design, offline baseline estimation not only reduces the online computation burden substantially, but also gives better recovery accuracy than online baseline estimation for our model. Once the baseline is computed by solving (21), the interaction part is segregated from the actual ratings. Our model is applied to this ‘interaction component’. After the interaction estimate for the entire matrix is computed, baseline terms are added back to get final rating values for making relevant prediction.

Modelling of the interaction part is done following the latent factor based design approach. In line with the conventional latent factor model, we also propose to factorize the rating matrix into two sub matrices – user latent factor and item latent factor. Existing latent factor matrix factorization models solve the problem of the form

$$\min_{F_U, F_I}\|Q - M(F_U \times F_I)\|_F^2 + \lambda\left(\|F_U\|_F^2 + \|F_I\|_F^2\right) \qquad (22)$$

where, $Q = Y - B_u - B_i - m_g I$ is the observed interaction part, F_U and F_I are the matrix composed of user and item latent factor vectors, respectively and M is the subsampling operator. Above optimization problem promotes recovery of a dense item and a dense user latent factor matrix i.e. latent factor vectors for both users and items are dense, with non-zero values for all latent factors.

A user latent factor vector can be reasonably assumed to be dense, but the same cannot be applied to the item's latent factors. Let us consider the case of a Restaurant recommender system. In this case the relevant latent factors (defining characteristics) include those related to cuisine, location, price, ambience, service and alike. A user might have a liking for continental cuisine but will be completely against Indian food. Similarly a user having an affinity for fine dining restaurants, might not be opposed to going to a self-service café. Hence, it can be safely presumed that a user's affinity to almost all factors, to varying degree, will translate into a dense user factor matrix. On the other hand, if a restaurant is fine dining, it cannot have self-service. Similarly a bakery won't serve Indian cuisine. Hence, if we construct a restaurant's latent factor vector, it will have a large number of zeros, as no restaurant can possess all latent factors concurrently. Hence, the dense latent factor assumption does not hold true for items.

In contrast to previous works, we propose to factorize the rating (interaction) matrix into a dense user factor matrix and a sparse item latent factor matrix. The problem can be mathematically formulated as

$$\min_{F_U, F_I} \left\| Q - M(F_U \times F_I) \right\|_F^2 + \lambda_u \left\| F_U \right\|_F^2 + \lambda_i \left\| vec\left(F_I\right) \right\|_1 \qquad (23)$$

Where, $vec\left(F_1\right)$ is the vectorized (column concatenated) form of item factor matrix. Equation (23) has equivalence to blind compressed sensing formulation (16) discussed in the previous section. The Frobenius norm penalty on the user matrix is in accordance with the constraint on dictionary in BCS framework. Together with the sparsity constraint on item factor matrix it provides conditions for recovery of a unique solution.

Our above formulation (23) captures the sparse nature of item latent factor vectors but fails to capture the dependence of these factors on each other. Carrying on with the case of a restaurant RS, a fine dining restaurant will inevitably be expensive, as also will be a restaurant in a star property. It can be observed that certain latent factors are linked together, i.e. they will usually occur together. Hence, item latent factor vectors follows a group sparse structure. Equation (23) may select certain factors, but keep related latent factors as zero, as it fails to exploit their correlation. But, the lack of knowledge about which factor (position of latent factor in the entire vector) corresponds to which trait prevents us from imposing a strict group sparse penalty.

Elastic net regularization allows us to embed this group nature and inter factor dependence into the matrix factorization model. Incorporating elastic net type penalty term into our previous formulation we get

$$\min_{F_U, F_I} \left\| Q - M(F_U \times F_I) \right\|_F^2 + \lambda_u \left\| F_U \right\|_F^2 + \lambda_i \left\| vec\left(F\right)_I \right\|_1 + \lambda_{enet} \left\| F_I \right\|_F^2 \quad (24)$$

Inclusion of both l_1 and l_2 norm penalty on the item latent factor matrix promotes recovery of a sparse solution with correlated factors being chosen together.

To enable efficient implementation of our approach to very big datasets, we place it in the structure of Divide and Combine methodology [26]. The observed interaction (component) matrix (Q) is split into several disjoint sub parts - smaller dimension matrices - by splitting randomly along the longer matrix dimension. Our latent factor based approach is then applied to each of the sub matrices. After all the partial estimates are obtained, they are combined in accordance with the procedure outlined in preceding sections. This approach helps us in efficient parallel implementation of our algorithm on a distributed platform.

3.2 Algorithm Design

In this section we present the design of an algorithm with low computational complexity to solve (24). Our algorithm is based on the principle of Majorization Minimization (MM) [9].

Majorization Minimization scheme proposes to map computationally intensive optimization problems into much simpler and effective iterative procedures. We briefly discuss the MM approach before using it for our algorithm design.

In several applications, we are required to solve least square optimization of the form

$$\min_x \left\| y - Ax \right\|_2^2 \qquad (25)$$

The solution to above is given by $x = \left(A^T A\right)^{-1} A^T y$.

If the size of signal x, is very large, computation of pseudo inverse of A is computationally very intensive – forming the algorithm's bottleneck. MM technique eliminates this bottleneck. It involves replacing the existing function $h\left(x\right) = \min_x \left\| y - Ax \right\|_2^2$, by another function $g\left(x\right)$ which is much simpler to minimize. It is essentially a majorizer of $h\left(x\right)$ and the two are related as follows

- $g(x_k) = h(x_k)$
- $g(x) \geq h(x) \forall x$

As shown in fig. 1, new function is defined such that it touches the existing function at the point of definition, and lies above it otherwise.

For (25), at an initial (guess of minima) point x_k , $g\left(x_k\right)$ can be defined as

$$g(x_k) = \left\| y - Ax \right\|_2^2 + \left(x - x_k\right)^T \left(\gamma I - A^T A\right)\left(x - x_k\right) \qquad (26)$$

Under the constraint that $\gamma \geq \max eig\left(A^T A\right)$, (26) will satisfy conditions for majorizer. Now, instead of minimizing our original function, (26) is minimized. Its minima forms the new point of definition $x_{k+1} = \min_x g\left(x\right)$.

After some mathematical manipulations, minimization of $g\left(x\right)$ can be converted into two iterative steps

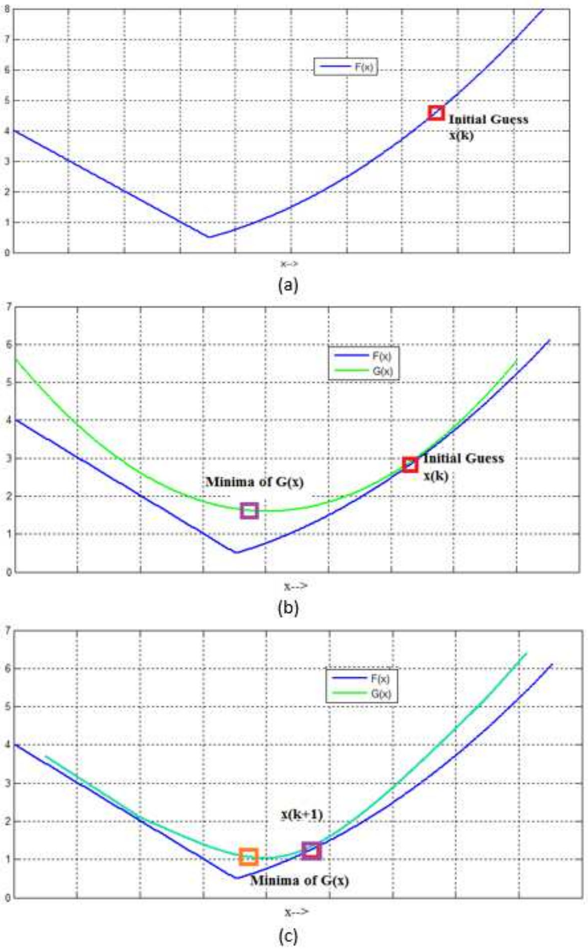


Figure 1. Majorization – Minimization Approach

$$Step\ 1:\ b = x_k + \frac{1}{\gamma}A^T\left(y - Ax_k\right) \tag{27a}$$

$$Step\ 2:\ \min_x \|b - x\|_2^2 \tag{27b}$$

Hence, the solution to (25) no longer requires pseudo inverse computations.

Before extending the same methodology to our formulation (24), we apply Alternating Direction Method of Multipliers (ADMM) to (24), to split this bilinear formulation into two convex sub problems (as both variables are separable) – one optimizing over F_U and other over F_I .

Sub problem 1:

$$\min_{F_U} \|Q - M(F_U \times F_I)\|_F^2 + \lambda_u \|F_U\|_F^2 \tag{28}$$

Sub problem 2:

$$\min_{F_I} \|Q - M(F_U \times F_I)\|_F^2 + \lambda_i \left\|vec\left(F\right)_I\right\|_1 + \lambda_{enet}\|F_I\|_F^2 \tag{29}$$

Sub problem 1 can be simplified using MM approach discussed above into following steps

```
Initialization: No. of partitions for DnC - n
Split rating matrix into n subparts
for i = 1:n
    % Perform MF on each submatrix
        Initialize variables, $F_{U_0}, F_{I_0} = rand$
        Set maximum no. of iterations, T
        while $k < T$ or $obj(k) - obj(k-1) \le 1e-7$
            $D = F_{Uk}F_{Ik} + \frac{1}{\nu}M^T\left(y - M\left(F_{Uk}F_{Ik}\right)\right)$
            $F_{Uk+1} \leftarrow \min_{F_U}\left\|\begin{pmatrix} D \\ 0 \end{pmatrix} - F_U\begin{pmatrix} F_I \\ \sqrt{\lambda_u}I \end{pmatrix}\right\|_F^2$
            $W = F_{Uk+1}F_{Ik} + \frac{1}{\nu}M^T\left(y - M\left(F_{Uk+1}F_{Ik}\right)\right)$
            $F_{Ik+1} \leftarrow \min_{F_I}\left\|\begin{pmatrix} W \\ 0 \end{pmatrix} - \begin{pmatrix} F_{Uk+1} \\ \sqrt{\lambda_{enet}}I \end{pmatrix}F_I\right\|_F^2 + \lambda_i\left\|vec\left(F_I\right)\right\|_1$
        end while
end for
for i = 1:n
    C_estimate ← Project estimates on column space of C(n)
end for
Net_Estimate ← Mean (C_estimate)
```

Figure 2. Algorithm

$$\begin{aligned} &Step\ 1:\ \mathrm{D} = F_{Uk}F_{Ik} + \frac{1}{\nu}M^T\left(y - M\left(F_{Uk}F_{Ik}\right)\right); \nu \ge \max eig\left(M^TM\right) \\ &Step\ 2:\ \min_{F_U}\|D - F_U \times F_I\|_F^2 + \lambda_u\|F_U\|_F^2 \end{aligned} \tag{30}$$

Step 2 can be recast as a simple least squares optimization as

$$\min_{F_U}\left\|\begin{pmatrix} D \\ 0 \end{pmatrix} - F_U\begin{pmatrix} F_I \\ \sqrt{\lambda_u}I \end{pmatrix}\right\|_F^2 \tag{31}$$

Which can be efficiently solved using any least square solver like gradient descent.

Similarly, sub problem 2 can also benefit from MM approach and written as following iterates

$$\begin{aligned} &Step\ 1:\ \mathrm{W} = F_{Uk}F_{Ik} + \frac{1}{\nu}M^T\left(y - M\left(F_{Uk}F_{Ik}\right)\right); \nu \ge \max eig\left(M^TM\right) \\ &Step\ 2:\ \min_{F_U}\|W - F_U \times F_I\|_F^2 + \lambda_i\left\|vec\left(F\right)_I\right\|_1 + \lambda_{enet}\|F_I\|_F^2 \end{aligned} \tag{32}$$

Step 2 can put reformulated as in (31)

$$\min_{F_I}\left\|\begin{pmatrix} W \\ 0 \end{pmatrix} - \begin{pmatrix} F_U \\ \sqrt{\lambda_{enet}}I \end{pmatrix}F_I\right\|_F^2 + \lambda_i\left\|vec\left(F_I\right)\right\|_1 \tag{33}$$

Equation (33) can be solved using iterative soft thresholding [7].

Both the sub problems are iteratively solved till convergence criteria is satisfied, i.e. maximum number of iterations reached or objective function variation between consecutive iterations falls below the threshold (1e-7).

The complete algorithm for implementation of our formulation on big datasets is given in fig. 2.

4. EXPERIMENTAL SETUP AND RESULTS

This section describes our experimental setup for testing our novel proposition. Also, comparison with various standard matrix factorization algorithms is given in terms of mean absolute error (MAE) and execution times.

4.1 Experimental Setup

We conducted experiments on Movielens 10M dataset [48] which has been used extensively for benchmarking collaborative filtering algorithms. The rating matrix has a dimension of 71567 users and 10667 items with 10 million ratings valued 1-5. The size of rating matrix is justifies the use of divide and combine approach. We performed fivefold cross validation, the available ratings are split into five parts. Four of the parts (80% of available ratings) form the training data and the last set (20% of available data) constitutes the test set. For each of the test-train pair, 50 independent runs of the algorithms were carried out.

Baseline estimation was done offline, using (21) and the interaction part fed into our model framework. The value of regularization parameter for the same was kept at 0.001 and 250 iterations were carried out using stochastic gradient algorithm.

For our experimentation, the complete interaction (user-item rating) matrix was split into four disjoint sub matrices of almost equal dimensions – by splitting along the column. Our matrix factorization algorithm (EBCS-BD) was applied to each of the sub matrices in parallel. The value of regularization parameters were selected using L-curve method [13]. The optimum values were found to be $\lambda_i = 1e-3$, $\lambda_{enet} = 1e-2$, $\lambda_u = 1e5$. The dimensionality of the model – number of latent factors – were experimentally selected to be 50. The resulting predicted estimates were combined as per the design procedure underlined in fig. 2. After the interaction component of the ratings are predicted, baseline data computed offline is added back to recover completely filled rating matrix.

4.2 Results

Our model and design algorithm was compared against the traditional and state of the art methods for matrix factorization, namely Accelerated Proximal Gradient (APG) [41], Singular Value Thresholding (SVT) [4] and optSpace [17].

Table 1 shows the comparison between all techniques on the basis of recovery accuracy measured in terms of MAE (34).

$$MAE = \frac{\sum_{m,n} \Re_{m,n} - \hat{\Re}_{m,n}}{|\Re|} \tag{34}$$

Where, $\Re_{m,n}$ and $\hat{\Re}_{m,n}$ are the actual and predicted ratings and $|\Re|$ is the cardinality of the rating matrix $\Re$. It's the standard measure for benchmarking algorithm for recommender system design.

It can be observed that our algorithm gives better recovery accuracy than the other standard algorithms compared against. Our algorithm performs around 4% better than optSpace and around 7.5% improvement is shown with respect to SVT in terms of mean absolute error. APG algorithm shows erratic behavior and does not give 100% coverage, i.e. not all ratings can be predicted in all the cases. The values shown in table 1 are the best case values. On the other hand, our algorithm ensures 100% coverage and consistently perform well for various test sets and multiple runs. Even for best case results, our algorithm is able to achieve a decrease of 2% in MAE values over APG. By RS standards, this improvement is substantially relevant.

Table 1. Mean Absolute Error for Various Algorithms

Algorithm	Mean Absolute Error
EBCS-BD (proposed)	0.6185
APG	0.6307
OptSpace	0.6437
SVT	0.6645

Mean absolute error is a measure of overall accuracy of the algorithm. However, for each user what's important is how close the predictions to his /her actual choice are. Hence, in table 2 we show the spread of prediction error of various algorithms. Prediction error, *PE=n* indicates that the error between the actual and predicted ratings is *n*. The values shown in the table are the percentage of ratings having the stated prediction error. On this measure also, our algorithm performs the best, with most of ratings having an error of less than 2.

Table 2. Spread of Prediction error

Algorithm	PE=0	PE=1	PE=2	PE=3	PE=4
EBCS-BD	38.47	54.71	6.60	0.18	0.02
APG	37.67	52.34	8.82	1.04	0.12
OptSpace	36.67	53.10	7.99	1.99	0.14
SVT	35.40	53.46	8.71	1.27	0.16

It's important for a RS design algorithm to not just be accurate but be sufficiently fast. A faster algorithm ensures that the model can be updates more frequently and also online computation of rating can also be carried out more efficiently.

Table 3. Run Times for Various Algorithms

Algorithm	Run Times (seconds)
EBCS-BD (proposed)	170.61
APG	276.05
OptSpace	1159.89
SVT	265.74

The run times for various algorithms is shown in table 3. The times are for all three phases combined i.e., split, perform matrix factorization and combine to yield final estimate. It's evident that our algorithm is considerably faster than other algorithms. Our algorithm (because of use of MM approach) is almost 1.5 times faster than APG and SVT, nearest to it in terms of time requirement. This aids in design of an efficient recommender system.

5. CONCLUSION

In this work, we propose a novel recommender system design approach based on the latent factor model implemented on a distributed platform. Existing latent factor based models aim to recover the rating matrix as a product of a dense item and a dense user latent factor matrix. We also propose to recover the rating matrix as a product of user and item factor matrices, but do not impose the dense structure constraint. We claim that the user latent factor matrix is dense but for the item latent factor matrix the same doesn't hold true. This is because every user will demonstrate certain degree of affinity towards all traits but, no item can concurrently possess all the traits. Hence, we promote recovery of a dense user and a sparse item latent factor matrix.

Along with this, we also argue that the various traits defining the items are not independent. This correlation and interdependence between the items is captured by use of an elastic-net regularization based penalty term. This addition promotes a group sparsity effect in the sparse item factor vector - correlated factors are selected together. We show that our proposed formulation naturally fits into the blind compressive sensing framework with an add-on elastic net penalty term.

We also derive and efficient algorithm for solving our problem formulation using Majorization minimization approach. Use of MM technique helps in breaking complex and computationally intensive optimization problem into simple iterative procedure. Thus, use of MM method greatly reduces the computational burden and run times.

Also, we employ divide and combine approach to employ our formulation efficiently on a distributed platform to very large rating matrices.

In this work we have experimented on the movielens dataset. It is shown that our algorithm outperforms other collaborative filtering techniques compared against. Our algorithm is able to achieve improved quality of prediction and a reduction in mean absolute error. Also, our algorithm using MM approach ensures that the run times for our design is much smaller than for other algorithms. Hence, our design incorporates two basic requirements of recommender systems – high accuracy and smaller execution.

6. REFERENCES

[1] Adomavicius, G., and Tuzhilin, A. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. IEEE Transactions on Knowledge and Data Engineering, 17(6), 2005, 734-749.

[2] Baraniuk, R. G. Compressive sensing. IEEE signal processing magazine, 24(4), 2007

[3] Bell, R. M., and Koren, Y. Scalable collaborative filtering with jointly derived neighborhood interpolation weights. In Seventh IEEE International Conference Data Mining, 2007. ICDM 2007, pp. 43-52

[4] Cai, J. F., Candès, E. J., and Shen, Z. A singular value thresholding algorithm for matrix completion. SIAM Journal on Optimization, 20(4), 2010, 1956-1982.

[5] Candès, E. J.,and Recht, B. Exact matrix completion via convex optimization. Foundations of Computational mathematics, 9(6), 2009, 717-772.

[6] Candes, E. J. The restricted isometry property and its implications for compressed sensing. Comptes Rendus Mathematique, 346(9), 2008, 589-592.

[7] Daubechies, I., Defrise, M., and De Mol, C. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. Communications on pure and applied mathematics, 57(11), 2004, 1413-1457.

[8] Donoho, D. L. Compressed sensing. IEEE Transactions on Information Theory, 52(4), 2006, 1289-1306.

[9] Figueiredo, M. A., Bioucas-Dias, J. M., and Nowak, R. D. Majorization–minimization algorithms for wavelet-based image restoration. IEEE Transactions on Image Processing, 16(12), 2007, 2980-2991.

[10] Frieze, A., Kannan, R., and Vempala, S. Fast Monte-Carlo algorithms for finding low-rank approximations. Journal of the ACM (JACM), 51(6), 2004, 1025-1041.

[11] Gleichman, S., & Eldar, Y. C. Blind compressed sensing. IEEE Transactions on Information Theory, 57(10), 2011, 6958-6975.

[12] Gross, D. Recovering low-rank matrices from few coefficients in any basis. Information Theory, IEEE Transactions on, 57(3), 2011, 1548-1566.

[13] Hansen, P. C., and O'Leary, D. P. The use of the L-curve in the regularization of discrete ill-posed problems. SIAM Journal on Scientific Computing, 14(6), 1983, 1487-1503.

[14] Herlocker, J.L., Konstan, J.A., Terveen, L.G., and Riedl, J.T. Evaluating collaborative filtering recommender systems. ACM Transactions on Information Systems (TOIS), 22(1), 2004, 5-53

[15] Hoerl, A. E., and Kennard, R. W. Ridge regression: Biased estimation for nonorthogonal problems. Technometrics, 12(1), 1970, 55-67.

[16] Hofmann, T. Latent semantic models for collaborative filtering. ACM Transactions on Information Systems (TOIS), 22(1), 2004, 89-115.

[17] Keshavan, R. H., and Oh, S. A gradient descent algorithm on the grassman manifold for matrix completion. arXiv preprint arXiv:0910.5260.

[18] Koren, Y., and Bell, R. Advances in collaborative filtering. In Recommender Systems Handbook, Springer US, 145-186

[19] Koren, Y., Bell, R., and Volinsky, C. Matrix factorization techniques for recommender systems. Computer, 42(8), 2009, 30-37.

[20] Lee, D. D., and Seung, H. S. Algorithms for non-negative matrix factorization. In Advances in neural information processing systems, 2001, 556-562

[21] Lee, J., Recht, B., Srebro, N., Tropp, J., & Salakhutdinov, R. (2010). Practical large-scale optimization for max-norm regularization. In Advances in Neural Information Processing Systems , 1297-1305, 2010J.D

[22] Linden, G., Smith, B., and York, J. Amazon. com recommendations: Item-to-item collaborative filtering. IEEE Internet Computing, 7(1), 2003, 76-80.

[23] Lingala, S. G., and Jacob, M. Blind compressive sensing dynamic MRI.Medical Imaging, IEEE Transactions on, 32(6), 2013, 1132-1145.

[24] Liu, C., Yang, H. C., Fan, J., He, L. W., and Wang, Y. M. Distributed nonnegative matrix factorization for web-scale dyadic data analysis on mapreduce. In Proceedings of the 19th international conference on World wide web , April 2010, 681-690

[25] Ma, S., Goldfarb, D., & Chen, L. Fixed point and Bregman iterative methods for matrix rank minimization. Mathematical Programming, 128(1-2), 2011, 321-353.

[26] Mackey, L. W., Jordan, M. I., and Talwalkar, A. Divide-and-conquer matrix factorization. In Advances in Neural Information Processing Systems, 2011, 1134-1142

[27] Majumdar, A., and Ward, R. K. Classification via group sparsity promoting regularization. In IEEE International Conference on Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. 861-864

[28] Melville, P., Mooney, R.J., and Nagarajan, R. Content-boosted collaborative filtering for improved recommendations. In AAAI/IAAI, July 2002, 187-192

[29] Mnih, A., and Salakhutdinov, R. Probabilistic matrix factorization. InAdvances in neural information processing systems, 2007, 1257-1264

[30] Mooney, R. J., Bennett, P. N., and Roy, L. Book recommending using text categorization with extracted information. In Proc. Recommender Systems Papers from 1998 Workshop, Technical Report WS-98-08, 1998

[31] Natarajan, B. K. Sparse approximate solutions to linear systems. SIAM journal on computing, 24(2), 1995, 227-234.

[32] Pazzani, M.J., and Billsus, D. Content-based recommendation systems. In The adaptive web. Springer Berlin Heidelberg, 2007, 325-341

[33] Recht, B., Fazel, M., and Parrilo, P. A. Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. SIAM review,52(3), 2010, 471-501.

[34] Rennie, J. D., & Srebro, N. Fast maximum margin matrix factorization for collaborative prediction. In Proceedings of the 22nd international conference on Machine learning, August 2005, 713-719

[35] Sarwar, B., Karypis, G., Konstan, J., and Riedl, J. Item-based collaborative filtering recommendation algorithms. In Proceedings of the 10th international conference on World Wide Web, April 2001, 285-295

[36] Schafer, J. B., Konstan, J., and Riedl, J. Recommender systems in e-commerce. In Proceedings of the 1st ACM conference on Electronic commerce, New York, NY, USA, 1999, 158-169

[37] Salakhutdinov, R., and Mnih, A. Bayesian probabilistic matrix factorization using Markov chain Monte Carlo. In Proceedings of the 25th international conference on Machine learning, July 2008, 880-887

[38] Shamir, O., and Shalev-Shwartz, S. Collaborative filtering with the trace norm: Learning, bounding, and transducing, COLT, JMLR Proceedings, 19, 2011, 661-678

[39] Srebro, N., Rennie, J., and Jaakkola, T. S. Maximum-margin matrix factorization. In Advances in neural information processing systems, 2004, 1329-1336

[40] Tibshirani, R. Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society. Series B (Methodological), 1996, 267-288.

[41] Toh, K. C., and Yun, S. An accelerated proximal gradient algorithm for nuclear norm regularized linear least squares problems. Pacific Journal of Optimization, 6(15), 2010, 615-640)

[42] Yu, K., Schwaighofer, A., Tresp, V., Xu, X., and Kriegel, H. P. Probabilistic memory-based collaborative filtering. IEEE Transactions on Knowledge and Data Engineering, 16(1), 2004, 56-69.

[43] Xiong, L., Chen, X., Huang, T. K., Schneider, J. G., and Carbonell, J. G. Temporal Collaborative Filtering with Bayesian Probabilistic Tensor Factorization. In *SDM* , 10, 2010, 211-222

[44] Xue, G. R., Lin, C., Yang, Q., Xi, W., Zeng, H. J., Yu, Y., and Chen, Z. Scalable collaborative filtering using cluster-based smoothing. In Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval , August 2005, 114-121

[45] Zou, H., and Hastie, T. Regularization and variable selection via the elastic net. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 67(2), 2005, 301-320.

[46] Zhou, Y., Wilkinson, D., Schreiber, R., and Pan, R. Large-scale parallel collaborative filtering for the netflix prize. In Algorithmic Aspects in Information and Management, Springer Berlin Heidelberg, 2008, 337-348

[47] http://sifter.org/~simon/journal/20061211.html

[48] http://grouplens.org/datasets/movielens/

Subgraph Rank: PageRank for Subgraph-Centric Distributed Graph Processing

Nitin Chandra Badam
Dept. of Computer Science and Engineering,
IIT Guwahati
chandra.nitin@iitg.ernet.in

Yogesh Simmhan
Supercomputer Education and Research Centre,
Indian Institute of Science, Bangalore
simmhan@serc.iisc.in

ABSTRACT

The growth of Big Data has seen the increasing prevalence of interconnected graph datasets that reflect the variety and complexity of emerging data sources. Recent distributed graph processing platforms offer vertex-centric and subgraph-centric abstractions to compose and execute graph analytics on commodity clusters and Clouds. Naïve translation of existing graph algorithms to these programming models can offer sub-optimal performance. We analyze the effectiveness of PageRank, a popular graph centrality measure, for a subgraph-centric programming model, and propose variations based on the existing BlockRank algorithm to improve the performance. We evaluate these algorithms on real-world graphs using the GoFFish platform on Amazon EC2 Cloud VMs, and demonstrate that the proposed Subgraph Rank algorithm outperforms the native PageRank and BlockRank algorithms, and is faster by $23 - 74\%$ for most graphs we evaluated, while achieving an equivalent PageRank quality.

1. INTRODUCTION

Data processing has seen a sea change in the recent decade. The term "Big Data" has been coined to reflect the potential of – and complexity of – managing, exploring and analyzing this massive influx, in order to offer knowledge and insights. Google's MapReduce [9] has proven seminal not just in providing a framework for processing large data volumes, but in allowing us to easily leverage distributed commodity resources and Clouds to achieve the same. As such, the characteristics of these large scale datasets have been evolving since the era of Google's massive web logs, whose text and tuple based processing motived MapReduce. We are now seeing the pervasiveness of *interconnected data* – linked data from the web [5], billions of social network users [36], online sensing networks from Internet of Things [27], genome networks [3], to name a few – that reflect the variety and complexity of emerging Big Data sources.

This need for large scale graph processing has motivated the development of *distributed graph platforms* such as Pregel [28], GraphLab [26], and Giraph++ [35]. Some of these also operate on special graph structures, such as PowerGraph [14] for power law graphs and GoFFish [34] for time-series graphs. Such graph platforms complement the extensive work on parallel graph processing [2] on high performance computing hardware by instead using commodity clusters and Clouds that are more broadly accessible. At the same time, these distributed platforms also, arguably, offer simpler programming abstractions than, say, MPI to compose graph applications and analytics. Vertex- [1, 28] and subgraph- [34, 35] centric abstractions, for example, allow users to compose the graph application from the perspective of a single vertex or subgraph and operate across iterative supersteps, much like MapReduce allows users to write their logic over individual key-value(s) pairs. The platforms handle scalable, data parallel execution of the graph applications once mapped to their programming model. Recent results have shown the *subgraph-centric programming models to out-perform the vertex-centric abstractions* [34, 35], and thus hold promise for wider adoption.

At the same time, the introduction of these novel graph programming abstractions means that existing shared memory or parallel graph algorithms may not be a direct fit on these platforms. And a naïve translation of existing algorithms to these new abstractions may offer sub-optimal performance. It is well known that algorithmic innovations at design-time, that effectively use the underlying abstractions, can significantly improve the application performance compared to relying exclusively on runtime optimizations provided by the platform. As a result, there is a need to examine where existing algorithms fit directly, need to be adapted or new algorithms are required, to make the best use of such platforms.

Graph centrality measures are a key analytic that is used in real-world networks, from understanding critical junctions in power grids [10] to the spread of ideas (or diseases) in social (or human) networks. *PageRank* [29] proposed by Google for web graphs is a special case of Eigenvalue Centrality [7], and is often used as a canonical algorithm for evaluating graph platforms. As graph structures and sizes have evolved over the past decade, research into PageRank has contributed more scalable algorithms that handle heterogeneous and distributed topologies. This paper continues in that spirit.

There has been extensive work on improving the PageRank algorithm to fit different platforms, including MapReduce [4, 18, 20]. As a result, it is useful to understand how

The 20th International Conference on Management of Data (COMAD),
17th-19th Dec 2014 at Hyderabad, India.

effectively the PageRank for a graph can be computed using such novel distributed graph processing frameworks – where existing PageRank algorithms work, and when new ones need to be developed.

To this end, we analyze how the PageRank algorithm and its parallel variants map to a subgraph-centric programming model, and their performance for real-world graphs, including non-web graphs. In particular, we use the *BlockRank* algorithm [22], that naturally appears to suit a subgraph-centric model, as an algorithmic starting point and propose variations to better leverage the subgraph-centric abstraction. In the process, we make the following specific contributions in this paper:

1. We map the BlockRank algorithm to a subgraph-centric programming abstraction, and analyze its deficiencies,
2. We propose variations of the BlockRank algorithm including *Subgraph Rank*, and hypothesize their behavior in a subgraph-centric model, and
3. We implement the PageRank, BlockRank variations and Subgraph Rank algorithms using the GoFFish sub -graph-centric distributed graph platform, and experimentally evaluate their quality and performance across diverse real-world graphs on Amazon EC2 Cloud Virtual Machines (VMs).

The rest of the paper is organized as follows: in § 2, we offer a background of PageRank, BlockRank and subgraph-centric graph programming abstractions; in § 3, we introduce variations to the BlockRank algorithm that can improve its performance, and also propose the Subgraph Rank algorithm as a suitable candidate for subgraph-centric programming abstractions; in § 4, we evaluate our proposed algorithms on three real-world graphs using the GoFFish platform running on Amazon's public Cloud; we discuss related work in § 5, and summarize our contributions and provide directions for future work in § 6.

2. BACKGROUND

We provide background material on PageRank and BlockRank algorithms from prior literature, and subgraph-centric programming abstractions that is our target distributed platform.

2.1 Subgraph-Centric Abstractions

Vertex-centric graph programming abstractions have been proposed by distributed platforms like Pregel [28] and GraphLab [26]. In Pregel, vertices of a graph are partitioned across multiple machines, and the computation is performed from the view of a single vertex in a pleasingly parallel manner. Coordination between distributed vertices takes place through synchronized message passing at superstep boundaries. The graph applications progress iteratively, one superstep at a time, interleaving computing and communication. Bulk Synchronous Parallel (BSP) [12] processing allows efficient bulk transfer of messages on slower, commodity networks, while also avoiding potential race conditions and circular dependencies of distributed task execution.

In a *subgraph-centric programming abstraction*, the computation is performed at the coarser granularity of a subgraph, with synchronized messages passed by subgraphs to its neighboring subgraphs at superstep boundaries. While platforms like Giraph++ [35] treat each graph partition as a (disconnected) subgraph, others like our own GoFFish platform [34] identify weakly connected components within each partition as units of subgraph execution. For consistency, we assume the latter definition of subgraphs as weakly connected components though our results translate to both definitions. Subgraph centric programming has been shown to be faster than vertex centric programming due to better use of shared memory algorithms on entire subgraphs, reduced message passing overheads, and fewer supersteps to convergence for graph applications.

Algorithm 1 Subgraph-Centric PageRank (SGPR)

```
procedure COMPUTE(Subgraph $SG$,Message msg[])
    if $superStep$ > MAX then                          ▷ Sanity check
        VoteToHalt()
    end if
    if superStep == 1 then
        for $v$ in $SG.vertices$ do
            pr[$v$] = $\frac{1}{|G.vertices|}$                   ▷ Initialize PR
        end for
        SendMessageToNeighbors(pr[])
    else
        sums[] = ComputePRSums(pr)
                    ▷ also includes local contributions
        for $v$ in $SG.vertices$ do                     ▷ Update PR
            pr[$v$] = 0.85×sums[$v$]+0.15 × $\frac{1}{|G.vertices|}$
        end for
        if superStep == 30 then
            VoteToHalt()                        ▷ Halt after 30 iters
        else
            SendMessageToNeighbors(pr[])
        end if
    end if
end procedure
```

GoFFish, used to evaluate our algorithms in this paper, partitions graphs across multiple hosts or VMs, identifies subgraphs within each partition, and stores the subgraph and its attributes within its GoFS distributed storage. It is optimized for a write-once, read-many batch processing model. Subgraphs have local edges between their local vertices, and remote edges connecting to subgraphs in other partitions. Graphs are partitioned using METIS [23] to minimize the edge-cuts between partitions and balance the number of vertices per partition. Subgraph centric applications composed in GoFFish are executed on distributed hosts using the Floe Dataflow engine which implements the BSP execution model.

2.2 PageRank and BlockRank

PageRank is a centrality measure that indicates the relative importance of a vertex within a graph [29]. The PageRank of a webpage (vertex) in a web graph, with hyperlinks forming edges, is the probability that a web surfer who is performing a random walk on the web graph will end up at that page when following the links [7]. For a given webpage $v \in \mathbb{V}$ in a web graph $G = (\mathbb{V}, \mathbb{E})$ with the set of vertices $\mathbb{V}$ (webpages) and edges $\mathbb{E}$ (hyperlinks), its PageRank $\mathcal{P}(v)$ is given by the following iterative logic:

$$\mathcal{P}_0(v) = \frac{1}{n} \tag{1}$$

$$\mathcal{P}_{i+1}(v) = \alpha \times \left(\sum\nolimits_{\omega \in \mathcal{I}(v)} \frac{\mathcal{P}_i(\omega)}{|\mathcal{O}(\omega)|} \right) + (1 - \alpha) \times \frac{1}{n} \tag{2}$$

$$\mathcal{P}(v) = \mathcal{P}_{i+1}(v) \mid \forall v \in \mathbb{V}, |\mathcal{P}_{i+1}(v) - \mathcal{P}_i(v)| < \epsilon \quad (3)$$

where $n = |\mathbb{V}|$ is the number of vertices in the web graph, $|\mathcal{O}(v)|$ is the out-degree of the vertex v and $\mathcal{I}(v)$ is the set of neighboring vertices that have incoming edges into v. α gives the probability with which the random walk will follow an outgoing link from a vertex, with $(1 - \alpha)$ being the probability of taking a jump elsewhere. ϵ is a distance threshold beyond which successive iterations of PageRank should not change by, for the algorithm to terminate. Eqn. 1 initializes the PageRank to the probability of starting at any random vertex in the graph, Eqn. 2 is the iterative step which updates a vertex's PageRank values based on the weighted values reported by its neighbors, while Eqn. 3 is the halting condition where the PageRank has quiesced for all vertices across the graph.

PageRank has been a *de facto* graph algorithm for validating graph platforms, and the two recent subgraph-centric platforms, Giraph++ and GoFFish, have both mapped PageRank using a subgraph-centric programming model. Alg. 1 lists the pseudo-code for a subgraph-centric PageRank. After initialization the PageRank for all vertices in the subgraph in the first superstep, each subgraph sends the PageRank values for its vertices to their neighboring vertices present in remote subgraphs. In each subsequent superstep, the subgraph uses the PageRank values for neighboring vertices available from the received messages to update the PageRank value for its vertices, taking a damping factor into consideration. The algorithm either runs till a threshold value of convergence is reached, or for a fixed number of supersteps (30 shown in Alg. 1). As can be seen, the subgraph-centric algorithm still iterates through every vertex and applies the update in each superstep, mimicking the behavior of a vertex-centric algorithm. This makes this algorithm a naïve mapping to a subgraph-centric model.

Though PageRank was defined for web graphs and for link analysis, the measure is used in many other domains [13] like in Social Network, Citation Networks, and Road Network analysis. Hence, scaling PageRank on distributed platforms for diverse graphs has the potential to benefit multiple domains.

Different flavors of PageRank algorithms have been proposed, both to improve the quality of the ranks and to speed up convergence. These include Topic-sensitive PageRanks [18], Personalized PageRanks [20], and BlockRank [22]. The *BlockRank* algorithm is based on the idea that a general web graph has an inherent block structure, i.e., sets of webpages with a high concentration of interconnected hyperlinks, such as found between pages within a web domain. The intra-block edges tend toward a clique-like structure within a block, while the inter-block edges are sparse.

The generic BlockRank algorithm operates over three phases. In the *Local PageRank* phase, localized PageRank values are calculated for each vertex in a block by omitting all the remote links between the blocks. Next, in the *BlockRank* phase, a BlockRank value, which represents the relative importance of each block in the graph, is computed. For this, we consider each block as a meta-vertex in a meta-graph, and edges between blocks as meta-edges. A variation of the PageRank algorithm is performed on this meta-graph, and the PageRank for each block (meta-vertex) is its BlockRank. In the last *Global PageRank* phase, we estimate the initial PageRank values for each vertex by combining their Local PageRank with the BlockRank, and then perform the standard PageRank algorithm using these initial values.The authors [22] show that the BlockRank algorithm converges faster than traditional PageRank, and they offer theoretical performance bounds for block-structured graphs.

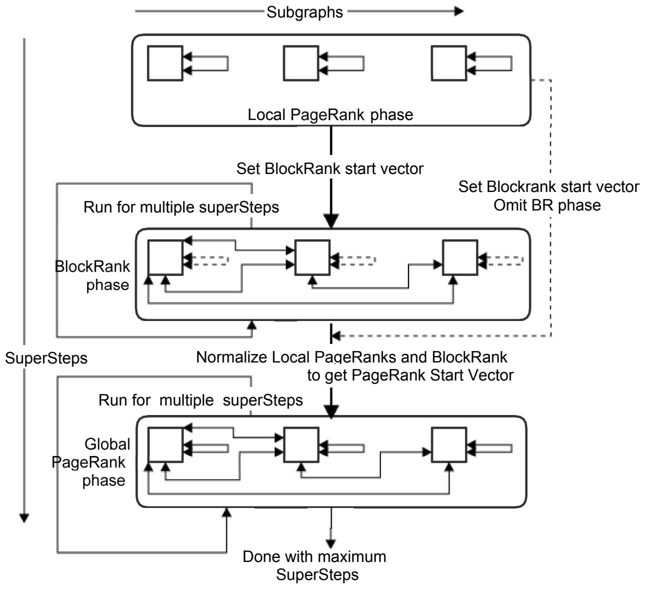

Subgraphs
Local PageRank phase
Set BlockRank start vector
Set Blockrank start vector
Omit BR phase
Run for multiple superSteps
BlockRank
phase
SuperSteps
Normalize Local PageRanks and BlockRank
to get PageRank Start Vector
Run for multiple superSteps
Global
PageRank
phase
Done with maximum
SuperSteps

Figure 1: Flowchart of BlockRank algorithm phases. Boxes listed horizontally show different subgraphs, and boxes listed vertically span supersteps.

3. BLOCKRANK & SUBGRAPH RANK

In this section, we discuss BlockRank and its variations for a subgraph-centric model, and introduce the Subgraph Rank algorithm.

3.1 Native BlockRank Algorithm (BRNA)

One of the challenges of running a naïve PageRank algorithm using a subgraph-centric model (SGPR) is that its execution behavior mimics that of a vertex-centric version [34]. In every superstep, the PageRank value of a vertex is updated and passed as messages to its neighbors. There by, the subgraph as a whole is unable to make progress without each localized vertex making identical progress. Exposing subgraph-level computation would mitigate this downside, and effectively leverage the abstraction.

The BlockRank algorithm leverages the block structure of the web to speed up the convergence of PageRank. A graph is said to have a block-structure if, when considering it as an adjacency matrix, there are blocks within the matrix that have high intra-vertex edge connectivity and the inter-block edge connectivity is sparse. This algorithm is theoretically proven to perform better than PageRank for block-structured graphs. Since blocks loosely correspond to the notion of subgraphs, it is worthwhile mapping BlockRank to a subgraph-centric model to potentially do better than naïve PageRank.

Fig. 1 shows the phases involved in performing BlockRank using a subgraph-centric model. The corresponding pseudocode is listed in Alg. 2. There are three key phases in BlockRank, as mentioned before: (1) Local PageRank (LPR) computation, (2) BlockRank computation, and (3) Global

PageRank (GPR) computation. Each of these phases are performed using a subgraph-centric abstraction, with computation distributed across subgraphs (horizontal boxes in Fig. 1) and proceeding iteratively over one or more supersteps (vertical boxes).

The LPR phase runs in the first superstep and here, each subgraph operates independently as its own graph, ignoring remote edges to other subgraphs, and calculates the PageRank for each of its vertices (pr[v]) using the standard iterative PageRank algorithm, in-memory. This is shown in Alg. 2, lines 6–17. It then proceeds to the BlockRank phase where each subgraph (block) is treated as a vertex in a meta-graph, and the BlockRank for each subgraph is calculated, starting with an initial value that equals $\frac{1}{|Subgraphs|}$ and iterating over supersteps to exchange BlockRank values between subgraphs after each superstep (Alg. 2, lines 20–32). This phase runs for a fixed number of supersteps – we set this to 10 supersteps since that achieves a reasonable BlockRank convergence for meta-graphs with up to 2700 subgraphs (meta-vertices) that we observe. The BlockRank values (br) calculated at the end of this phase are used as weighing factors to initialize the PageRank for each vertex of the graph (Alg. 2, line 30), referred to as "distributing" the BlockRank. Following this BlockRank distribution, the GPR phase starts with these initialized PageRank values and runs over multiple supersteps until convergence.

While we can map the BlockRank to a subgraph-centric model, we perceive shortcomings to this algorithm (substantiated in the evaluation, § 4) in practice:

- General graphs may not have a strong block-structure to them, and as such the BlockRank algorithm may not perform well for such graphs.
- The block structure only loosely maps to a subgraph. Since the block sizes of graphs can vary widely, partitioning these graphs can end up splitting blocks across multiple subgraphs, thereby impacting the benefits of block-level computations.

Hence, we propose a more robust solutions to calculating PageRank that goes beyond the native BlockRank algorithm.

3.2 Dimensions for Varying BlockRank

We can vary the native BlockRank algorithm along several dimensions to explore a variant that is more suitable to subgraph-centric models and for general graphs. Some of these change dimensions are discussed first, followed by the variations of BlockRank algorithms that include one or more of these. In this section, consider the graph $G = (\mathbb{V}, \mathbb{E})$ with the set of vertices $\mathbb{V}$ and edges $\mathbb{E}$ is partitioned into a set of subgraphs (blocks) $S_i = (V_i, E_i) \in \mathbb{S}$, where $V_i \subset \mathbb{V}$ and $E_i \subset \mathbb{E}$.

3.2.1 BlockRank Initialization Vector

SGC-inverse: The BlockRank phase has to start with an initial BlockRank value for each subgraph (block) in the first iteration. This choice is significant and affects the BlockRank phase. A natural choice is a uniform distribution, similar to PageRank, across all the blocks given by $\frac{1}{|\mathbb{S}|}$, where $|\mathbb{S}|$ is the count of subgraphs in the graph (hence, subgraph count or SGC). So every subgraph gets the same initial BlockRank value irrespective of its structure.

Algorithm 2 Subgraph-centric Native BlockRank (BRNA)

```
 1: procedure Compute(Subgraph $SG$,Message msg[])
 2:     if $superStep$ > MAX then                          ▷ Sanity check
 3:         VoteToHalt()
 4:     end if
 5:     if $superStep == 1$ then                       ▷ Local PageRank
 6:         for $v$ in $SG.vertices$ do
 7:             pr[$v$] = $\frac{1}{|SG.vertices|}$                     ▷ Initialize PR
 8:         end for
 9:         do
10:             sums[] = ComputePRSums(pr)
11:             $L1Norm = 0$
12:             for v in $SG.vertices$ do                     ▷ Update PR
13:                 $prev$ = pr[$v$]
14:                 pr[$v$] = 0.85×sums[$v$]+0.15× $\frac{1}{|SG.vertices|}$
15:                 $L1norm = L1norm+$ Abs($prev$ - pr[$v$])
16:             end for
17:         while $L1Norm > \epsilon$                      ▷ Test convergence
18:         $isBRActive$ = TRUE
19:     else if $isBRActive$ then                          ▷ BlockRank
20:         if IsFirstBRSuperStep() then
21:             $br = \frac{1}{|Subgraphs|}$                         ▷ Initialize BR
22:         else            ▷ Update BR
23:             $sum$ = ComputeBRSum(msg[])
24:             $br = 0.85 \times sum + 0.15 \times \frac{1}{|Subgraphs|}$
25:         end if
26:         SendMessageNeighbors($br$)
27:         if IsMaxBRSuperSteps() then
28:             isBRActive = FALSE
29:             for v in $SG.vertices$ do
30:                 pr[$v$] = $br$× pr[$v$]                    ▷ Distribute BR
31:             end for
32:         end if
33:     else        ▷ Global PageRank
34:         DoPageRank($SG$, pr[], msg[])
35:                 ▷ VoteToHalt() on convergence
36:     end if
37: end procedure
```

SG-by-G: Here the initial value of BlockRank is weighed by the number of vertices contributed by the subgraph to the entire graph, and given as $\frac{|V_i|}{|\mathbb{V}|}$ for the subgraph S_i, where $|V_i|$ is the number of vertices in S_i and $|\mathbb{V}|$ is the number of vertices in the entire graph. Intuitively, this is introduces fairness in the allocation that accounts for the varying sizes of different blocks.

3.2.2 BlockRank Distribution

Native: In the PageRank algorithm, a fraction $\frac{1}{n}$ of the PageRank values $\mathcal{P}(v)$ for a vertex v is passed at the end of each iteration to each of its n neighboring out vertices, uniformly. This fraction $\frac{1}{n}$ is called the transition probability distributed from vertex v to each its neighbors. Similarly, in the BlockRank phase for the BRNA algorithm, the BlockRank of each subgraph (block) is distributed to its neighboring subgraphs after each BlockRank iteration. This distribution logic is more complex as we consider both *self-edges* from a subgraph to itself, and the *local PageRank* calculated in the LPR phase for vertices in that subgraph in calculating the *block*-transition probability.

For a vertex $v \in \mathbb{V}$, its local PageRank is given by $\mathcal{LP}(v)$. The BlockRank at iteration 0 for a subgraph S_i is given by $\mathcal{BR}_0(S_i) = \frac{1}{|\mathbb{S}|}$. Now for every vertex $v \in V_i$ for the subgraph (block) S_i, we use $\mathcal{LP}(v)$ to calculate the block-transition probability, $\beta_k^{S_i \to S_j}$, from one subgraph S_i to S_j at the k^{th} iteration.

$$\forall S_i = (V_i, E_i) \in \mathbb{S}, \text{we have } \sum\nolimits_{v \in V_i} \mathcal{LP}(v) = 1 \tag{4}$$

$\forall S_i, S_j \in \mathbb{S}$, we have

$$\beta_k^{S_i \to S_j} = \sum_{v \in S_i, w \in S_j, \langle v,w \rangle \in E_i \& E_j} \frac{\mathcal{LP}(v)}{|\mathcal{O}(v)|} \tag{5}$$

$$\mathcal{BR}_{k+1}(S_j) = \alpha \times \Big(\sum\nolimits_{S_i \in \mathbb{S}} \beta_k^{S_i \to S_j} \times \mathcal{BR}_k(S_i) \Big) + (1-\alpha) \times \frac{1}{|\mathbb{S}|} \tag{6}$$

where $|\mathbb{S}|$ is the number of subgraphs (blocks) in the graph, $\langle v, w \rangle \in E_i \& E_j$ refers to edges incident from vertices in S_i to S_j, and $|\mathcal{O}(v)|$ and α follow same conversion as for PageRank. Eqn. 4 asserts that the sum of local PageRank values inside a subgraph is 1. Eqn. 5 calculates the block-transition probability from subgraph S_i to S_j if there is an edge from one to the other. It also considers self-edges ($S_i \to S_i$) and local PageRanks; a fraction $\frac{\mathcal{LP}(v)}{|\mathcal{O}(v)|}$ of the BlockRank will be passed to S_j. Eqn. 6 applies this block-transition probability to get the BlockRank of the subgraph for the next iteration.

So this gives consideration to self edges from the subgraph to itself ($\beta_k^{S_i \to S_i}$), and also includes the local PageRank so that important local vertices within the subgraph pass on a higher fraction of the BlockRank values to its neighbors.

PageRank-like: In this adaptation, self-edges within a subgraph and local PageRank values for vertices in the subgraph are ignored in the block-transition probability. Only edges that go from one subgraph to another are taken into consideration. So we construct a meta-graph with the subgraphs as the meta-vertices and edges between subgraphs as the meta-edges and run PageRank on the meta-Graph. In the equation below, $\mathcal{I}()$ and $\mathcal{O}()$ refer to the in-coming and out-going meta-vertices (subgraphs) connected to a subgraph.

$$\mathcal{BR}_{k+1}(S_j) = \alpha \times \left(\sum\nolimits_{S_i \in \mathcal{I}(S_j)} \frac{\mathcal{BR}_k(S_i)}{|\mathcal{O}(S_i)|} \right) + (1-\alpha) \times \frac{1}{|\mathbb{S}|} \tag{7}$$

3.3 Variations of BlockRank Algorithm

3.3.1 Native BlockRank (BRNA)

BRNA is a direct implementation of the BlockRank to a subgraph-centric model, as discussed in § 3.1 where each subgraph is treated as a block. Since while partitioning the graphs across distributed machines [23] we balance the number of vertices and minimizing the edge-cuts across partitions, a subgraph partially behaves like a block. This uses native BlockRank distribution logic and SGC-inverse initialization Vector.

3.3.2 BlockRank with PageRank-like Distribution Logic (BRDL)

Here the BRNA algorithm is used with a PageRank-like BlockRank distribution Logic instead of the native logic. Consequently, since the BlockRank phase effectively runs a PageRank algorithm on the meta-graph, it is more intuitive. BRDL completes the LPR phase in the first superstep and then proceeds to the BlockRank phase. We start with SGC-inverse initialization vector and follow PageRank-like BlockRank distribution for 10 supersteps, before switching to a global PageRank phase.

3.3.3 BlockRank with SG-by-G Initialization Vector (BRIV)

In BRIV, the BRNA algorithm changed to use SG-by-G initialization vector. Using SG-by-G ensures a fairer initial BlockRank value. One of the key rationales for using BlockRank is to compute a better start value for the global PageRank to allow rapid convergence. Vertices in smaller subgraphs have a higher local PageRank value compared to vertices in a larger subgraph since the sum of the local PageRank is 1 in both cases. Since the local PageRank is weighted with the BlockRank computed for that subgraph, using SG-by-G as the initial BlockRank allows vertices in larger subgraphs to regain their importance. As a result, we get a better estimate of the relative importance of the subgraphs, which is in the spirit of the original BlockRank paper. Note that SGC-inverse is a special case of SG-by-G where all subgraphs are of equal size.

3.3.4 BlockRank with PageRank-like Distribution Logic and SG-by-G Initialization Vector (BRDI)

Here, BRNA is modified to use SG-by-G as the initialization vector for the BlockRank Phase and uses PageRank-like BlockRank distribution logic. This couples the features of both BRDL and BRIV.

3.3.5 BlockRank without BlockRank Distribution (BRNO)

In this variation from BRNA, we skip running the BlockRank phase by setting the BlockRank value as $\frac{1}{|\mathbb{S}|}$ and passing it on to the initialization of the Global PageRank. We retain the initialization vector of SGC-inverse. As discussed before, it is observed that both large and small subgraphs end up converging to a uniform BlockRank value at the end of the BlockRank phase in the other variations. So this algorithm altogether avoids the BlockRank phase and sets the BlockRank directly to this uniform value, $br = \frac{1}{|\mathbb{S}|}$ (Alg. 2, line 30).

3.3.6 Subgraph Rank (SGRK)

The Subgraph Rank algorithm operates by changing the initialization vector for BlockRank in BRNO to SG-by-G, thereby intuitively captures the spirit of the original BlockRank algorithm for a general graph. The local PageRank phase is the same as for all BlockRank algorithms. In the BlockRank phase, we set the initial BlockRank for each subgraph as SG-by-G, and pass this value on directly to the global PageRank initialization without running any BlockRank supersteps. The pseudo-code for the Subgraph Rank algorithm is identical to the BRNA algorithm, Alg. 2, except that lines 20–26 to calculate BlockRank are replaced by the function: $br = \frac{|SG.vertices|}{|Graph.vertices|}$, i.e., the value of *br* is the ratio of the number of vertices in the subgraph to the total number of vertices in the whole graph.

4. EXPERIMENTAL EVALUATION

We present an empirical evaluation of the baseline subgraph-centric PageRank and native BlockRank algorithms, and demonstrate the relative performance of the proposed BlockRank variants, including Subgraph Rank, compared to these baselines. The algorithms are implemented on our GoFFish subgraph-centric distributed graph platform, and run on Amazon EC2 Cloud VMs. We expect these results to generalize to other subgraph-centric platforms such as Giraph++, when run on any commodity cluster or Infrastructure as a Service (IaaS) Cloud. We discuss the graph datasets used in the evaluation, the experimental setup and metrics for success, before we present the results.

4.1 Graph Datasets

We choose three real-world graphs available from Stanford's SNAP graph repository for our evaluation. These are summarized in Table 1. The graphs selected have diverse topological characteristics, and span different application domains, to ensure that our experimental results on quality, performance and scalability can be generalized. The Amazon Product Network (**AMZN**)[1]is a small graph from the eCommerce space, has a higher vertex degree (2.76) and medium diameter (44). The California Road Network (**CARN**)[2]is a medium sized graph from the transportation domain with a large diameter (849) and smaller degree (1.4). The Wikipedia Talk Network (**WIKI**)[3]is a larger social network community graph with over 5 *million* edges and a small diameter (9). These graphs have also been used in literature for evaluating other graph platforms [16, 34]. All these graphs are deployed as undirected.

Table 1: Graph datasets used and their properties

Graph	Vertices	Edges	Diameter	Vertex Degree
AMZN	334,863	925,872	44	2.76
CARN	1,965,206	2,766,607	849	1.40
WIKI	2,394,385	5,021,410	9	2.00

4.2 Execution Environment

All the existing and proposed algorithms are implemented and executed on the GoFFish graph analytics platform [4] [34]. Since the emphasis of this paper is on the improvements offered by the algorithm, implementing them all on GoFFish allows us to do a fair comparative evaluation. Alternatives such as Giraph++ [35] that offer a subgraph-centric model would also behave similarly. The subgraph-centric programming model is a natural superset of a vertex-centric programming model, and hence the naïve PageRank algorithm can also be implemented on platforms like Apache Giraph [34]. However, this does not extend to the other BlockRank algorithms that rely on the subgraph-centric model. The GoFFish platform and the algorithms evaluated are implemented in Java, and executed using JDK 1.7.

The experiments are run on Amazon Web Services (AWS) IaaS public cloud [5]. We use either `m3.large` or `m3.xlarge` Elastic Compute Cloud (EC2) Virtual Machines (VMs), as noted later. The specifications of the VM such as the number of virtual CPU cores and SSD-based local disk storage are given in Table 2. The guest OS on the VM is based on 64-bit Linux.

Table 2: AWS EC2 VM Specifications

VM Type	vCPU	Memory	Disk	Bandwidth[†]
`m3.large`	2	7.5 GiB	32 GB	moderate (∼700Mbps)
`m3.xlarge`	4	15 GiB	80 GB	high (∼1100Mbps)

[†] Indicative network bandwidth from http://blog.flux7.com/blogs/benchmarks/benchmarking-network-performance-of-m1-and-m3-instances-using-iperf-tool

All the VMs act as worker nodes for GoFFish, host the graph partitions and execute the subgraph-centric application. One these worker VM play an additional role of hosting the coordinator task, though it is a light weight process. GoFFish is multi-threaded, with each worker using twice as many threads as the number of CPU cores, and each thread working on one subgraph at a time.

4.3 Evaluation Metrics

We propose *quality* and *performance* measures to evaluate the success of the existing and proposed algorithms to calculate the PageRank of the three graphs. The quality measure calculates the proximity of the proposed algorithms' solutions to a *near-optimal PageRank* solution for the graph. We define this near-optimal PageRank for a graph as the PageRank value for its vertices arrived at the 100^{th} iteration (superstep) of running a naïve PageRank algorithm. We then calculate the L1 Norm at the k^{th} iteration of the PageRank values provided by the evaluated algorithms ($\mathcal{P}^{\star}_{k}$), against the PageRank values from this near-optimal solution ($\mathcal{P}_{100}$). This *Distance from Convergence* (DFC) for each vertex v in the graph at superstep k is given by:

$$\text{DFC}(k) = \sum\nolimits_{v \in \mathbb{V}} |\mathcal{P}^{\star}_{k}(v) - \mathcal{P}_{100}(v)| \tag{8}$$

This approximation helps avoid the oscillatory nature of the PageRank solution even as it incrementally narrows toward convergence as the supersteps increase. For e.g., Fig. 2 shows the incremental L1 Norm *between two successive supersteps* for the CARN graph using a naïve PageRank algorithm, and its oscillations past 55 supersteps. Due to this behavior, using the L1 Norm between supersteps, which estimates the incremental change in PageRank value (Eqn. 3), as a metric for convergence can lead to incorrect premature halting. This approach has been used elsewhere too [35].

For our evaluation, we consider representative threshold distance values for DFC: $0.1, 0.01, 0.001$, and 0.0001 ; these distances are in short denoted by $\Delta_{0.1}$, $\Delta_{0.01}$, $\Delta_{0.001}$ and $\Delta_{0.0001}$. Depending on the quality of the PageRank results

---

[1] http://snap.stanford.edu/data/com-Amazon.html

[2] http://snap.stanford.edu/data/roadNet-CA.html

[3] http://snap.stanford.edu/data/wiki-Talk.html

[4] http://github.com/usc-cloud/goffish

[5] http://aws.amazon.com/ec2

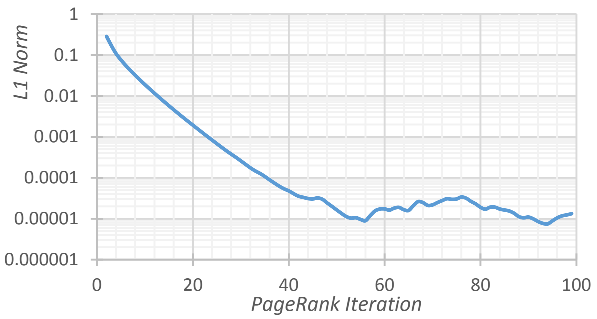


Figure 2: L1 Norm between PageRank values of successive supersteps for CARN using SGPR.

required for a graph, it suffices to iterate the algorithm till its PageRank reaches within one of these four Δ thresholds.

The performance measures we consider include the *number of supersteps* and the *wall clock time (Makespan)* the proposed algorithms take to reach within a particular Δ threshold from the near-optimal solution. We also consider the *scalability* of the algorithms given by the observed speedup as we increase or decrease the number of VMs on which the algorithms are run.

4.4 Baseline Experiments

Our prior work has shown that a naïve subgraph-centric implementation of the PageRank algorithm (**SGPR**) is only comparable to, but does not outperform, the vertex-centric PageRank algorithm [34]. The BlockRank algorithm [22], intuitively, is expected to improve the performance by leveraging the availability of the entire subgraph in local memory. We initially evaluate the effectiveness of this native BlockRank algorithm (**BRNA**), and compare it with the SGPR algorithm. Unless otherwise noted, all experiments are run thrice and the average of their values plotted.

We run the SGPR and BRNA algorithms on the AMZN and CARN graphs partitioned across 6 `m3.large` VMs, and run them to 100 supersteps. At every superstep, we record the DFC for the graph, and plot it in Fig. 3.

We see that the naïve Subgraph-centric PageRank, SGPR, that we wish to out-perform continues to converge in fewer supersteps than the native BlockRank algorithm, BRNA. BRNA takes 319 *secs* and 113 *secs* to run for 100 supersteps for AMZN and CARN, respectively, compared 321 *secs* and 86 *secs* for SGPR. While BRNA seems incrementally slower, the reality is worse when we consider the superstep and time at which BRNA reaches the $\Delta_{0.0001}$ threshold. For AMZN (Fig. 3a), BRNA reaches the threshold at superstep 70 (211 *secs*) and SGPR reaches it at superstep 31 (97 *secs*), while for CARN (Fig. 3b), BRNA reaches $\Delta_{0.0001}$ at superstep 74 (91 *secs*) and SGPR reaches it at superstep 32 (43 *secs*). So *BRNA is almost twice as slow as SGPR* in reaching convergence.

This is counter-intuitive, but can be explained. First, BRNA is reliant on a strict block-like structure of the graphs seen in web graphs, with subgraphs having a high edge density that borders on cliques [22]. The graph datasets from real-world networks that we consider do not have such an extreme block structure, and consequently the algorithm performs worse. Second, while GoFFish's partitioning algorithm (METIS) tries to reduce edge cuts between partitions and then identifies subgraph within each partition, the quality of partitioning may result in small subgraphs with just $100's$ of vertices, or split a large block across two subgraphs in different partitions. As a result, we see that BRNA is not usable as is for the general class of graphs, and for a subgraph-centric paradigm. This empirically motivates the need for better PageRank algorithms that can leverage the subgraph-centric programming abstraction that is available.

Note that the near-optimal solution is calculated using SGPR's PageRank value at superstep 100, and one would expect that the DFC for SGPR at superstep 100 to be 0 in Fig. 3. However, this is not the case since every run of the algorithm does not deterministically produce identical PageRank values. As a result, the DFC proximity measure is a more appropriate metric. Even there, *having a DFC value smaller than* 0.0001 *may not be meaningful*. So we adopt $\Delta_{0.0001}$ as our upper bound of PageRank quality.

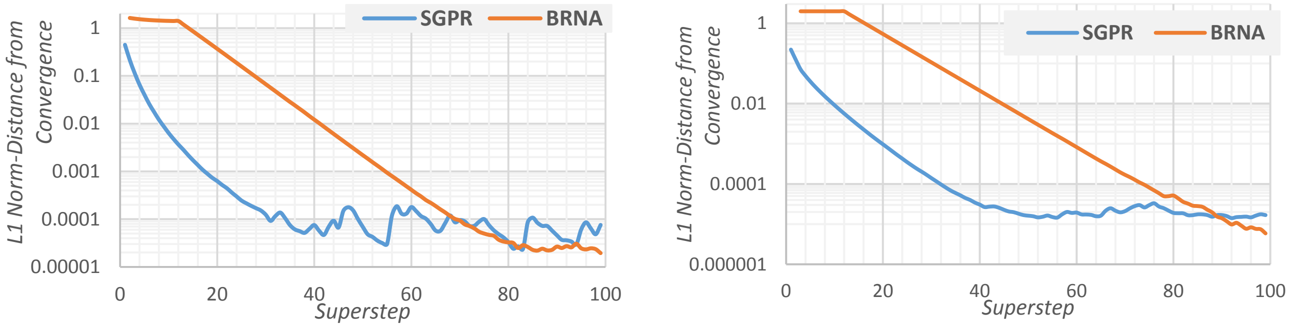
4.5 Quality Analysis of BlockRank Variations

We evaluate the proposed five variations to the BlockRank algorithms (**BRDL, BRIV, BRDI, BRNO**) from § 3, including the Subgraph Rank (**SGRK**) algorithm, and compare them against the SGPR and BRNA baseline algorithms. Fig. 4 shows the DFC for these seven algorithms for the AMZN and the CARN graphs when run on 6 `m3.large` worker VMs to 100 supersteps. We can make several observations from these plots.

The DFC for BlockRank variations are affected by the three phases of their algorithm: LPR, BlockRank and GPR. The artifact of LPR phase is seen in the DFC value at the initial superstep, of BlockRank phase (if present) over the next 10 supersteps, and the GPR phase for subsequent supersteps. Overall, we can see that SGRK consistently outperforms all the other algorithms, while the BlockRank variants, BRNA, BRDL, BRIV, BRDI, and BRNO underperform the baseline SGPR. However, there are subtle but consistent patterns unique to each phase, and each algorithmic variation that reflects the different initialization vectors and BlockRank distribution logic that are attempted. We discuss these next to help explain how SGRK is a successful refinement of the other BlockRank variations.

The BlockRank algorithms that use the native initialization vector of SGC-inverse – BRNA, BRDL and BRNO – end up with a larger distance from convergence at the first superstep, with a DFC value > 1.0. However, the algorithms that use the proposed alternative SG-by-G initialization vector end up with a lower (better) DFC value at their initial superstep (e.g. < 0.1 for AMZN and < 0.001 for CARN). This confirms our hypothesis that using a normalized BlockRank initialization value that considers the number of vertices in the subgraph is fairer and offers a better initial estimate of the eventual PageRank value.

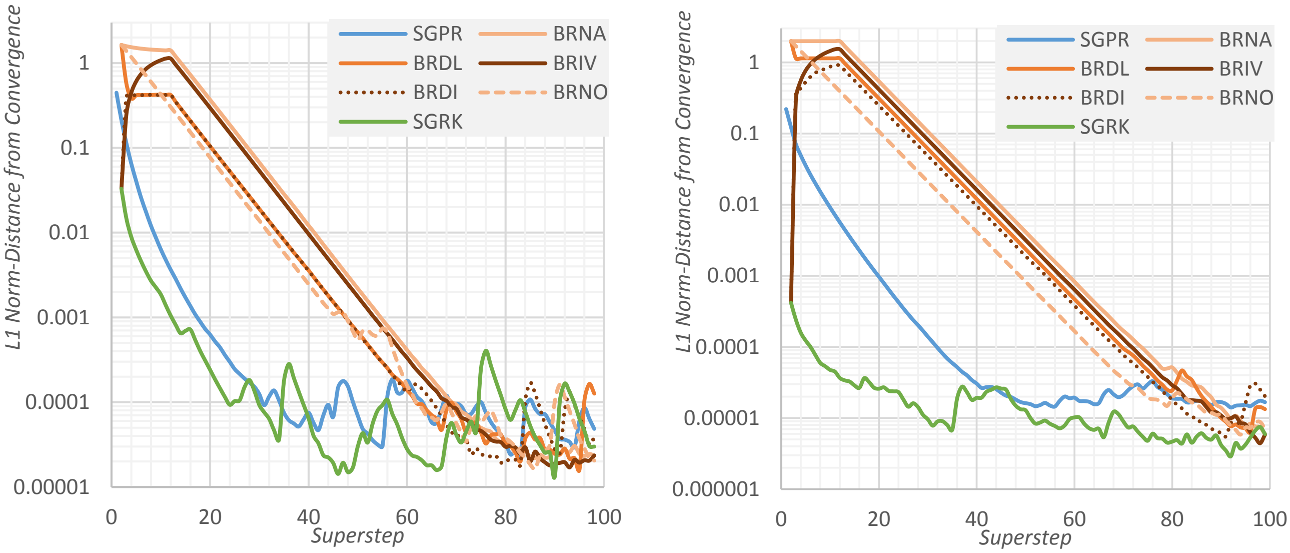
In the initial supersteps, when the BlockRank phase is taking place, we see the distinctive behavior of the algorithms. BRNA and BRIV both use the native BlockRank distribution logic that not only includes remote edges in its computation of BlockRank but also the internal edges within a subgraph and the local PageRank for their vertices (Eqn. 6. At the end of the BlockRank phase, the DFC for both these two algorithms end up at near identical points – gradually decreasing from a high DFC for BRNA, and sharply increasing from a lower DFC for BRIV to ~ 1.0 for AMZN. Even when using the alternative PageRank-like distribution logic



(a) AMZN

(b) CARN

Figure 3: DFC at every superstep, using SGPR and BRNA algorithms, for AMZN and CARN graphs.



(a) AMZN

(b) CARN

Figure 4: DFC at every superstep, using SGPR, the five BlockRank variations – BRNA, BRDL, BRIV, BRDI and BRNO – and SGRK algorithms, for AMZN and CARN graphs.

for BRDL and BRDI, we see a similar trend whereby the BlockRank phase causes the DFC values to converge to a similar point of ~ 0.4 for AMZN, though this value is closer to convergence than using the native distribution. In effect, the BlockRank phase, using either of the distributions, effectively converges to a specific value that does not offer any benefits. We observe that this constant value is close to $\frac{1}{|\mathbb{S}|}$ for the native BlockRank distribution.

Note that during the BlockRank phase, we assume that the PageRank value for each vertex in a subgraph at each superstep is the product of its local PageRank from the LPR phase and the current BlockRank value for the subgraph at that superstep. This allows us to compute the DFC at each superstep. Also, since the BlockRank phase operates on the meta-graph that is distributed across p partitions, the maximum diameter of the meta-graph is p, and this leads to the BlockRank values converging rapidly within a few supersteps.

In fact, BRNO, that skips having a BlockRank phase and directly uses a constant value as the BlockRank manages to make steady progress in converging during the 10 BlockRank supersteps it would otherwise have spent calculating BlockRank. The slope of the reduction in DFC is steady for the five BlockRank variants once they reach the global PageRank phase. Hence the initial PageRank at the start of the GPR phase impacts how quickly the algorithms converge.

SGRK, however, performs significantly better than the other BlockRank variations as well as SGPR. It leverages the best features of using a fair initialization vector like BRIV and BRDI, giving it an early advantage, and also avoids the pitfalls of the BlockRank phase, like BRNO. The impact of the initialization phase using the normalized SG-by-G weights is tangible. We see that the DFC at the start of the global PageRank in superstep 2 for SGRK is much smaller than the initial DFC for SGPR in superstep 1, with DFC values of 0.03291 vs. 0.21355 for AMZN (Fig. 4a), and 0.00042 vs. 0.22030 for CARN (Fig. 4b). This order(s) of magnitude improvement in the initial BlockRank phase is the key

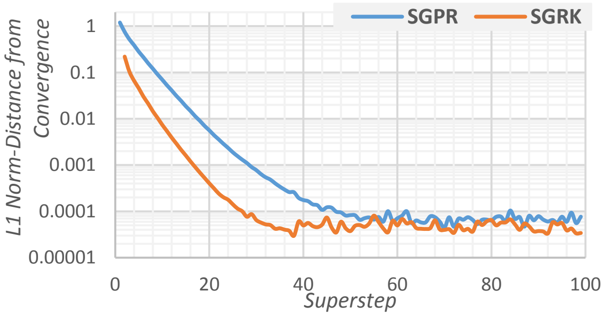


Figure 5: DFC at every superstep, using SGPR and SGRK algorithms, for WIKI graph.

benefit that we expect from leveraging the subgraph-centric programming abstraction. The subsequent supersteps, once we are in the global PageRank phase, show more modest reductions in DFC that is comparable to SGPR.

This benefit extends to the WIKI graph too. Fig. 5 shows the DFC for the WIKI graph when running SGPR and SGRK on 6 `m3.xlarge` worker VMs to 100 supersteps – the larger VMs are necessary to ensure that the graph partitions and messages buffered at the end of each superstep fit in memory. We see that the SGRK has a DFC of 0.22010 at superstep 2 while SGPR has a DFC of 1.19161 when it is initialized, and SGRK also has a sharper convergence slope for subsequent supersteps than SGPR.

As such, SGRK is qualitatively better than the other BlockRank algorithms or the naïve PageRank algorithm. The local PageRank values which we compute in the first superstep are relatively close to the near-optimal PageRank values. This is thanks to the partitioning that is done during graph deployment to minimize edge cuts across partitions and balance the number of vertices in each partition. This has the effect of identifying block-structures that may exist, and manifesting them as subgraphs that are more generally applicable to even graphs without a strong block structure (like CARN). When normalizing the local PageRank values to bootstrap the global PageRank phase, we factor in the fraction of vertices held by the subgraph as it enables fairer allocation and is robust to for subgraphs of varying sizes

4.6 Performance Analysis of BlockRank Variations

Performance is another important metric which determines the success of an algorithm. The proposed algorithms have different time complexities for their different phases. For the LPR phase, the time complexity is $O(m \times (V_i^{MAX} + E_j^{MAX}))$, where each in-memory iteration m is linear in terms of vertex count and edge count of the largest subgraph. For the BlockRank phase, the time complexity is similar since it is like PageRank applied to the meta-graph formed from the subgraphs as vertices: $O(10 \times (|\mathbb{S}| + E_i^{remote}))$, where 10 is the constant number of supersteps the BlockRank phase runs for, $|\mathbb{S}|$ is the number of subgraphs (meta-vertices) in the graph and E_i^{remote} is the number of remote (meta) edges. For the GPR phase, the time complexity is $O(\frac{q}{p} \times (|\mathbb{V}|+|\mathbb{E}|))$ where $|\mathbb{V}|$ and $|\mathbb{E}|$ are the number of vertices and edges for the whole graph, p is the number of partitions, and q is the number supersteps taken to converge to a required Δ value. As we can see, the LPR and BlockRank phases have much smaller time complexity than the GPR phase, and hence the more progress that can be made in the former toward convergence, the better the performance.

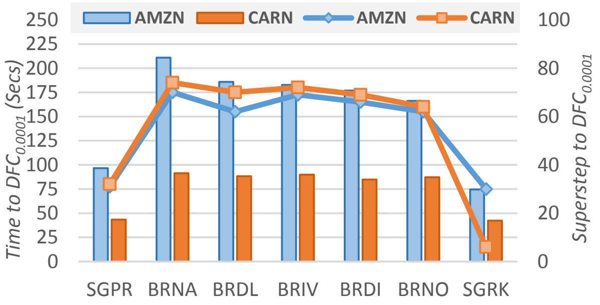


Figure 6: Time taken to reach $\Delta_{0.0001}$ (primary Y Axis, bar) and superstep reached at (secondary Y Axis, line), using the different algorithms, for AMZN and CARN graphs.

We quantify the performance of these algorithms by plotting the time that they take to achieve $\Delta_{0.0001}$, and the superstep that they achieve this in. Fig. 6 shows these plots for the five BlockRank variations and the two baseline algorithms for AMZN and CARN. These were run on 6 `m3.large` worker VMs.

The BlockRank algorithms spend the first few supersteps in the initialization and LPR phases, after which they switch to the GPR supersteps that is similar to SGPR. For all algorithms, the time taken per superstep in the GPR phase is uniform and observed to have a linear time complexity. So we see from the figure that the number supersteps and the time taken are usually proportional (line and bar graphs). For e.g., we observe (not plotted) that for AMZN, both SGPR and SGRK take 3.2 *secs* per superstep, with less than a second spent in the initial supersteps for SGRK. On the other hand, for CARN, both SGPR and SGRK spend 0.66 *secs* per superstep while the initialization superstep takes about 29 *secs* for SGRK. This is because, CARN has about 4× more vertices and edges than AMZN, and the LPR phase in each subgraph takes longer to converge to an L1 Norm of 0.0001. This slower LPR phase also explains why SGRK takes about 41 *secs* to converge to $\Delta_{0.0001}$ despite reaching there in 6 supersteps. The WIKI graph takes about 123 *secs* per superstep for SGPR and SGRK, and a relatively modest 36 *secs* is spent in LPR for SGRK. Note that WIKI runs on an `m3.xlarge` machine which is twice as capable as `m3.large` that CARN runs on. Most of the time (97%) per superstep for WIKI is taken for passing PageRank update messages over the network between supersteps, compared to the time spent on computing the PageRank (3%).

As we can see from Fig. 6, SGPR and SGRK reach $\Delta_{0.0001}$ earlier than the other BlockRank variations. BRNA is the slowest due to poorer BlockRank initialization vector and the time spent on the BlockRank phase. But this extra time taken during the initial supersteps does not offer a faster convergence. The other BlockRank variants are marginally better than the baseline BRNA. We also note that SGRK is consistently faster than SGPR for both AMZN and CARN, giving 23% and 2% speed improvements, respectively. Also, while CARN is a larger graph than AMZN, it converges

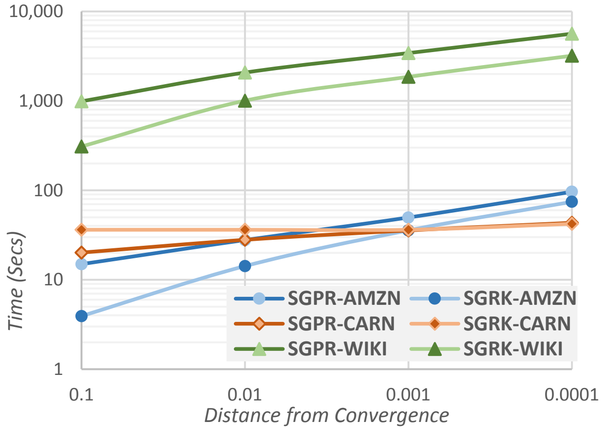


Figure 7: Time taken to reach $\Delta_{0.1-0.0001}$ using SGPR and SGRK for AMZN, CARN and WIKI graphs.

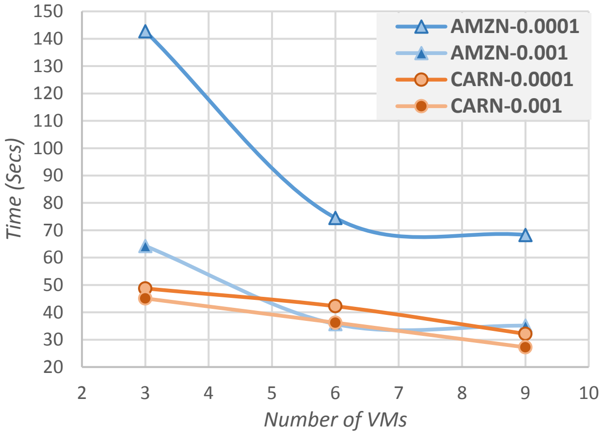


Figure 8: Time taken to reach $\Delta_{0.001}$ and $\Delta_{0.0001}$ using SGRK for AMZN and CARN graphs, as number of VMs increase from 3 to 9.

faster. This is because CARN has a much larger diameter than AMZN, which means subgraphs are sparsely connected, and hence local PageRank values dominate and network-overhead due to edge cuts across partitions is lower. Since the algorithms are often network bound than compute bound, this leads to a lower per-superstep time for CARN.

The performance benefits of SGRK over SGPR is even better, in many cases, when a lower PageRank quality is adequate. Fig. 7 shows the time taken to reach quality values of $\Delta_{0.1}$, $\Delta_{0.01}$, $\Delta_{0.001}$ and $\Delta_{0.0001}$ for AMZN, CARN and WIKI graphs for both these algorithms. These were run on 6 `m3.large` worker VMs for AMZN and CARN, and on as many `m3.xlarge` VMs for WIKI. SGRK for AMZN converges to $\Delta_{0.01}$ 49% faster than SGPR, while it converges to $\Delta_{0.0001}$ 23% faster than SGPR. Similarly, we see that the time for SGRK on WIKI is smaller than SGPR by $69\%, 52\%, 46\%$ and 43% for $\Delta_{0.1}$, $\Delta_{0.01}$, $\Delta_{0.001}$ and $\Delta_{0.0001}$, respectively. This means that SGRK gives comparable results as SGPR in half the time for WIKI, saving between $11-40$ *mins*. These indicate diminishing returns as we run more supersteps: the rapid gain in convergence distance within the first few supersteps of the Subgraph Rank is mitigated as more time is spent on the GPR supersteps.

SGRK for CARN does not outperform SGPR for larger values of DFC, above 0.0001. Since CARN has a large diameter of 849, the LPR phase took much longer to converge to an L1 Norm of 0.0001. So even though the GPR supersteps were fewer to reach a higher quality, and each GPR superstep is faster, the initial overhead of LPR cannot be surmounted. In fact, just the LPR phase lands the DFC value at < 0.001, and the incremental time in the GPR phase to reach $\Delta_{0.0001}$ is relatively negligible.

4.7 Scalability of Subgraph Rank

Lastly, we examine the scalability of these SGRK algorithm as the graph is partitioned to 3, 6 and 9 worker VMs for AMZN and CARN. We use the time taken to reach PageRank qualities of $\Delta_{0.001}$ and $\Delta_{0.0001}$ to estimate the scaling, and these are plotted in Fig. 8. We see that for AMZN, there is a linear speedup as the number of VMs increases from 3 to 6 to reach both $\Delta_{0.001}$ and $\Delta_{0.0001}$, with the time taken reducing from 142 *secs* to 74 *secs*. However, this benefit flattens out when moving from 6 to 9 VMs, with only an $\sim 8\%$ reduction in time. For CARN, the scaling is more gentle, giving a 25% and 66% speedup with a $2\times$ and $3\times$ more VMs. Increasing the number of VMs can increase the degree of parallelism but also cause more (and costlier, due to network I/O) supersteps to be required in the GPR phase. Having a fewer VMs can help identify larger blocks (subgraphs) per partition, and the LPR phase can potentially find an better quality initial value for the GPR that requires fewer steps to converge. However, running LPR on larger blocks (subgraphs) also increases the time complexity for it. As such, there may be a sweet spot of number of VMs for each graph that offers the best time to cost ratio.

5. RELATED WORK

There as been a large body of work on scalable graph processing platforms as well as algorithms for PageRank.

We can broadly classify large scale graph processing platforms as shared memory systems running on individual machines, parallel processing systems that use high-performance and accelerate hardware, and distributed computing systems that scale on commodity hardware. Shared memory graph analysis systems [33][6] offer memory efficient graph data structures to load and analyze networks, along with optimized graph kernel algorithms, but offer limited scalability. Some provide fast data structures such as indexes on top of large graphs to optimize the design of graph algorithms such a reachability queries [31]. Yet others provide domain-specific language (DSL) for easing the composition of graph analytics using richer abstractions like iterators, traversers and reducers, with a compiler that generates optimized OpenMP applications [19]. These shared memory systems however suffer from a hard memory limit.

Parallel graph processing frameworks such as the Parallel Boost Graph Library [15] provide graph APIs that are transparently mapped to MPI-based execution on HPC clusters, with some also mapping kernel algorithms like single source shortest path onto GPU accelerators [17]. Oth-

---

[6] http://snap.stanford.edu

ers offer a more traditional programming abstraction like BSP [12] but for massively multi-threaded systems like the Cray XMT [11]. However, access to such high end HPC systems is limited for a majority of users.

Distributed computing platforms such as commodity clusters and Clouds offer a more democratized access numerous frameworks for distributed graph processing [26,28,32,34,35] have emerged recently. These offer simple yet scalable programming abstractions such as vertex or subgraph centric models [28, 34, 35], provide a shared-memory view over distributed machines [32], or have specialized graph libraries suited for domains like machine learning [26]. Our work fits in this space, and specifically explores the use of novel abstractions such as subgraph-centric programming. However, the recency of these systems means that the suite of available distributed algorithms is limited [30] and consequently, their scalability for many graph algorithms is unknown. In this paper, we try to explore algorithmic adaptations of PageRank to effectively leverage these distributed graph programming abstractions.

Similar to GoFFish, Blogel [37] provides a block-centric graph parallel abstraction, which additionally uses a Graph Voronoi Diagram (GVD) partitioner to improve the scalability. They implement naïve PageRank and BlockRank without algorithmic optimizations, but handle the problem of PageRank loss due to sink vertices. In our algorithmic contributions, we efficiently compute the PageRank values rather than perform platform-level optimizations. Their results too show that BlockRank is much costlier than naïve PageRank, due the extra, unnecessary computations caused by poor Global PageRank initialization; this short-coming has been addressed in our work.

Significant research has gone into distributed algorithms for PageRank for partitioned graphs. Broder et al. [6] use an approach similar to BlockRank, but they compute approximate PageRank and not the actual PageRank. Their method involves computing eigenvectors, which can get costly for larger graphs. Qiuhong et al. [25] explore the notion of locality in a distributed system when running PageRank on MapReduce. They gain performance by reducing the number of MapReduce jobs per iteration, which is achieved by using subgraph as a processing unit. However, this approach resembles the naïve PageRank algorithm using a Subgraph Centric Abstraction, and MapReduce does not offer a natural way to represent iterative graph processing without costly disk I/O.

Davis et al. [8], try to estimate the global PageRank of a community by selectively crawling the non-local vertices after computing the local PageRanks to convergence in the community. Communities have a block-like structure with high internal edge degree and sparse remote edge-cut. They avoid running the PageRank algorithm on the whole graph but like us, they compute the local PageRanks is a community and then use these to estimate the Global PageRanks of the vertices in that community. We have generalized this approach to operate on dense or sparse subgraphs, and demonstrated the efficacy of this technique to diverse graphs. Kamvar et al. [21] define adaptive methods for PageRank convergence. They use the idea that several vertices reach their convergence well ahead of others, and hence they speed-up the rest of the algorithm by pruning these converged-vertices. Using this heuristic, they avoid redundantly re-computing the PageRank to offer performance gains. Such pruning can be adopted to our Subgraph Rank algorithm too as an extensions, as has been shown for performing top-k betweenness centrality using a subgraph-centric model [24].

6. DISCUSSION AND CONCLUSIONS

In this paper, we have identified deficiencies in natively mapping PageRank and BlockRank algorithms to a subgraph-centric model, and propose alternate algorithmic strategies for BlockRank. One such Subgraph Rank (SGRK) variation demonstrably outperforms the baseline and other BlockRank alternatives. Algorithmically, Subgraph Rank is better than BlockRank as it omits unnecessary input-specific optimizations; computing block-level PageRank which is negatively impacts the general case.

Considering the wide applications of PageRank, we needed a more general and better algorithm which improves the scalability and performance of PageRank. The global PageRank computation step of the BlockRank algorithm is costly since it requires more supersteps to converge than the Subgraph Rank (and even PageRank) algorithm due to the poor initialization values. Blocks of varying sizes and imbalance in the remote edges deteriorates the performance of the BlockRank algorithm, and this was motivates the need for Subgraph Rank. Subgraph Rank is a more general solution than BlockRank, and also scales better, partially due to the intelligent choice of initialization vector for the Global PageRank phase.

SGRK shows a performance gain of between $23-74\%$ for equivalent PageRank quality, for AMZN and WIKI graphs; its performance matches SGPR for PageRank qualities of $\Delta_{0.001}$ or better for CARN. SGRK successfully exploits the subgraph structure of the graph to offer high quality initial PageRank values from the LPR and BlockRank phases that help it to rapidly converge during the GPR phase. As such, our work shows the need for developing new algorithms or adapting existing ones to best utilize the innovative graph programming abstractions and storage models that are emerging.

Small-world networks like WIKI graph have a high clustering coefficient, and are sparsely connected with a low diameter. For such graphs the performance improves for Subgraph Rank due to the head-start they get in the local-PageRank (in-memory) phase, i.e., when well partitioned, the high clustering coefficient and sparse connectivity enables local PageRank values to be more significant. However, if the number of partition is much higher than the diameter of the graph, then the initial Global PageRank values can deteriorate the performance, and this is something we will study in the future. Similarly, for Barabasi-Albert model (Power Law) graphs, we have low diameters and the clustering decreases as the size of the graph increases. Hence with an optimal number of partitions for a given graph, the performance of Subgraph Rank will be better than PageRank. The effect of increase in the number of partitions is something that can be studied both empirically and theoretically.

As part of future work, we propose to examine the scalability of the SGRK algorithm to see if shows weak scaling with any of the topological features of the graph such as edge cuts. We also plan to investigate other graph algorithms that can benefit from subgraph centric abstractions. Heuristics like graph pruning are also viable in offering further performance gains.

Acknowledgment

We would like to thank Neel Choudhury from the DREAM:Lab, IISc for feedback on the paper and Arjun Kamath for offering insights on Giraph++. This work was supported through an AWS in Education Research Grant from Amazon.

7. REFERENCES

[1] Apache giraph. `giraph.apache.org`.

[2] The graph 500 list. http://www.graph500.org/, June 2014.

[3] T. Aittokallio and B. Schwikowski. Graph-based methods for analysing networks in cell biology. *Briefings in bioinformatics*, 7(3):243–255, 2006.

[4] B. Bahmani, K. Chakrabarti, and D. Xin. Fast personalized pagerank on mapreduce. In *SIGMOD*, 2011.

[5] C. Bizer, T. Heath, K. Idehen, and T. Berners-Lee. Linked data on the web. In *World Wide Web*, 2008.

[6] A. Broder, R. Lempel, F. Maghoul, and J. Pedersen. Efficient pagerank approximation via graph aggregation. *Information Retrieval*, 9(2):123–138, 2006.

[7] F. Chung and W. Zhao. Pagerank and random walks on graphs. In G. Katona, A. Schrijver, T. Sznyi, and G. Sgi, editors, *Fete of Combinatorics and Computer Science*, volume 20 of *Bolyai Society Mathematical Studies*, pages 43–62. Springer Berlin Heidelberg, 2010.

[8] J. V. Davis and I. S. Dhillon. Estimating the global pagerank of web communities. In *ACM SIGKDD*, 2006.

[9] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113, Jan. 2008.

[10] R. Diestel. *Graph Theory*, volume 173 of *Graduate Texts in Mathematics*. Springer-Verlag, 2012.

[11] D. Ediger and D. A. Bader. Investigating graph algorithms in the bsp model on the cray xmt. In *IPDPS Workshops*, 2013.

[12] A. V. Gerbessiotis and L. G. Valiant. Direct bulk-synchronous parallel algorithms. *JPDC*, 22(2):251–267, 1994.

[13] D. F. Gleich. Pagerank beyond the web. *CoRR*, abs/1407.5107, 2014.

[14] J. E. Gonzalez, Y. Low, H. Gu, D. Bickson, and C. Guestrin. Powergraph: Distributed graph-parallel computation on natural graphs. In *OSDI*, 2012.

[15] D. Gregor and A. Lumsdaine. The Parallel BGL: A generic library for distributed graph computations. In *POOSC*, 07/2005 2005.

[16] Y. Guo, M. Biczak, A. L. Varbanescu, A. Iosup, C. Martella, and T. L. Willke. How well do graph-processing platforms perform? an empirical performance evaluation and analysis. In *IPDPS*, 2014.

[17] P. Harish and P. Narayanan. Accelerating large graph algorithms on the gpu using cuda. In *HiPC*, 2007.

[18] T. H. Haveliwala. Topic-sensitive pagerank. In *WWW*, pages 517–526. ACM, 2002.

[19] S. Hong, H. Chafi, E. Sedlar, and K. Olukotun. Green-marl: A dsl for easy and efficient graph analysis. In *ASPLOS*, 2012.

[20] G. Jeh and J. Widom. Scaling personalized web search. In *WWW*, 2003.

[21] S. Kamvar, T. Haveliwala, and G. Golub. Adaptive methods for the computation of pagerank. *Linear Algebra and its Applications*, 386:51–65, 2004.

[22] S. Kamvar, T. Haveliwala, C. Manning, and G. Golub. Exploiting the block structure of the web for computing pagerank. *Stanford University Technical Report*, 2003.

[23] G. Karypis and V. Kumar. Analysis of multilevel graph partitioning. In *SC*, page 29, 1995.

[24] A. G. Kumbhare, M. Frincu, C. S. Raghavendra, and V. K. Prasanna. Efficient extraction of high centrality vertices in distributed graphs. In *HPEC*, 2014.

[25] Q. Li, W. Wang, P. Wang, K. Dai, Z. Wang, Y. Wang, and W. Sun. Combination of in-memory graph computation with mapreduce: A subgraph-centric method of pagerank. In *WAIM*, 2013.

[26] Y. Low, D. Bickson, J. Gonzalez, C. Guestrin, A. Kyrola, and J. M. Hellerstein. Distributed graphlab: A framework for machine learning and data mining in the cloud. *PVLDB*, 5(8):716–727, Apr. 2012.

[27] L. Mainetti, L. Patrono, and A. Vilei. Evolution of wireless sensor networks towards the internet of things: A survey. In *SoftCOM*, pages 1–6. IEEE, 2011.

[28] G. Malewicz, M. H. Austern, A. J. Bik, J. C. Dehnert, I. Horn, N. Leiser, and G. Czajkowski. Pregel: A system for large-scale graph processing. In *SIGMOD*, 2010.

[29] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, November 1999.

[30] S. Salihoglu and J. Widom. Optimizing graph algorithms on pregel-like systems. In *VLDB*, 2014.

[31] S. Seufert, A. Anand, S. Bedathur, and G. Weikum. Ferrari: Flexible and efficient reachability range assignment for graph indexing. In *ICDE*, 2013.

[32] B. Shao, H. Wang, and Y. Li. Trinity: A distributed graph engine on a memory cloud. In *SIGMOD*, 2013.

[33] J. Shun and G. E. Blelloch. Ligra: A lightweight graph processing framework for shared memory. *SIGPLAN Not.*, 48(8), 2013.

[34] Y. Simmhan, A. Kumbhare, C. Wickramaarachchi, S. Nagarkar, S. Ravi, C. Raghavendra, and V. Prasanna. Goffish: A sub-graph centric framework for large-scale graph analytics. In *EuroPar*, 2014.

[35] Y. Tian, A. Balmin, S. A. Corsten, S. Tatikonda, and J. McPherson. From think like a vertex to think like a graph. *PVLDB*, 7(3), 2013.

[36] J. Ugander, B. Karrer, L. Backstrom, and C. Marlow. The anatomy of the facebook social graph. *CoRR*, abs/1111.4503, 2011.

[37] D. Yan, J. Cheng, Y. Lu, and W. Ng. Blogel: A block-centric framework for distributed computation on real-world graphs. *PVLDB*, 7(14), 2014.

S-SUM: A System for Summarizing the Summaries

C Ravindranath Chowdary
Department of Computer Science and Engineering,
Indian Institute of Technology (BHU),
Varanasi, India 221 005
rchowdary.cse@iitbhu.ac.in

P Sreenivasa Kumar
Department of Computer Science and Engineering,
Indian Institute of Technology Madras,
Chennai, India 630 036
psk@iitm.ac.in

ABSTRACT

Query-specific summarization of multiple documents is a useful task in the current day context of the WWW, that is containing huge amount of information. When different summarizers have access to different sets of documents for a query, generating a summary of the summaries produced by the multiple summarizers becomes an interesting and useful task. In this paper, we propose an efficient solution for this problem. Sentences from the individual summaries are used to construct an Integrated Linear Structure (ILS) and are given unique position numbers. All the sentences in the ILS are then assigned weights that reflect the importance of the sentences to the given query. Sentences are selected according to these weights using the Maximal Marginal Relevance (MMR) approach for inclusion into the final summary of summaries. Finally, the sentences in the final summary are sorted based on their position numbers given using ILS. Experimental results show that *S-SUM* is efficient.

1. INTRODUCTION

Huge amount of information is present in the World Wide Web and a large amount of information is being added to the WEB regularly. Information on a topic is distributed across the multiple pages/documents. It is a nontrivial task for a user to go through all these documents to find the information of her interest. Most of the times, there will be a lot of redundancy in the information content and it will be a tedious task for the user to read all the documents. To overcome this problem, query specific multi-document summarizers were proposed [13, 15, 19, 20, 10]. With the increasing need for quality summarizers, this field is gaining momentum.

In extractive summary generation, the sentences are selected from the documents and are arranged in a meaningful order. Summaries can be generated either from a single document or from multiple documents and a summary can be either generic [18, 5] or query-specific. In this paper, we deal with query-specific extractive summarization. Choosing a sentence for inclusion in a summary is determined by the amount of importance it has with the query. The challenge lies in assigning importance to a sentence. Sentences in all the documents should be given scores based on the importance they have with the query posed by the user.

The approach used by a typical query specific multiple document summarizer is: Sentences from *all* the documents are arranged as nodes in a graph. Similarities among all the sentences are calculated using a *similarity* measure. Sentence s is selected into the summary based on both the importance of s with the query and the importance of sentences that have high similarity with s. The quality of summaries generated by these systems is good but these systems cannot be used for on-line/real time purpose due to their high computational complexity. The complexity is directly proportional to the number of sentences/nodes in the graph. One possible solution is to divide a huge set of documents into smaller sets and generate summaries on these smaller sets. By doing so, efficiency would be increased but there may be some information loss. Further, these individual summaries are to be summarized as the final summary has a restriction on its size *i.e.*, 250 words, 500 words, *etc.*

1.1 Motivation

A search engine retrieves the ranked list of documents for a given query. Each search engine may retrieve a different set of documents. These individual documents retrieved by search engines can be summarized independently *i.e.*, a different summarizer could be used for each of the search engine. In this scenario, if a user wants the overall gist/summary on a topic, the only solution is to go through all the summaries generated by different summarizers. Also, it is likely to have a fair amount of overlap between the retrieved sets of documents. This overlap of information in documents may lead to overlap of information in summaries. Therefore the individual summaries may have both diverse and redundant information. If there is a system to generate a summary from the summaries generated by different summarizers, then it would save a lot of users time.

Another scenario is if n users write their brief opinion on a topic, it would be of great help to generate a summary of these n opinions. This motivated us to develop a system that summarizes opinions/summaries. In this paper, we propose and report experimental results of a system, called *S-SUM*, that generates a *summary* from the summaries/opinions.

The 20th International Conference on Management of Data (COMAD), 17th-19th Dec 2014 at Hyderabad, India.

1.2 Introduction to S-SUM

To the best of our knowledge, there are no extractive summarizers in the literature that take query-specific summaries as input. An initial solution to this problem can be applying a query-specific multiple document summarizer on the given summaries. However, there are issues to be considered while generating summary of summaries. They are 1) ordering of sentences in the final summary. Since sentences are selected from different summaries, it is a challenging task to have a logical flow in the final summary. 2) All the summaries are query specific and the sentences within the summary are highly related to the query. So, direct centrality [15, 13] based sentence score may not give good results. Both these challenges are effectively addressed in this paper. In this paper, sentences are given scores as a combination of both centroid [5] and centrality [15, 13] methods.

2. RELATED WORK

Text summarization has picked up pace in the recent years. Multi-document generic summary generation is discussed in MEAD [5]. MEAD generates summary by following centroid based approach. Given a set of documents about a particular topic *i.e.*, a cluster of documents, a centroid of the cluster is calculated. Each sentence in the cluster is given a score- with respect to the centroid obtained, similarity with the first sentence in the document and the relative position with respect to the first sentence. Sentences are selected in the decreasing order of sentence scores and are arranged chronologically.

Centrality based summarization approaches are discussed in [1, 8, 9, 7, 17]. Degree centrality is discussed in [1] and eigenvector centrality is discussed in [8, 9, 7]. Degree centrality of a node is calculated by counting the number of other nodes it is connected to i.e., degree of a node. An edge is placed between two nodes if and only if there is a considerable amount of overlap of words between the nodes. Concept of bushy path was introduced in [1]. A node with high degree is called as bushy node. A path connecting top n bushy nodes is bushy path. Eigenvector centrality of a node is calculated by taking into consideration both the degree of the node and the degree of the nodes connecting to it, this is inspired by PageRank[2].

In [12] topic focused single document summarization is addressed. Document is modeled as a graph. Irrespective of the threshold condition, an edge is placed between adjacent nodes in the graph. Node score is calculated with respect to the query. A minimal set of nodes are picked which will cover(include) all the query terms. This minimal set may not be having direct edges among them, so, intermediate nodes are added to make the selected set of nodes a connected sub graph of the original graph.

Query specific summary generation is discussed in [11]. In [11] a document is modeled as a graph and similarity between nodes is calculated using *cosine similarity* measure. An edge is present between nodes if the similarity value exceeds a threshold. Node scores are calculated by taking both term frequencies(tf) and inverse document frequencies (idf) into consideration. A node gets high score if it is connected to the nodes with high score. The node scores are computed iteratively till the values converge. Query is considered as a node of the document graph in[13] and pairwise similarity between sentences is calculated and an edge is placed between nodes if the similarity value is greater than *zero*. As the query is part of the graph, centrality based approach is followed to select the nodes into the summary.

Generating Non-Redundant summaries is addressed in [3]. Node scores are calculated based on the similarity w.r.t the query and the summary is generated incrementally. To start with, a node with highest score is selected into the summary. All the scores of remaining nodes are recalculated based on both their current node scores and the similarity with the nodes already selected into the summary. From the recalculated scores, the node with the highest score will be added to the summary. In most of the models discussed above, node scores are calculated by following ideas similar to PageRank[2] and HITS[4] and edge scores are calculated based on the amount of similarity between nodes. All the models mentioned above address the issue of generating summary from a single document or multiple documents for a generic/specific purpose. None of these systems address the problem of generating a summary of summaries that are generated from different summarizers on different sets of documents for a given topic.

3. FRAMEWORK

The task in query specific summary generation is to extract sentences from the documents. These sentences should be very relevant to the query. A summary is said to be *complete* if it contains information about the whole query. Completeness is calculated based on the presence of query terms in a summary. A summarizer will extract a sentence from the documents based on the score given to it. The order in which the sentences are extracted need not follow a logical sequence. Hence, these sentences have to be rearranged to get a logical flow to the summary. The quality of having logical flow in a summary is termed as *coherence*. In multi-document summarization, some amount of information would be present in more than one document. If the information that is repeated across the multiple documents is found to be important, such information should be present in the summary but should not be repeated. Such non-repetition of information in a summary is termed as *non-redundancy*. In this paper, coherence and non-redundancy are addressed explicitly and completeness is achieved implicitly.

In this paper, we model all the sentences from all the summaries as nodes in a graph. Each sentence is considered as a vector of words. An edge is placed between two sentences if similarity between them is above a threshold. *Similarity* is calculated as given in Equation 1

$$sim(\overrightarrow{n_i}, \overrightarrow{n_j}) = \frac{\overrightarrow{n_i}.\overrightarrow{n_j}}{|\overrightarrow{n_i}||\overrightarrow{n_j}|} \tag{1}$$

where $\overrightarrow{n_i}$ and $\overrightarrow{n_j}$ are term vectors for the nodes n_i and n_j respectively. Weight of each term(t) in the term vector is calculated as $tf_t * isf_t$ where tf_t is the *term frequency* and isf_t is *inverse sentential frequency*. isf_t is calculated as $log(\frac{n}{n_t+1})$ where n is the total number of nodes in the graph and n_t is the number of nodes containing the term t in the graph. All the stop words are removed and the remaining words are stemmed before computing the weights.

4. INTEGRATED LINEAR STRUCTURE (ILS)

In this section, we discuss the construction of *ILS*. *ILS* contains all the sentences of the summaries. Each sentence in *ILS* is assigned a position number. Sentences of the final summary are arranged in ascending order of these position numbers. We observed that this arrangement of sentences ensures coherence in the final summary. In this paper, we assume that the individual summaries from which *ILS* is constructed are coherent. We also observed that a summary generated on a smaller set of documents will be more coherent than a summary generated on a larger set. The summary with the highest number of sentences is taken as the base summary. Each sentence in the base summary is taken as a *context* node. Position numbers are assigned to these context nodes as the integral multiples of the parameter "gap". *ILS* is constructed by inserting every sentence from the remaining summaries into the base summary. Each non-context sentence will be in the *Neighbourhood* of a context node. *Neighbourhood* of a context node c is the set of all sentences from remaining summaries that have highest similarity with c when compared other context nodes. The similarity value should be above a threshold. This threshold has to be chosen appropriately. If the similarity value of a node is less than this threshold with all the context nodes, then the node is introduced as a new context node.

For example, if *gap* is taken as 200, the position values of the contexts will be *0, 200, 400, 600, 800, etc.* All the nodes that are neighbours to the context node at 200 will be inserted between the context nodes with position numbers 200 and 400 respectively *i.e.,* all the neighbours of 200 will have position values in the range [201,399]. The exact procedure for positioning of neighbours is described in the later part of this section. The concept of *neighbourhood* is introduced based on the following observations *i,e.,*

- As all the summaries are generated for a query/topic, it is highly likely to have similarity between sentences across the summaries.
- These similar sentences may be having information on a topic.
- These similar sentences may have repetition of information.

So, all the nodes that have similar information are introduced as neighbours to their context node. This arrangement is useful to achieve logical flow in the final summary. Usually, the nodes with new information are introduced as new context nodes. A node is said to have a new information if it is not similar with any of the existing context nodes. In this case, it is better to introduce it as a new context node. This will be useful to identify the theme of the summary, discussion of which is out of scope of this paper.

In Algorithm 1, the construction of ILS is given. The base summary, S_0, is identified and position numbers are assigned to the nodes in the base summary. Each node in the base summary is made as a context node. The parameter *gap* is a constant that is chosen appropriately to accommodate the neighbours. In Lines 13 - 29, similarity of a sentence d with all the context nodes is computed. The context node with which d has highest similarity is found. If this similarity value exceeds a threshold, η(>0.1), then d is assigned a position number as explained in Algorithm 2. Otherwise, d is made as a new context node as outlined in Algorithm 3.

Algorithm 1 Construction of Integrated Linear Structure

```
1: Input: Summaries arranged in the decreasing order of their size(number of sentences)
2: Output: Integrated Linear Structure $ILS$
3: Integrated Linear Structure $ILS = S_0$ {//base summary}
4: $k = |S_0|$
5: $j = 0$
6: while $j < k$ do
7:     {//Assign position to each node $n_j$ in $ILS$}
8:     position($n_j$) = $j * gap$
9:     $j++$
10: end while
11: $i = 1$
12: while $i \leq$ number of $Summaries$ do
13:     for each node $d \in S_i$ do
14:         {// find the context node, $MaxSimilarContextNode$, with which $d$ has maximum similarity}
15:         for each context node $n_j$ do
16:             if $MaxSimilarContextNode = Null$ then
17:                 $MaxSimilarContextNode = n_j$
18:             else
19:                 if $Sim(MaxSimilarContextNode, d) < Sim(n_j, d)$ then
20:                     $MaxSimilarContextNode = n_j$
21:                 end if
22:             end if
23:         end for
24:         if $sim(d, MaxSimilarContextNode) \geq \eta$ then
25:             Introduce $d$ as a neighbour in the context of $MaxSimilarContextNode$ as explained in Algo 2
26:         else
27:             Make $d$ as the new context in $ILS$ as explained in Algo 3
28:         end if
29:     end for
30:     $i++$
31: end while
```

In Algorithm 2, inserting a node d in the neighbourhood of a context node is discussed. All the neighbours of a context will have increasing position values in the decreasing order of their similarity with the context. For example, if the context node at *200* has nodes with position numbers *201, 202 and 203* as neighbours then *sim(nodeAtPosition(200), nodeAtPosition(201))* will be greater than *sim(nodeAtPosition(200), nodeAtPosition(202))*. d will be given position value accordingly. Algorithm 3 gives the methodology for insertion of a new context. If d is the first node in a summary, then it is added as a new context, positioned immediately after the current last context. If the current last context has position number 600 then d will take position number 800 (recollect that we assumed *gap* value as 200). If d is not the first node of a summary, then it is added as a context, following the context in which the parent of d is present. $parent(d)$ is the node which precedes d in the summary. If the $parent(d)$ has position value 302 then d will take position number 400. If the $parent(d)$ has position value 800 (*i.e.,* context node) then d will take position number 1000. Note that all the sentences (including duplicates) from the summaries will be

Algorithm 2 Insertion of a node in the neighbourhood of the context node

```
Input: Partially constructed ILS, context node($c$) and the node to be inserted($d$)
Output: ILS with the node inserted as a neighbour to $c$
$i$ = position($c$) + 1
while ((A node is present at position $i$) AND (sim($c, nodeAtPosition(i)$) $\geq$ sim($c, d$))) do
    $i++$
end while
{//Increment the position values of all the neighbouring nodes of $c$ that are having position numbers greater than or equal to $i$}
$j = countNeighbours(c)$
$m = j$
while $m >= i$ do
    $positionOfNodeAt(m) = positionOfNodeAt(m) + 1$
    $m--$
end while
Place $d$ at position $i$
```

included in ILS.

5. SUMMARY OF SUMMARY GENERATION

Each node in the ILS is assigned a score that is calculated based on both the importance with respect to the query and its importance across the ILS. A part of our node score calculation is inspired by the method proposed in [11]. A significant role is played by the neighbouring nodes i.e., if a neighbour of the node contains a query relevant information then the node is assigned a positive score even in the absence of query relevant information in it. The *neighbourhood* of a node is the set of all the nodes that have a similarity value above a threshold with the node. *Note that neighbourhood in this section is different from previous section.*

$$X_{q_i}(s) = d \frac{sim(s, q_i)}{\sum_{m \in N} sim(m, q_i)} \qquad (2)$$

Equation 2 computes relevancy of a node with a query term. d is the bias factor which lies between 0 and 1, N is the set containing all the nodes of ILS and $sim(i, j)$ is computed as given by Equation 1. The value of $X_{q_i}(s)$ will be *zero* in the absence of q_i in s. It will have a positive value if q_i is present in s. The denominator in Equation 2 will be small if q_i is present is fewer nodes. Therefore if a node contains a query term that is present in fewer nodes, then the value of $X_{q_i}(s)$ will be higher. Conversely, if a node contains a query term that is present in majority of the nodes, then the value of $X_{q_i}(s)$ will be lesser.

$$Y_{q_i}(s) = (1-d) \sum_{v \in adj(s)} \frac{sim(s, v)}{\sum_{u \in adj(v)} sim(u, v)} w_{q_i}(v) \qquad (3)$$

Equation 3 computes the score by considering node scores of neighbours of s. $adj(i)$ is the set of all nodes in N that have similarity value above 0.1 with the node i. The bias factor d is the trade-off between these two equations i.e., Equations 2 and 3. The value of d is determined empirically. If d is chosen close to 1 then more importance is given to the similarity of a node with the query and less importance is given to its neighbours. We experimentally found that $d = 0.85$ is the optimal value. Computation of Equation 3 is repeated till the convergence is achieved in Equation 4.

Algorithm 3 Adding a context to the partially constructed ILS

```
Input: Partially constructed $ILS$ and node($d$) that is to be inserted as a context
Output: $ILS$ with $d$ included
if $d$ is the first sentence/node of the summary then
    ADD $d$ as a new context node to ILS, following the current last context node and set position($d$) = position($CurrentLastContextNodeOfILS$) + $gap$
else
    $i = \lfloor position(parent(d))/gap \rfloor * gap + gap$
    for each node $n$ starting from position $i$ do
        position($n$) = position($n$) + $gap$
    end for
    Insert $d$ as a context node and set position($d$) = $i$
end if
for each non context node $m$ do
    if $sim(m, d) > sim(m, context(m))$ then
        $x = position(m)$
        $y = countNeighbours(context(m))$
        Insert $m$ as a neighbour to $d$ by applying Algo 2
        {//Decrement the position values of all the nodes that were following $m$ in previous context}
        $z = y - x$
        $p = 0$
        while $p < z$ do
            $positionOfNodeAt(x+1) = positionOfNodeAt(x+1) - 1$
            $x++$
            $p++$
        end while
    end if
end for
```

$$w_{q_i}(s) = X_{q_i}(s) + Y_{q_i}(s) \qquad (4)$$

$$W_Q(s) = \sum_{1 \leq i \leq t} w_{q_i}(s) \;+\; \frac{1}{|N|} \sum_{m \in N \ \& \ m \neq s} sim(s, m) \qquad (5)$$

The node score for each node with respect to each query term $q_i \in Q$ where $Q = \{q_1, q_2, ..., q_t\}$ is computed using Equation 4. Equation 4 is iterated till the scores converge. Here, $w_{q_i}(s)$ is the node score of node s with respect to query term q_i. Score of a node with respect to a query Q is calculated using Equation 5, $W_Q(s)$ is the summation over node scores calculated with respect to each query term using Equation 4 and second part of Equation 5 is to capture the salience of s's information in ILS. For a given query Q, node scores for each node with respect to each query term are calculated. So, a node will have a high score if:

1. It has information relevant to the query.
2. It has neighbouring nodes sharing query relevant information.
3. It has information that is in majority of nodes.

After assigning scores, the node with the highest score is selected as the first sentence to the summary. The node n_i that has the highest score among the remaining $(|N| - 1)$

nodes, calculated with Equation 6, will be included into the summary. s_j is the node that is in the summary. In a similar fashion, other nodes are selected into the summary. Selection process continues till the target summary size is reached. After completing the selection, sentences in the summary are arranged in the increasing order of their position values (computed during ILS construction).

Our claim is: this rearrangement preserves coherence in the summary generated. Recollect, our assumption that individual summary generated on a small set of documents is coherent. In accordance with this assumption, the *base summary* is coherent. ILS is constructed with this *base summary* as its skeleton. This structure remains intact expect during augmenting a new context. This mechanism ensures logical flow to be preserved during the construction of ILS. This is the reason behind achieving a better flow by rearranging the sentences in the order of their position numbers.

$$\underset{i}{Max}\{\lambda W_Q(n_i) - (1-\lambda)\underset{j}{Max}\{sim(n_i, s_j)\}\} \qquad (6)$$

Equation 6 is inspired by Maximal Marginal Relevance (MMR) [3] approach. λ ranges from 0 to 1. If λ is 1 then the most responsive sentence with respect to query is chosen. If λ is 0 then the least redundant sentence is chosen. If λ is close to 1 then more importance is given to responsiveness and less importance to redundancy. λ has to be appropriately chosen so that a node is selected if it is having information highly relevant to the query (first part of the equation) and if its selection does not add redundancy (second part of the equation) to the already selected set. Note that Equation 6 is used to select nodes into the summary and the original node scores that are calculated using Equation 5 are unaltered. The proposed methodology for summary generation captures both the importance of a sentence with respect to query and its significance across the summaries. In this way, both centrality and centroid methods are integrated while assigning score to a sentence. Therefore, the performance of our system is expected to be good. Experimental results show this fact.

6. EXPERIMENTAL SETUP AND RESULTS

S-SUM requires summaries as its input. A summarizer is required to generate summaries on the subsets of documents on a given topic/query. For this purpose, we have chosen a system called $ESUM$ [16]. $ESUM$ is one of the efficient query specific text summarizer. In fact, any such summarizer can be used to provide the input to *S-SUM*, but we found $ESUM$ to be as good as any other summarizer. The performance of $ESUM$ on DUC 2005 [1] data is close to the best system of DUC 2005. In any case, using a better individual summarizer will further improve the performance of *S-SUM*. We compared the results of *S-SUM* with the results of $ESUM$ run on the *entire* cluster of documents on a specific topic, using the standard $ROUGE$ (Recall-Oriented Understudy for Gisting Evaluation) measure used in the community and found that performance of *S-SUM* is better than $ESUM$.

6.1 Experimental Setup

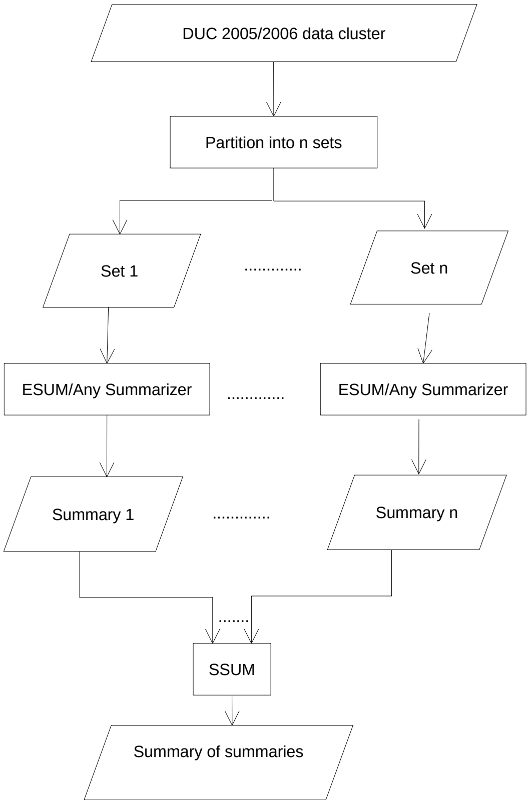


Figure 1: A block diagram of experimental setup

The DUC data has 50 clusters and each cluster has number of documents discussing about a particular topic. Figure 1 outlines the detailed experimental setup for a cluster. Documents in a cluster are divided into n disjoint subsets. For example, if we take number of documents in a cluster as 25 and the number of disjoint subsets as 3, then the first 8 documents are assigned to the first set and the next 8 to the second set and the remaining to the third set. Figure 1 illustrates the block diagram of experimental setup. In our experiments, we have chosen the number of sets to be *three*. $ESUM$ is used to summarize the documents in each set. After summarization, each set has one summary, called a partial summary[2]. All these partial summaries will be given as an input to *S-SUM*. *S-SUM* summarizes these three summaries. As DUC data has 50 clusters, this process is repeated on all the clusters.

To the best of our knowledge, there is no other system in literature which performs the task of generating summary from the summaries. Therefore, we have designed our own methodology to evaluate the performance of *S-SUM*. The summary of summaries generated by *S-SUM* is compared with the summary generated by $ESUM$ on the *whole* cluster. We also give $ROUGE$ [6] values of summaries of individual sets and the summary generated by $ESUM$ on these three summary sets.

6.2 Results

---

[1] http://duc.nist.gov

[2] We call it partial because the summary is generated on a subset of the cluster

Table 1: ROUGE Values on DUC 2005 Data

System	ROUGE-1	ROUGE-2	ROUGE-SU4
ESUMC	0.37167	0.07140	0.12768
System-15	0.37515	0.07251	0.13163
S-SUM	0.37733	0.07284	0.13121
SIGIR08	0.35006	0.06043	0.12298
SET1	0.36344	0.06273	0.12054
SET2	0.36444	0.06296	0.12089
SET3	0.35654	0.06301	0.11874
ESUMS	0.35746	0.06384	0.12008

Table 2: ROUGE Values on DUC 2006 Data

System	ROUGE-1	ROUGE-2	ROUGE-SU4
ESUMC	0.38215	0.07514	0.13160
System-24	0.41108	0.09558	0.15529
S-SUM	0.39884	0.08223	0.14037
SET1	0.37846	0.07236	0.12874
SET2	0.37619	0.06910	0.12698
SET3	0.37493	0.07215	0.12835
ESUMS	0.38303	0.07531	0.13248

To evaluate the quality of the summaries generated, *DUC* provides us with *ROUGE* [6] values. Also, *DUC* provides model summaries that are written by volunteers. *Recall* value is calculated for the generated summaries with respect to these model summaries. Through *ROUGE* values, we can determine the quality of a summary as these values are computed by comparing the summary with the summaries written by volunteers.

ROUGE-N uses n-gram recall measures of system generated summaries and the summaries generated by the volunteers(model summaries). *ROUGE-N* is calculated as given in Equation 7

$$ROUGE\text{–}N = \frac{\sum\limits_{s \in model\ summaries} \sum\limits_{gram_n \in s} count_{match}(gram_n)}{\sum\limits_{s \in model\ summaries} \sum\limits_{gram_n \in s} count(gram_n)} \quad (7)$$

Here n is the length of a n-gram. $gram_n$ stands for a specific n-gram. $Count_{match}(gram_n)$ is the maximum number of n-grams co-occurring in both the generated summary and in the reference summaries. *ROUGE-1* is the recall measure of uni-grams. *ROUGE-2* is the recall measure of bi-grams. *ROUGE-SU4* is the recall measure which computes the skip bi-grams with skip distance four and uni-grams are also considered while computing this measure. Finer details of *ROUGE* measurements can be found at [6].

The values of all the variables in the equations are fixed empirically after experimenting on test data of *DUC*. The same set of values are used for both *DUC* 2005 and 2006. The values for bias factor d is 0.85 (based on [11]) and λ is 0.6 (based on [3]). The values in the tables indicate that the generated summaries are consistent and the quality of summaries in terms of the *ROUGE* measures is satisfactory.

All the values in the tables are the *mean* of 50 clusters of DUC. *ESUMC* values are computed on the summaries generated by *ESUM* over the complete set of documents in a cluster. System-15 and System-24 are the best performing systems at DUC 2005 and 2006 respectively. $SIGIR08$ [14] is an efficient summarizer but *ROUGE* values are available for DUC 2005 only. *SETi* values are computed on summaries generated by *ESUM* on the i^{th} set of cluster as discussed earlier. *ESUMS* values are computed on summaries that are generated by summarizing partial summaries generated by *ESUM* (*i.e.,* 3 partial summaries). The *ROUGE* values in the tables clearly demonstrate the performance of *S-SUM* system. Though our system is given only 3 summaries as input, the performance is comparable with the best system of DUC. Note that systems at DUC generate summary on the whole cluster. On contrary $S - SUM$ generates summary on partial summaries. So, the performance of DUC systems should be much better than *S-SUM* and in fact it is unfair to compare our system with DUC systems. But the performance of $S - SUM$ is much better than we anticipated. Also, the performance of our system is dependent on the quality of the input summaries. As *ESUM* was close to the performance of the best system, system-15 in DUC 2005, *S-SUM* was able to outperform system-15. This result is in fact a surprise. In DUC 2006, *ESUM* was well behind the best system, system-24, so, *S-SUM* was not able to outperform system-24.

S-SUM generates summary of summaries efficiently and it can also be used as an additional module to the existing summarizers: majority of the summarizers follow an unified approach (all the documents are merged into a single document or inter and intra similarities among the sentences in the documents are taken into consideration) for generating multi-document summaries. To boost the efficiency and performance, the document cluster can be partitioned into multiple sets and a summary can be generated for each set using a summarizer. All these summaries can be given as an input to *S-SUM* to generate summary of summaries and as evident from the results, the quality would be better than the summary generated by that summarizer on the whole cluster. The choice of the partition is by itself a challenging problem. Overall, by partitioning a cluster, both efficiency is achieved and the quality of summary is also improved.

7. CONCLUSIONS

In this paper we have addressed the issue of generating summary of summaries that are generated by extractive summarizers. We have developed a system called *S-SUM* that generates summary from summaries. Integrated Linear Structure(ILS) is introduced by us. *ILS* preserves logical flow between sentences from different summaries. Experimental results demonstrate that the methodology introduced for summary generation produces quality output. Coherence of the summaries is preserved via the position values given to sentences in the *ILS*. Non-redundancy and completeness are achieved by MMR approach. The encouraging experimental results suggest that *S-SUM* can also be used to boost the performance of any extractive summarizer. One limitation of *S-SUM* is it assumes that the base summary is coherent.

8. REFERENCES

[1] Gerard Salton and Amit Singhal and Mandar Mitra and Chris Buckley. Automatic text structuring and summarization. *Inf. Process. Manage.*, 33(2):193–207, 1997.

[2] L. Page and S. Brin and R. Motwani and T. Winograd. The pagerank citation ranking: Bringing order to the

web. pages, 161–172, Brisbane, Australia, 1998.

[3] Jaime G. Carbonell and Jade Goldstein. The use of mmr, diversity-based reranking for reordering documents and producing summaries. pages, 335–336, Melbourne, Australia, 1998. ACM.

[4] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *J. ACM*, 46(5):604–632, 1999.

[5] Dragomir R. Radev and Hongyan Jing and Malgorzata Budzikowska. Centroid-based summarization of multiple documents: sentence extraction, utility-based evaluation, and user studies. pages, 21–30, Seattle, Washington, 2000. Association for Computational Linguistics.

[6] Chin Yew Lin and Franz Josef Och. Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics. pages, 605–612, Barcelona, Spain, 2004. Association for Computational Linguistics.

[7] Rada Mihalcea. Graph-based ranking algorithms for sentence extraction, applied to text summarization. page, 20, Barcelona, Spain, 2004. Association for Computational Lingu.

[8] Güneş Erkan and Dragomir R. Radev. LexPageRank: Prestige in multi-document text summarization. pages, 365–371, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[9] Mihalcea, Rada and Tarau, Paul. In . pages, 404–411, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[10] Jagadeesh, J and Pingali, Prasad and Varma, Vasudeva. A relevance-based language modeling approach to duc 2005. In *Proceedings of Document Understanding Conferences (along with HLT-EMNLP 2005), Vancouver, Canada*, 2005.

[11] Jahna Otterbacher and Güneş Erkan and Dragomir R. Radev. Using random walks for question-focused sentence retrieval. pages, 915–922, Vancouver, British Columbia, Canada, 2005. Association for Computational Linguistics.

[12] Ramakrishna Varadarajan and Vagelis Hristidis. A system for query-specific document summarization. pages, 622–631, Arlington, Virginia, USA, 2006. ACM Press.

[13] Xiaojun Wan and Jianwu Yang and Jianguo Xiao. Manifold-ranking based topic-focused multi-document summarization. pages, 2903–2908, Hyderabad, India, 2007.

[14] Dingding Wang and Tao Li and Shenghuo Zhu and Chris Ding. Multi-document summarization via sentence-level semantic analysis and symmetric matrix factorization. pages, 307–314, Singapore, Singapore, July 2008. ACM.

[15] M Sravanthi and C R Chowdary and P Sreenivasa Kumar. QueSTS: A query specific text summarization system. pages, 219–224, Florida, USA, may 2008. AAAI Press.

[16] C Ravindranath Chowdary and P Sreenivasa Kumar. ESUM: An efficient system for query-specific multi-document summarization. In *ECIR '09: Proceedings of the 31th European Conference on IR Research*, pages, 724–728, Toulouse, France, April 2009. Springer.

[17] C. Ravindranath Chowdary and M. Sravanthi and P. Sreenivasa Kumar. A system for query specific coherent text multi-document summarization. *International Journal on Artificial Intelligence Tools*, 19(5):597–626, 2010.

[18] Alexandra Balahur and Mijail Alexandrov Kabadjov and Josef Steinberger and Ralf Steinberger and Andrés Montoyo. Challenges and solutions in the opinion summarization of user-generated content. *J. Intell. Inf. Syst.*, 39(2):375–398, 2012.

[19] Yin, Wenpeng and Pei, Yulong and Zhang, Fan and Huang, Lian'en. Query-focused multi-document summarization based on query-sensitive feature space. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages, 1652–1656, New York, NY, USA, 2012. ACM.

[20] Zhang, Lanbo and Zhang, Yi and Chen, Yunfei. Summarizing highly structured documents for effective search interaction. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, pages, 145–154, New York, NY, USA, 2012. ACM.

A comparative study of two models for celebrity identification on Twitter

MS Srinivasan *
IBM India, Embassy Golf Links
Koramangala Intermediate
Ring Road, Bangalore
smuthusw@in.ibm.com

Srinath Srinivasa
IIIT-Bangalore,26/C,
Electronics City, Hosur Road,
Bangalore
sri@iiitb.ac.in

Sunil Thulasidasan
Los Alamos National
Laboratory
NM, USA
sunil@lanl.gov

ABSTRACT

The concept of *celebrities* has shaped societies throughout history. This work addresses the problem of celebrity identification from social media interactions. “Celebritiness” is a characteristic assigned to persons that are initially based on specific achievements or lineage. However, celebritiness often transcends achievements and gets attached to the person itself, causing them to capture popular imagination and create a public image that is bigger than life. The celebrity identification problem is argued to be distinct from similar problems of identifying influencers or of identification of experts. We develop two models for celebrity identification. In this paper, we compare the two models on twitter data and highlight the characteristics of each of the models.

Keywords: Celebrity, Social media, Twitter, Influence

1. INTRODUCTION

All societies have celebrity figures that are admired, respected or idolized. Celebrities are well-known personalities, and their actions attract a lot of attention. They are therefore a subject of interest for marketing teams, policy makers, social workers, preachers, teachers, etc. Understanding the dynamics of celebrity formation and identification of potential celebrities is an important problem of interest.

Web based social media adds an extra dimension to celebrity dynamics. Usually, celebrities in the real world are also well-known [27] and admired on social media, and are also employed to promote causes and interests. However, the participatory nature of social media often breeds its own celebrities. There are several cases of personalities who go on to become well-known in the outside world due to their celebrity status on social media. Similarly, several well-known people are not savvy enough on social media to elicit the same level of adulation as they receive in the real world.

A celebrity is not the same as an influencer. Influence of a person is typically specific to a topic [11] and the person’s position in the social network. Often, lesser known persons may end up wielding enormous influence on a topic because of their network centrality.

Celebrities, on the other hand, are well-known personalities. They are usually idolized by a subset of the population, and their actions receive more than average attention from the population. While, celebrities are typically influencers, not all influencers are celebrities. Also, when influence is defined as *persuasive power*, celebrities typically do not fare better than influencers in our immediate social network. People are far more likely to be persuaded by a close friend or a family member, than by a celebrity.

“Celebritiness” is characterized more by the *attention* and *adulation* that it elicits, and not by its persuasive or influencing abilities. Thus, celebrities are of interest more as “fronts” or the face of a particular product or idea, that primarily aims to catch attention or create awareness about the product or the idea[1].

While influence is a characteristic associated with a topic or event, “celebritiness” is a characteristic associated with a *person*. A characteristic definition of a celebrity, attributed to Boorstin [4], says: *A celebrity is the person who is well-known for their well-knownness.* Once a person acquires a celebrity status, they become the object of attention themselves.

Also, while celebrities are well-known personalities, not all well-known persons are celebrities. People could also be well-known for notorious reasons. Celebritiness is also not about fame for a specific reason. People usually become famous based on their specific achievements in some domain (winning the Wimbledon) or because of their lineage (son/daughter/spouse of a celebrity). But a celebrity is one, whose fame *transcends* the specific reasons why they became famous in the first place, and becomes associated with the person itself, giving them a larger than life persona in popular imagination.

With this perspective, we have found that celebrity identification per se, has not received much interest from researchers in social media. In this paper, we address the space of celebrity dynamics and explore two computational models for identifying celebrities from social media data,

---

[*] This author is a part-time research scholar at IIIT-Bangalore and this research work is carried out as part of his MS thesis

[1] http://mediakix.com/2013/10/fashion-panel-celebrities-vs-influencers-wins/. Last accessed: 08 Aug 2014.

The 20th International Conference on Management of Data (COMAD),
17th-19th Dec 2014 at Hyderabad, India.

based on Twitter. A preliminary form of the first model called Acquaintance-Affinity-Identification (AAI) was proposed by the authors in [26]. This model is based on the notion of source credibility and source attractiveness. In this paper, we propose another model called Action-Reaction (AR), that is based on loyalty and attention. Our AAI and AR models are derived from two different theories in social psychology that look at orthogonal factors towards celebritiness. In this paper, we also provide a comparative study on the two proposed models.

2. RELATED LITERATURE

2.1 Models of influence and expertise

A lot of research effort has gone into identifying "influencers" and computational models for "influence maximization." Motivating applications include viral marketing, sales prediction and counter terrorism. Some examples are [3, 28, 20, 1, 15, 7, 9, 8]. Though several models are proposed for influence maximization in networks, most of them are found to be intractable. In response, several heuristic approximations have been proposed [28, 3, 9, 8] based on degree, centrality-based measures, greedy heuristics and data mining approaches. Such approaches apply diffusion principles to mimic the "word of mouth" behavior in human social environments for spread of information.

Hajian, et al. [16] propose a formal model for measuring influence on *FriendFeed* social network[2]. This model computes "Magnitude of Influence" (MOI) for each user based on the number of hits generated for a user posting. The model then computes the "Influence Rank" using MOI. Further, Gosh, et al. [14] define influence as the number of in-network votes a user's post generates and applied it on *Digg* social network[3].

Forestier, et al. [12] propose a framework for extracting celebrities from the online discussions[4]. This framework uses three different meta-criteria. The first meta-criteria identifies the potential celebrities who have more than average number of in-degree, out-degree, posts compared to other people in the community. The second meta-criteria is based on the participation of the user in different forums. The third meta-criteria is based on the citations of names and the quoted texts.

Our problem of finding celebrities is slightly different from the influence maximization problem. Celebritiness is a characteristic attributed to people and have to do with who they are, more than their position in the network.

A related problem is of identification of "experts" in a community and recommending experts based on need [10, 30, 31]. While people with superlative expertise tend to become celebrities, expert identification is not the same as celebrity identification. Expertise is defined within some context or topic, while a celebrity figure may be context-free. Celebrities need not be experts and not all experts are likely to be celebrities.

2.2 Models of Celebrity

The concept of celebrities has been a topic of interest in social science and social psychology for several decades. A definition of celebrity by Mills [24] says:

> Celebrities are names that need no further identification. *Those who know them so far exceed those of whom they know as to require no exact computation.* Wherever they go, they are recognized, and moreover, *recognized with some excitement and awe.* Whatever they do has publicity value. More or less continuously, over a period of time, they are the material for the media of communication and entertainment.

The emphasized parts of the definition above provide vital inputs into understanding and building computational models for celebrity identification.

Another popular definition for a celebrity from Boorstin [4] that says: *A celebrity is the person who is well-known for their well-knownness*, suggesting a recursive or self-fulfilling nature of the phenomenon of celebrity formation.

In social psychology, research on the topic of celebrity endorsement rests on two general models:

1. The source-credibility model
2. The source-attractiveness model

The source-credibility model [18] defines celebritiness of a communicator to be a function of expertise and in turn, credibility. In this model, "expertness" is defined as the perceived ability of the celebrity to make valid assertions and "trustworthiness" is defined as the perceived willingness of the celebrity to make value assertions. Celebrities exhibiting expertness and trustworthiness are credible and to this extent, persuasive.

The source-attractiveness model [23], considered to be a component of an earlier model called the "source valence" model and draws on the research in social psychology [22], considers factors like: *familiarity*, *likeability*, and *similarity* between the source of communication and its recipient. In this model, "familiarity" is defined as knowledge of the celebrity through exposure. The second factor, "likeability" is defined as affection for celebrity as a result of celebrity's physical appearance and behavior. The last factor, "similarity" is defined as a supposed resemblance between the celebrity and consumers who are mostly celebrity fans. This model holds that celebrities who are known to, liked by, and/or similar to the consumer are attractive and, to this extent persuasive.

One of our models, called the AAI model [26] discussed in this paper is based on a combination of the source-credibility and the source-attractiveness model. This model is simplified to three factors and adapted to fit into Twitter communications.

Another model by Friedman and Friedman [13] determines celebritiness based on three other factors: *attention*, *recall* and *loyalty* from the population. The *attention* factor measures the amount of attention the celebrity gets from their fans. The *recall* factor defines the ability of the fan to re-collect the celebrity names. The *loyalty* factor measures the fans' loyalty towards the celebrity by providing support for the given celebrity.

The source-credibility and source-attractiveness models focus on *familiarity*, *likeability* and *similarity* between the celebrity and the fans. But the Friedman and Friedman

---

[2] http://friendfeed.com

[3] http://digg.com

[4] http://huffingtonpost.com

model [13] focus on *attention*, *loyalty* and *recall* factors. As the two models of social psychology look at orthogonal factors, we develop our second computational model called the AR model, inspired from the Friedman and Friedman model. Further, the properties of the celebrities identified using the two models are compared.

2.3 Influence Models on Twitter

Twitter being a popular social media in recent days, lots of research has been performed on its data sets to analyze influence. Cha, et al. [6] measure influence in Twitter using in-degree, replies and mentions. They find that the most followed users do not necessarily score highest on the other measures. PageRank-like scores have been proposed to measure influence on Twitter [29]. The *Social Network Potential (SNP)* score [2] is based on the average of two measures *Retweet and mention ratio* and *Interactor Ratio*. *Retweet and Mention ratio* is calculated as the amount of tweets that are amplified or lead to a communicative action between the communicator and another user divided by total amount of tweets of the communicator. The Interactor Ratio is measured as the ratio between the number of users who retweet or mention the communicator and the total amount of followers of the communicator.

Weng, et al. [29] propose *Twitter Rank*, an extension of the Page Rank algorithm, to measure influence by not only taking followers and interactions into account, but also by analyzing topical similarities with the help of a ranking method similar to PageRank. An interesting aspect of this work is that in the analyzed sample of Singapore-based users a high reciprocity (e.g.,mutual following relationship) was found.

Kwak, et al. [19] compared three different measures of influence: number of followers, Page Rank, and number of retweets, finding that the ranking of the most influential users differed depending on the measure.

Hatcher, et al. [17] develop an influence metric on twitter based on both "content" and "conversation" aspect. The "content" aspect is measured based on the number of tweets posted by the user and also based on the number of tweets posted by the members in the user's network. The "conversation" aspect is measured based on the number of replies, retweets and mentions received from the members of the user's network and also based on the number of replies, retweets and mentions received by the members of the user's network.

3. IDENTIFICATION OF CELEBRITIES

In our work, we use Twitter as the social media of interest, for celebrity identification. The participatory nature of online social media adds a new dimension to the celebrity problem. Usually celebrities of the outside world are also treated as celebrities in social media, attracting a lot of followers. So just looking at follower count may appear to be a good measure of celebritiness. However, this is not always the case [6].

There is a thriving ecosystem of "buying" followers[5][6][7] on Twitter, discrediting the follow score. In addition, celebrities from the outside world, often are not as active or elicit the same amount of attention for their activities, inside Twitter.

We hence develop and explore two computational models of celebrity scoring. The first, called the AAI model (for *acquaintance*, *affinity* and *identification*) is derived from the source-credibility and source-attractiveness models used in social psychology. We provide interpretations for acquaintance, affinity and identification on Twitter. The second model called the *Action-Reaction* (AR) model is a more direct measure of one's fame, based on the reactions they elicit.

Before introducing the specific models, we first formally describe the Twitter dataset on which the models were built.

3.1 Twitter data model

A Twitter dataset is formally modeled as follows:

$$\mathcal{D} = (T, N, \alpha, \mu, \rho, \tau, \gamma) \tag{1}$$

Here T is the set of tweets in the sample and N is the set of twitter accounts in the sample. The terms α, μ, ρ, τ and γ refer to ensembles of functions representing different elements of the Twitter universe. They are described in detail below.

α refers to an ensemble of functions around authorship. The function $author : T \to N$ maps a tweet to its author. The function $tweets : N \to 2^T$, gives the set of all tweets authored by a given twitter account. Given a set of tweets W, we will also be using a function $authors : 2^T \to 2^N$, defined as

$$authors(W) = \bigcup_{t \in W} author(t)$$

that gives the set of authors, given a set of tweets.

μ, ρ and τ refers to an ensemble of functions pertaining to mentions, replies and retweets respectively. The functions $mentionsof : N \to 2^T$, $repliesof : N \to 2^T$ and $retweetsof : N \to 2^T$ define the set of tweets that represent mentions, replies or retweets of a given twitter account.

Analogously, the functions *mentionsby*, *repliesby* and *retweetsby* are defined, which are all of the form $N \to 2^T$ and define the set of mentions, replies and retweets performed by an author.

γ refers to an ensemble of functions pertaining to followship. The function $follows : N \to 2^N$ depicts the set of accounts followed by a twitter account. Analogously, the function $followers : N \to 2^N$ gives the set of followers for a given account.

3.2 The AAI Model

The first of the two models, called the AAI model, is presented here. Based on the various studies in the psychology of celebrity [23, 18, 22] around source-credibility and source-attractiveness, we arrived at three attributes defining a celebrity:

1. Well-knownness
2. Likeability
3. Identification

A celebrity is someone who is well-known. But they are not just well-known, they are also *liked* by the population. Finally, idolization of celebrities happen because a

---

[5] http://pinchlikes.com/buy-twitter-followers#. Last accessed: 21 June 2013.

[6] http://www.wordstream.com/blog/ws/2013/05/16/buying-twitter-followers-pros-and-cons#. Last accessed: 21 June 2013.

[7] http://buytwitterpro.com/buy-twitter-followers#. Last accessed: 21 June 2013.

large number of people in the population *identify* with the celebrity in some form.

It can be seen that the three measures can be pipelined to form three hierarchical layers. We can only like someone whom we know, and we can only identify ourselves with someone whom we like. This gives us the 3-layer AAI model.

The proposed 3-layer model is called the AAI model and has three separate scores:

1. Acquaintance score (A)
2. Acquaintance-Affinity score (AA)
3. Acquaintance-Affinity-Identification score (AAI)

Acquaintance is a measure that determines how well a person is known in the community. Affinity measure determines how much the person is liked by the community and the Identification measure determines the extent to which others identify themselves with the person being studied. Their interpretations on Twitter are provided below.

3.2.1 Acquaintance Score

Acquaintance Score $A(i)$ for twitter account i is the measure of the number of people who knows the person i as a proportion of the population of the sample. In Twitter, acquaintance is established by *any* evidence that depicts knowledge of one account by another. This includes a follow or reply or retweet or mention.

The acquaintance score $A(i)$ for twitter account i thus given by:

$$A(i) = \frac{|followers(i) \cup authors(mentionsof(i) \cup repliesof(i) \cup retweetsof(i))|}{|N|} \tag{2}$$

3.2.2 Acquaintance-Affinity Score

Acquaintance-Affinity Score $AA(i)$ is a measure of how well a person is liked by the community and by whom. The affinity score is weighted by the acquaintance score so that being liked by well-known people increases the affinity content, as compared to being liked by lesser known persons.

Affinity is measured as a function of how much others respond to the activities of the person in question. It might be argued that reaction to one's action may also be due to animosity. While it may well be the case, it is unlikely that animosity will elicit sustained reactions over time. In addition, to "love to hate" someone can also be viewed as some form of affinity. Person j who "loves to hate" i is perhaps displaying envy which in turn is a form of affinity for what constitutes the characteristics of person i.

To measure affinity, we calculate three scores. Reply score (R), Mention Score (M) and Retweet Score (RT).

Reply Score $R(j|i)$ is defined as a conditional probability measure. Given that the person i replied to a tweet, the probability that the tweet was created by person j is represented as the Reply Score $R(j|i)$

$$R(j|i) = \frac{|repliesby(i) \cap repliesof(j)|}{|repliesby(i)|} \tag{3}$$

Mention Score $M(j|i)$ and Retweet Score $RT(j|i)$ are calculated in an analogous fashion.

There have been observations in the literature that retweets and replies are dependent on various factors like content influence, commonality in interests and network influence [25, 21]. However, since celebritiness is associated with people rather than with content or issues, we find these nuances irrelevant for our problem.

The Acquaintance-Affinity score AA(j) is then calculated as;

$$\begin{aligned} AA(j) = \sum_{i \in E_r} & A(i) * R(j|i) + \\ \sum_{i \in E_m} & A(i) * M(j|i) + \\ \sum_{i \in E_{rt}} & A(i) * RT(j|i) \end{aligned} \tag{4}$$

where

$$E_r = authors(repliesof(j))$$

$$E_m = authors(mentionsof(j))$$

$$E_{rt} = authors(retweetsof(j))$$

3.2.3 Acquaintance-Affinity-Identification Score

The final layer of scoring is the AAI score, which is a measure of how well a person is identified in the community and how likeable are the people in the community who identified the person in question.

To measure how well the person is identified by the people in the community, we use the follower measure. The rationale here is that following someone is a decision for the long term – indicating that we value their tweets in our timeline.

Then the Acquaintance-Affinity-Identification Score $AAI(j)$ is calculated as;

$$AAI(j) = \sum_{i \in followers(j)} \frac{AA(i)}{|follows(i)|} \tag{5}$$

Thus, the AA score of a person is divided among all the people followed by them, to contribute to their AAI score. The AAI score is tagged as the celebrity score.

3.3 The Action-Reaction Model

While the AAI model is based on scores assigned by users to other users, there is no focus on the amount of activity around a celebrity. This prompted us to develop another model based on observing how much of activity does a person elicit and how well it reflects his/her celebrity status. This model is called the Action-Reaction (AR) model. The AR model is based on the Friedman and Friedman model around attention, recall and loyalty. We arrived at two attributes of defining a celebrity:

- Attention
- Loyalty

The Action-Reaction model is based on two measures: Action measure and Reaction measure. The Action measure attributes the amount of *loyalty* fans pay to the celebrity. The attention is measured in terms of replies, mentions, retweets in Twitter. And the Reaction measure attributes to the amount of *attention* the celebrity elicits for every action.

The Action measure is a conditional probability measure. For a given person j, the Action measure of person i towards

j computes the probability that if i has acted on someone else's tweet, what is the probability that the tweet was from j.

$$A(j|i) = \frac{|A_m(i \to j) \cup A_r(i \to j) \cup A_{rt}(i \to j)|}{|mentionsby(i) \cup repliesby(i) \cup retweetsby(i)|} \tag{6}$$

where

$$A_m(i \to j) = mentionsof(j) \cap mentionsby(i)$$

$$A_r(i \to j) = repliesof(j) \cap repliesby(i)$$

and

$$A_{rt}(i \to j) = retweets(j) \cap retweetsby(i)$$

The Reaction measure is also a conditional probability measure. Given that the tweet from person j has elicited a response, the reaction measure for a target person i measures the probability that the response was from i. It is given by:

$$R(i|j) = \frac{|A_m(i \to j) \cup A_r(i \to j) \cup A_{rt}(i \to j)|}{|tweets(j)|} \tag{7}$$

The Action score $A(j)$ of person j is measured as the sum of all the action measures of the set of people S who acted upon j:

$$A(j) = \sum_{i \in S} A(j|i) \tag{8}$$

We normalize the Action score between 0 and 1. The normalized Action score $\|A(j)\|$ is calculated as follows:

$$\|A(j)\| = \frac{A(j)}{A_{max}} \tag{9}$$

where A_{max} is the maximum Action score across the sample.

Similarly, the Reaction score $R(j)$ of the person j is measured as the sum of all the reaction measures of the set of people S who reacted up on j.

$$R(j) = \sum_{i \in S} R(i|j) \tag{10}$$

We normalize the Reaction score between 0 and 1. The normalized Reaction score $\|R(j)\|$ is calculated as follows:

$$\|R(j)\| = \frac{R(j)}{R_{max}} \tag{11}$$

where R_{max} is the maximum Reaction score across the sample.

The action score $\|A(j)\|$ and reaction score $\|R(j)\|$ represents two dimensions of the celebrity j. The action score measures the *loyalty* of the person's fans and the reaction score measures the *attention* celebrity j elicits for every action. We represent the action-reaction measure as a vector $\vec{AR}_j$.

$$\vec{AR}_j = (\|A(j)\|, \|R(j)\|) \tag{12}$$

Since the scores are normalized, the maximum values they take is 1. We represent the "ideal celebrity measure" as $\vec{I} = (1, 1)$. We then measure the cosine similarity θ_j between the Action-Reaction vector $\vec{AR}_j$ and the ideal celebrity measure $\vec{I}$.

$$\theta_j = \frac{\vec{AR}_j \cdot \vec{I}}{\|AR_j\| \|I\|} \tag{13}$$

The celebrity score $C(j)$ is then represented as the product of the magnitude of $\vec{AR}_j$ and the cosine similarity θ between action-reaction $\vec{AR}_j$ and "ideal measure" $\vec{I}$.

$$C(j) = \theta_j * \vec{AR}_j \cdot \vec{I} \tag{14}$$

We can see that the celebrity score $C(j)$ can be simplified as:

$$C(j) = \frac{\|A(j)\| + \|R(j)\|}{\sqrt{2}} \tag{15}$$

4. EVALUATION RESULTS

Dataset

We have used Twitter APIs to collect tweets, user details and relationship between users (follow network). Twitter APIs impose limitations [8] on the number of requests per application for a time duration. Considering these limitations, We started of with a seed of 10 users, identified randomly from the tweet streams.

We collected all the tweets from the users' timeline. We also collected user details and the people who they followed. The new users identified in this step formed the level 1 users. This was repeated till a depth of three. The table 1 shows statistics about our evaluation dataset

Table 1: Tweet Data set Statistics

Number of Users	66 thousand
Number of Tweets	99 million
Number of Replies	6.9 million
Number of Mentions	21.9 million
Number of Re-tweets	4.8 million

Figure 1 shows the follower distribution in our data set. Figure 2 shows tweet distribution. The follower distribution and the tweet distribution of our data set resembles much of the large twitter data set [29]. We use this as an indicator of the representativeness of our sample.

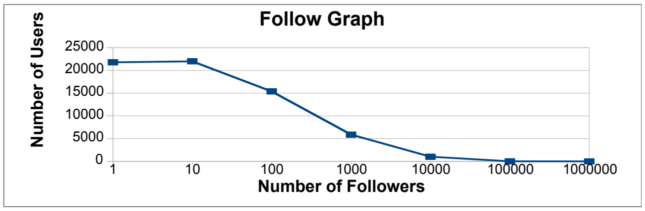


Figure 1: Followers Distribution

Celebrity and Influencers

For purposes of comparison, we identified two existing algorithms to find influencers: PageRank [5] and SNP (Social

---

[8] `https://dev.twitter.com/docs/rate-limiting/1.1/limits`

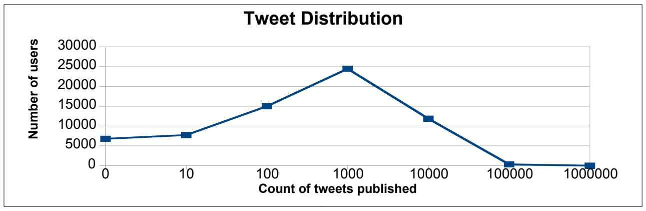


Figure 2: Tweet Distribution

networking potential) [2]. A more recent measure called the Twitter rank [29] was rejected in favor of PageRank. This is because, Twitter rank measures *influence* based on topical similarities, which is quite different from the celebritiness problem. "Celebritiness" is associated with a person and can be independent of topic. Hence we used the more straightforward PageRank measure for comparison. We used JUNG (Java Universal Network/Graph Framework) for computing PageRank.

We picked the top 25 celebrities identified from each of the algorithms (AAI model, AR model, PageRank and SNP) and we prepared a single list of celebrities for user evaluation by removing duplicates. We then presented the celebrity list of 65 celebrities to the 200 volunteers for user evaluation. We requested volunteers to identify or vote for celebrities from the given list. The user evaluation was "blindfolded" – meaning, the volunteers were unaware of the algorithms behind the celebrity list used for the evaluation.

We picked up the top 20 celebrity candidates based on user votes. Then we compared the top 20 celebrities voted by the volunteers with the top 20 celebrities identified by each of the algorithms.

Figure 3 shows that the AAI model performed well in terms of identifying celebrities. Celebrities identified by the AR model also agreed with user votes. But in addition to people, it also identified popular twitter accounts like YouTube, Überfacts, Funny Tweets and others, which were not identified as celebrities by users. We discuss the characteristics of the celebrities identified by AAI model and AR model in the later section.

We also computed the *average agreement* among the users based on the user votes in the top 20 results. Both AAI model and AR model showed an average agreement of 65% among users voting for these candidates.

Most celebrities identified in our algorithm have a good influence scores as well, going by their SNP measures. This result indicates that most celebrities are influencers where as not all influencers are celebrities. Table 2 shows the top 20 celebrities identified by various algorithms used in the experiment.

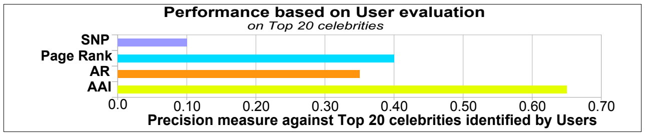


Figure 3: User evaluation results

We also compared the performance of the algorithms with Forbes top celebrities list and Klout Score [9]. We took top 25 people from the Forbes celebrity list Celebrities[10] and top 25 people from the Forbes India celebrity list (as the initial seed users considered in the sample were from India) [11] and merged them as 50 Forbes celebrities. We removed names from this list that did not feature in the sample at all.

Table 2: Top 20 Celebrities

AAI Model	AR Model	Page Rank Model	SNP Model
Barack Obama	Justin Beiber	OMG Facts	The Wanted
Kim Kardashian	Youtube	I Do That Too	Delta Goodrem
Amitabh Bachchan	Niall Horan	yfrog Social	Follow @WizKhalifa
Rihanna	Liam Payne	yfrog	ZoomTV
Justin Bieber	zaynmalik1D	Techmeme	Saj
Ellen DeGeneres	Shah Rukh Khan	Mediagazer	Beyonce Knowles
Oprah Winfrey	Jay Sean	Barack Obama	Star Plus
Shah Rukh Khan	Harry Styles	Kim Kardashian	iTunes
Lady Gaga	Yuvraj Singh	The White House	Annie Mac
Dalai Lama	Xstrology	Oprah Winfrey	Pope Francis
Kanye West	Amitabh Bachan	Ellen DeGeneres	LMAO
Priyanka	Uber Facts	jimmy fallon	RDB
OMG Facts	Ariana Grande	Office of VP Biden	New York Post
jimmy fallon	Lady Gaga	Conan O'Brien	DJ Khaled
Conan O'Brien	Funny Tweets	LMAO	Blake Griffin
Drizzy	Nicki Minaj	Michelle Obama	darkchild
Karan Johar	Jai J.D. Brooks	Kanye West	Keri Hilson
Ryan Seacrest	Mr.Carter	Rihanna	Sohanny
Abhishek Bachchan	Bruno Mars	Lady Gaga	Shriya Saran
Salman Khan	Rihanna	Ashton Kutcher	Kanye West

We computed the precision measure and Figure 4 shows the comparative performance of different algorithms. We observed that our AAI model performed well compared to other algorithms. This result shows the celebrities identified by the AAI model as also globally recognized celebrities.

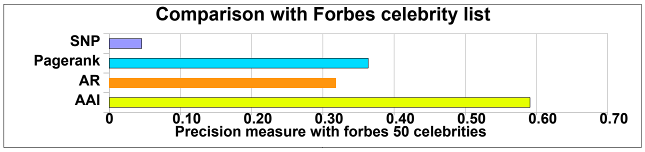


Figure 4: Comparison with Forbes 50 Celebrities

We identified the Klout score[12] manually for each of the top 25 celebrities identified by each of the algorithms. We plotted the Klout score against each celebrity identified from 4 algorithms. Figure 5 shows that the celebrities identified by the AAI model and AR model have consistent Klout score unlike the PageRank and SNP model.

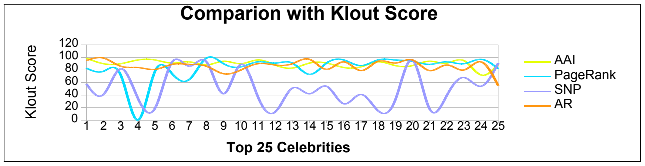


Figure 5: Comparison with Klout Score

In order to match the negatives, we measured the Klout scores of people who featured in the *bottom* 20 of the AAI and AR model as well. The average Klout scores for the top 20 and bottom 20 groups are showed in Table 3. The Klout scores seem to be positively correlated with both AAI and AR scores, but since it is a proprietary measure, we could not explore further than this.

---

[9] `http://www.klout.com`

[10] `http://www.forbes.com/celebrities/list/`

[11] `http://forbesindia.com/lists/2012-celebrity-100/1395/1`

[12] We used Klout score only as a comparative backdrop and is not intended for benchmarking, as the Klout score is a proprietary measure.

Table 3: Average Klout score Comparision

	AAI model	AR model
Average Klout score for Top 20	90	86
Average Klout score for Bottom 20	83	75

Real world Celebrity versus Twitter Celebrities

We analyzed the results of our experiment to find whether real-world celebrities are also Twitter celebrities. We plotted the AAI score versus number of followers in the graph. We expect Twitter celebrities to have high *AAI score* and real-world celebrities to have large number of followers.

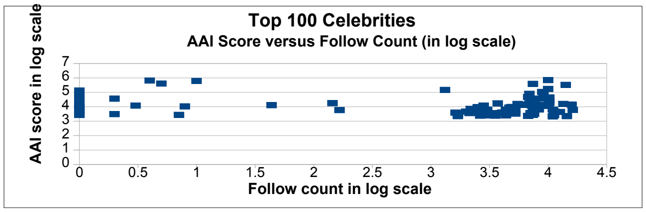


Figure 6: AAI score versus Number of Followers

Figure 6 shows strong co-relation between the *AAI score* and the number of followers. It also shows that a large portion of Twitter celebrities are also real-world celebrities. However, there is also significant portion of Twitter celebrities who do not have a large following. They seemed to have attained celebrity status based on their activities inside Twitter.

Twitter also has another interesting attribute called the "verified" flag. These represent Twitter accounts (typically of celebrities) that are manually verified to belong to the well-known personality. This is by far the most credible benchmark in Twitter for celebrity identification.

We picked the top 100 celebrities from AAI model and AR model and calculated the number of celebrities with the "verified" flag. We found a huge match of 95% of the celebrities in the AAI model as having the "verified" flag. The AR model accounted for a 80% match.

Comparing AAI and AR models

To understand more about the celebrities identified by the AAI model and AR model, we grouped the celebrities into 3 groups

1. Celebrities exclusive to AR model
2. Celebrities common to AR model and AAI model
3. Celebrities exclusive to AAI model

Table 4 shows the celebrities identified under the above mentioned groups. Figure 7 shows the average twitter statistics like number of mentions, number of replies, number of retweets, number of followers for the celebrity groups identified in the Table 4.

Figure 7 shows significant amount of mentions and the followers for the celebrities common to the AAI model and AR model. The celebrities exclusive to the AR model shows more Twitter actions like replies, mentions, retweets on the celebrity where as the celebrities exclusive to AAI model shows more of followers and less of actions when compared to the celebrities exclusive to AR model. Figure 7 hints that the celebrities identified from the AAI model are more of real world celebrities and attracts more people making them well-known, well-liked and well-identified in the community. It also hints that the celebrities identified from the AR model are more of "Twitter celebrities" and are more popular within the Twitter community and attracts more actions within the community.

Table 4: AR model and AAI model celebrities

Celebrity group	Celebrity names
Celebrities exclusive to AR Model	Niall Horan, Liam Payne zaynmalik1D, Xstrology UberFacts, Ariana Grande Funny Tweets, Jai J.D Brooks Mr. Carter
Celebrities common to AR and AAI Model	Justin Bieber, Shah Rukh Khan Amitabh Bachchan, Rihanna Lady Gaga
Celebrities exclusive to AAI Model	Oprah Winfrey, OMG Facts jimmy fallon, Conan O'Brien Ryan Seacrest

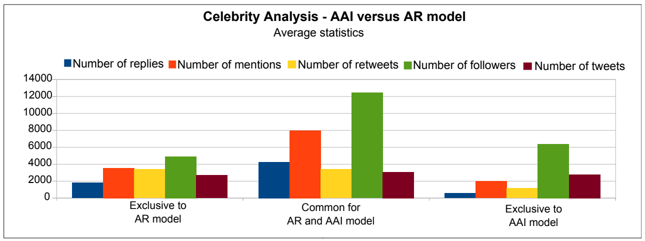


Figure 7: AAI celebrities versus AR celebrities based on Twitter statistics

Figure 8 plots the average action score and reaction score for the celebrity groups identified in Table 4.

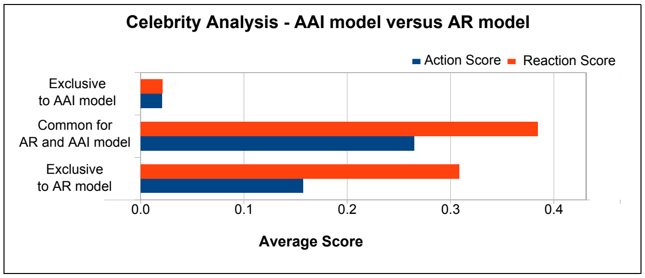


Figure 8: AAI celebrities versus AR celebrities based on AR score

Figure 8 shows that celebrities in general have a higher reaction score. This indicates that popular celebrities get discussed in the Twitter world irrespective of the celebrity tweets in terms of mentions.

Figure 9 plots the average celebrity score for AR model and AAI model for the celebrity groups identified in Table 4.

Figure 9 shows significantly higher AAI score for the celebrities exclusive to AAI model and the celebrities common to AR and AAI model. With respect to AR score, the

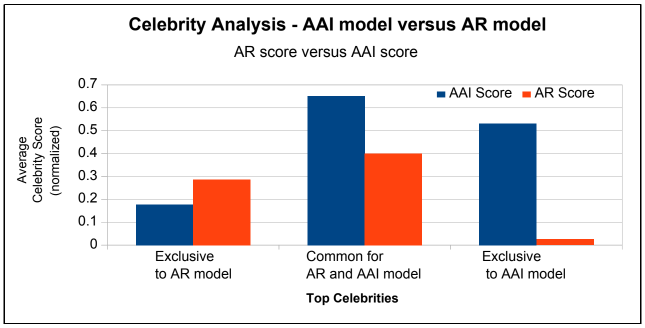


Figure 9: AAI celebrities versus AR celebrities based on celebrity score

celebrities exclusive to AR model and common to both AR and AAI shows significantly higher AR score compared to celebrities exclusive to AAI model.

Based on the observations from Figure 7, Figure 8 and Figure 9, we can understand that though the people identified by the AR model as well as AAI model are celebrities, the characteristics of the celebrities identified by the AR model and AAI model exhibit clear differences.

Celebrities identified by the AAI model show significantly high number of followers and good amount of actions, especially mentions. The celebrities with less action in the AAI model are compensated with their popularity and are well-identified by the people in the community and hence they appear as top celebrities.

Celebrities identified by the AR model show significantly high action (loyalty) scores and good amount of followers. Celebrities with less number of followers in the AR model are compensated with their popularity within the twitter community to generate significant number of actions and hence they appear as top celebrities.

Celebrities identified by the AAI model are more of real-world celebrities and attracts more people making them well-known, well-liked and well-identified in the community. Celebrities identified by the AR model are more of "Twitter celebrities" including non-person accounts like Überfacts that are popular within the Twitter community to generate significant number of actions.

Since both these algorithms seem to measure different signals, in an application setting one can envisage a generic celebrity score:

$$Celebrity(j) = \alpha * AAI(j) + (1 - \alpha) * C(j) \qquad (16)$$

where $0 \leq \alpha \leq 1$. When $\alpha = 0$, the identified celebrities are more of Twitter celebrities and when $\alpha = 1$, the identified celebrities are more of real-world celebrities.

5. CONCLUSIONS AND FUTURE WORK

Celebrity dynamics on social media is an interesting phenomenon, and the proposed algorithms provide promising results to be practically applicable. By providing interpretations for acquaintance, affinity, identification, loyalty and attention from appropriate signals, the proposed algorithms can be easily ported to datasets from other forms of social media. The distinction between AAI and AR distinguishes between celebrities within the social media versus celebrities in the real-world outside.

In the future, we plan to extend this model to identify celebrities in a given domain and apply this model on other forms of user generated content, in addition to social media datasets.

6. REFERENCES

[1] N. Agarwal, H. Liu, L. Tang, and P. S. Yu. Identifying the influential bloggers in a community. In *Proceedings of the international conference on Web search and web data mining*, WSDM '08, pages 207–218, New York, NY, USA, 2008. ACM.

[2] I. Anger and C. Kittl. Measuring influence on twitter. In *Proceedings of the 11th International Conference on Knowledge Management and Knowledge Technologies*, page 31. ACM, 2011.

[3] M. Anjerani and A. Moeini. Selecting influential nodes for detected communities in real-world social networks. In *Electrical Engineering (ICEE), 2011 19th Iranian Conference on*, pages 1 –6, may 2011.

[4] D. J. Boorstin. *The image: A guide to pseudo-events in America*. Vintage, 2012.

[5] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems*, 30(1):107–117, 1998.

[6] M. Cha, H. Haddadi, F. Benevenuto, and K. P. Gummadi. Measuring user influence in twitter: The million follower fallacy. In *4th international aaai conference on weblogs and social media (icwsm)*, volume 14, page 8, 2010.

[7] D. Chen, J. Tang, J. Li, and L. Zhou. Discovering the staring people from social networks. In *Proceedings of the 18th international conference on World wide web*, WWW '09, pages 1219–1220, New York, NY, USA, 2009. ACM.

[8] W. Chen, C. Wang, and Y. Wang. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '10, pages 1029–1038, New York, NY, USA, 2010. ACM.

[9] W. Chen, Y. Wang, and S. Yang. Efficient influence maximization in social networks. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '09, pages 199–208, New York, NY, USA, 2009. ACM.

[10] H. Deng, I. King, and M. R. Lyu. Formal models for expert finding on dblp bibliography data. In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, ICDM '08, pages 163–172, Washington, DC, USA, 2008. IEEE Computer Society.

[11] M. Forestier, J. Velcin, A. Stavrianou, and D. Zighed. Extracting celebrities from online discussions. In *Proceedings of the 2012 International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2012)*, pages 322–326. IEEE Computer Society, 2012.

[12] M. Forestier, J. Velcin, A. Stavrianou, and D. Zighed. Extracting celebrities from online discussions. In *Proceedings of the 2012 International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2012)*, ASONAM '12, pages 322–326,

Washington, DC, USA, 2012. IEEE Computer Society.

[13] H. Friedman and L. Friedman. Endorser effectiveness by product type. *Journal of Advertising Research*, 19(1):63–71, 1979.

[14] R. Ghosh and K. Lerman. Predicting influential users in online social networks. *arXiv preprint arXiv:1005.4882*, 2010.

[15] A. Goyal, F. Bonchi, and L. V. Lakshmanan. Discovering leaders from community actions. In *Proceedings of the 17th ACM conference on Information and knowledge management*, CIKM '08, pages 499–508, New York, NY, USA, 2008. ACM.

[16] B. Hajian and T. White. Modelling influence in a social network: Metrics and evaluation. In *Privacy, security, risk and trust (passat), 2011 ieee third international conference on and 2011 ieee third international conference on social computing (socialcom)*, pages 497–500. IEEE, 2011.

[17] D. Hatcher, G. S. Bawa, and S. Barry de Ville. How you can identify influencers in sas® social media analysis (and why it matters).

[18] C. I. Hovland, I. L. Janis, and H. H. Kelley. *Communication and persuasion: Psychological studies of opinion change.* Yale University Press New Haven, CT, 1953.

[19] H. Kwak, C. Lee, H. Park, and S. Moon. What is twitter, a social network or a news media? In *Proceedings of the 19th international conference on World wide web*, pages 591–600. ACM, 2010.

[20] C. Li, S. Lin, and M. Shan. Finding influential mediators in social networks. In *Proceedings of the 20th international conference companion on World wide web*, pages 75–76. ACM, 2011.

[21] Z. Luo, M. Osborne, J. Tang, and T. Wang. Who will retweet me?: finding retweeters in twitter. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, pages 869–872. ACM, 2013.

[22] G. McCracken. Who is the celebrity endorser? cultural foundations of the endorsement process. *Journal of Consumer research*, pages 310–321, 1989.

[23] W. J. McGuire. The nature of attitudes and attitude change. *The handbook of social psychology*, 3:136–314, 1969.

[24] C. W. Mills. The power elite [1956]. *New York*, 1981.

[25] H.-K. Peng, J. Zhu, D. Piao, R. Yan, and Y. Zhang. Retweet modeling using conditional random fields. In *Data Mining Workshops (ICDMW), 2011 IEEE 11th International Conference on*, pages 336–343. IEEE, 2011.

[26] M. S. Srinivasan, S. Srinivasa, and S. Thulasidasan. Exploring celebrity dynamics on twitter. In *Proceedings of the 5th IBM Collaborative Academia Research Exchange Workshop*, I-CARE '13, pages 13:1–13:4, New York, NY, USA, 2013. ACM.

[27] G. Turner. *Understanding Celebrity.* SAGE Publications, 2004.

[28] Y. Wang, G. Cong, G. Song, and K. Xie. Community-based greedy algorithm for mining top-k influential nodes in mobile social networks. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1039–1048. ACM, 2010.

[29] J. Weng, E.-P. Lim, J. Jiang, and Q. He. Twitterrank: finding topic-sensitive influential twitterers. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 261–270. ACM, 2010.

[30] J. Zeng, W. K. Cheung, C. hung Li, and J. Liu. Coauthor network topic models with application to expert finding. *Web Intelligence and Intelligent Agent Technology, IEEE/WIC/ACM International Conference on*, 1:366–373, 2010.

[31] L. Zhou. Trust based recommendation system with social network analysis. In *Information Engineering and Computer Science, 2009. ICIECS 2009. International Conference on*, pages 1 –4, dec. 2009.

SLEMAS: An Approach for Selecting Materialized Views Under Query Scheduling Constraints

Ahcene Boukorca
LIAS/ISAE-ENSMA - Poitiers University
Futuroscope, 86960, France
France
boukorca@ensma.fr

Ladjel Bellatreche
LIAS/ISAE-ENSMA - Poitiers University
Futuroscope, 86960
France
bellatreche@ensma.fr

Alfredo Cuzzocrea
ICAR-CNR & University Calabria
Rende (CS)
Italy
cuzzocrea@si.deis.unical.it

ABSTRACT

Materialized views are one of the most popular optimization techniques selected during the physical phase to speed up query processing in traditional and advanced databases. Their selection has been proven to be NP-hard. As a consequence large panoply of heuristics has been proposed to find near optimal solutions. Usually, the selected materialized views are whole life disk resident and their presence is not calling into question. Note that view maintenance can cause significant amounts of CPU and I/O usage, which can be detrimental to performance in a write-intensive database application. Typically materialized views are stored on disk; however with big number of queries, there are situations where not all a good candidate views will be selected. As a consequence, their dynamic selection becomes a necessity. In this paper, we address the problem of materialized view selection by considering the query scheduling. We first review the most important existing work on static and dynamic view selection. A formalization of the problem of view selection considering the re-ordering of a large number of queries is given. A system, called SLEMAS, playing the role of a generic advisor is described. Finally, intensive experiments are conducted to compare the efficiency of our system regarding the most important state of art algorithms.

1. INTRODUCTION

Nowadays, the complex OLAP queries involving joins and aggregations is part of the scenery of applications requiring extremely large databases such as data warehouses, scientific and statistical databases. Materialized views ($\mathcal{MV}$) are one of the most popular optimization structures used in several databases deployment platforms: centralized [10], distributed [4], Cloud [12]. $\mathcal{MV}$ are used to pre-compute and store aggregated data such as sum of extremely large tables such as the fact table of a given data warehouse schema. So, $\mathcal{MV}$ are suitable for queries with expensive joins or aggregations. Once selected, all queries will be rewritten using $\mathcal{MV}$[1] in order to avoid irrelevant base table accesses. A rewriting of a query q_i of a given workload using views is a query expression q_i' referencing to these views. The query rewriting is done *transparently* by query optimizer. To generate a best rewriting for a given query, a cost-based selection method is used [2].

Note that $\mathcal{MV}$ store data from base tables. In order to keep the $\mathcal{MV}$ in the database repository up to date, it is necessary to maintain them in response to the changes at the base tables. This process of updating views is called *view maintenance* which has evoked great interest in the past years [8]. A $\mathcal{MV}$ can be either recomputed from scratch, or incrementally maintained by propagating the base data changes onto that view. Note that re-computing the views can be prohibitively expensive.

Due to resources required for $\mathcal{MV}$ (disk space, computation time, maintenance overhead and cost required for query rewriting process), *DBA* cannot materialize all possible views [11]. Hence, she needs to select an appropriate set of views to materialize under some resource constraints. Historically, the Views Selection Problem ($\mathcal{VSP}$) has been formalized as follows [10]: given a set of most frequently used queries $\mathcal{Q} = \{q_1, q_2, ..., q_n\}$, where each query q_i has an access frequency f_i ($1 \leq i \leq n$) and a set of resource constraints $\mathcal{CS} = \{C_1, ..., C_k\}$. The $\mathcal{VSP}$ consists in selecting a set of $\mathcal{MV}$ that minimizes one or more objectives, possibly subject to one or more constraints. Many variants of this problem have been studied that concern the studied objective functions and the resource constraint(s). The main popular variations use respectively the following objective function and constraints: (i) *minimizing the query processing cost subject to a storage size constraint* [10], (ii) *minimizing query and maintenance costs subject to storage space constraint* [15], (iii) *minimizing query cost under a maintenance constraint* [10]. This problem is known as an NP-hard problem [10]. Several algorithms were proposed to deal with this problem. In [17], a classification of existing algorithms is given. Note that commercial and academic DBMS propose tools (advisors) (e.g., DB2 design Advisor, SQL access Advisor for SQL, Data Tuning advisor for SQL server, PARINDA for Postgres [16]) to recommend $\mathcal{MV}$ to DBA.

The most traditional formalization selects views in a static where it assumes that the queries is *a priori known and pre-*

The 20th International Conference on Management of Data (COMAD), 17th-19th Dec 2014 at Hyderabad, India.

---

[1] This process is known as *query rewriting*

ordered. To relax this hypothesis, the dynamic selection has been proposed [14]. Existing algorithms for dynamic selection of $\mathcal{MV}$ are divided into two categories based on their incoming workload [6]: algorithms with a predefined workload (the algorithm described in [21] belongs to this category) and algorithms with an unknown workload (DynaMat system is an example of this category [14]). Typically selected $\mathcal{MV}$ are stored on disk permanently with updating strategy on these $\mathcal{MV}$; however with disk space limit and with big number of queries to be optimized, there are situations where not all a good candidate views will be selected. As a consequence, a flexible selection of the views under space constraint is recommended. It is characterized by temporary presence of the views on the disk to maximize the benefit of using all best views . In this mode of views selection, the views are created as a workload execute. If a query need a view that is not created, the view will be materialized on demand. If there is not enough space, existing views may be dropped following LRU rules. The views selection manner called dynamic materialization. To maximize the benefit of using materialized views before their dropping, it is important to permit workload's query order. this query order permutation called query scheduling.

Most important studies related to the dynamic selection ignore the query scheduling, except the Phan et al.'s [21] and Diwan et al.'s [7] works. This ignorance may penalize the performance of the selected $\mathcal{MV}$. Note that the query scheduling defines an efficient order to evaluate a set of queries to take benefit from current content of a storage device before relevant data is evicted. The device may be a main memory buffer, a secondary memory device such as hard disk, flash, etc. A couple of existing studies showed the impact of query scheduling on managing buffer where the allocation database objects into the buffer is guided by the order of the queries [9]. Recent research efforts study the impact of query scheduling on selecting optimization structures such as $\mathcal{MV}$, indexes [21] and horizontal data partitioning schema [1]. To the best of our knowledge the work of Phan et al. [21] is the sole that dynamically selects $\mathcal{MV}$ by considering query re-ordering. Consequently, we propose to details the architecture of the proposed system. It is mainly composed of four components: Dynamic Materialized Query Table (MQT) Scheduler (DMS), MQT candidate generator, scheduled queries generator, and Dynamic MQT management module, which will be described below:

- The DMS receives query workload.
- MQT candidate generator: the MDS sends the queries to MQT advisor of DB2 that returns a set of candidate MQT and their associated indexes. At the highest level, the DB2 Advisor works as a black-box view-index recommendation engine. The black-box has two inputs: a set of SQL statements known as the workload, and statistics describing the target database. There is only one output: the recommended views and indexes.
- Scheduled queries generator: the candidate MQT and indexes are then submitted to DMS that runs a genetic algorithm to find the best query order that produces the highest MQT benefit. The optimal solution requires the exploration of a search space of $N!$ permutation of the query workload. The objective of the genetic algorithm is establish a tradeoff between maximizing MQT cache hits, minimizing MQT materialization and minimizing base table accesses.
- Dynamic MQT management module is core of the approach. It uses a probabilistic model defining the usage and the materialization of any MQT candidate. The MQT pool (called cache) is managed by LRU that provides non-zero hit probability to the entire candidate MQT.

The main limitation of Phan et al.'s work: (i) it is DB2 DBMS dependent, (ii) certainly DB2 advisor takes into account the interaction among optimization structures (indexes, $\mathcal{MV}$, partitioning, and clustering) [27], but the interaction between queries is not well highlighted. Another aspect related to the query workload is the small number of the used queries. Nowadays, the number of queries may be very large (era of big queries). Note that the interaction between queries is crucial for selecting $\mathcal{MV}$ in different database deployment platforms [18, 19, 26], and (iii) having a genetic algorithm with fitness function using probabilistic parameters that require a predictable approach to get them is time consuming. To overcome these limitations, we propose a new scalable approach; called SLEMAS. This system plays the role of generic advisor.

The paper is organized as follows: Section 2 presents the Background. Section 3 details our contributions, where we present the scalable algorithm to capture the query interaction, $\mathcal{MV}$ selection algorithm, query scheduling and materialization strategy. Section 4 implements our results and validation in Oracle 11g. Section 5 concludes the paper by summarizing the main results and suggesting future work.

2. BACKGROUND

In this section, we present some concepts to facilitate the understanding of our approach that considers the interaction of queries to generate the view candidates.

2.1 Multi-Query Optimization

The Multi-query optimization (MQO) problem has well studied since 1980s [24, 22]. MQO tries to perform a batch of queries by exploiting some share common results. MQO problem can be divided into two phases [22]. The first phase is to prepare alternative plans for each query (Each query may be represented by an algebraic tree corresponding to its execution plan). These plans can identify common shared results among a set of queries. The second phase is to select exactly one plan for each query. As a result of second phase, the query plans may be merged to generate a graph structure called, *unified query plan* (UQP). The structure of this plan is similar to that proposed in [26]. The leaf nodes of the UQP represent the base tables. The root nodes represent the final query results and the intermediate nodes represent the common sub-expressions shared by the queries. Among intermediate nodes, we distinguish: (i) unary nodes representing the selection and the projection operations (ii) binary nodes representing join, union, intersection, etc. Figure 3a. presents an UPQ for a workload of 10 queries issue form star schema benchmark. Several unified query plans may exist for a given query workload due to the properties of relational algebra operations [26]. Sellis et al. [24] are proposed an $A*$ search to explore all possible UQP. But

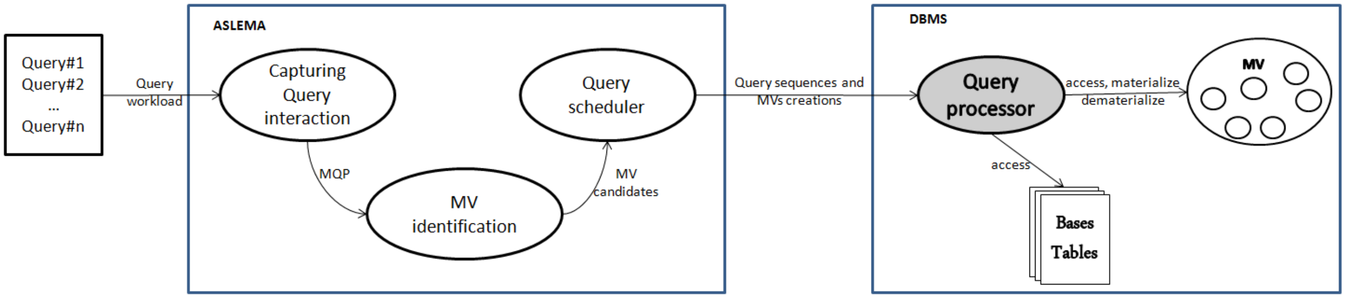


Figure 1: A global overview of our approach

exploring all possible UQP has been formulated as an NP-hard problem [24]. To reduce the number of exploring UQP many cost estimation functions are proposed [25]. Roy et al. [23] describe a greedy heuristic for generating a good alternative plans which maximize shared results. Dalvi et al. [5] extend the work of Roy et al.[23] by pipelining common shared results to maximize the benefit. With Big Queries Era, offering scalable algorithms for generating a best query plan becomes a crucial challenge [3].

2.2 Problem Formulation

Before formalizing the $\mathcal{MVP}$ considering the Query Scheduling Problem ($\mathcal{QSP}$), we think it would be wiser to propose a separate formalization of both $\mathcal{MVP}$ and $\mathcal{QSP}$. *Inputs:* (i) a relational data warehouse ($\mathcal{RDW}$), (ii) a workload with a set of queries represented by an UQP, (iii) a set of intermediate nodes candidates for materialization ; *Constraint:* a storage constraint $\mathcal{S}$; *Output:* a set of $\mathcal{MV}$ optimize the cost of processing $\mathcal{Q}$ and satisfying the constraint cost ($size(\mathcal{MV}) \leq \mathcal{S}$).

Similarly, the $\mathcal{QSP}$ is formalized as follows:
Inputs: (i) $\mathcal{RDW}$, (ii) a workload with a set of queries represented by an UQP, (iii) a set of intermediate nodes candidates for materialization; a disk allocation policy;
Output: scheduled queries of the workload into a new ordered set that the query having the least execution cost.

The $\mathcal{MVP}$ considering $\mathcal{QSP}$ takes (i) A $\mathcal{RDW}$ and (ii) a set of queries, (iii) a set of intermediate nodes candidates for materialization; a constraint representing the limited storage size. The problem aims at providing: (i) a scheduled set of queries and (ii) $\mathcal{MV}$, minimizing the overall processing cost of and satisfying the storage constraint. The search space of the combination of $\mathcal{MVP}$ and $\mathcal{QSP}$ becomes very huge [21]. In the next section, we propose a new approach supported by a tool (SLEMAS) dealing with this problem.

3. THE SLEMAS APPROACH

SLEMAS is a three-tier architecture as shown in the Figure 1: (1) an application tier representing the query workload, (2) SLEMAS with three main roles: (i) capturing of interaction among queries, (ii) generation of views candidate and (iii) query scheduling, and (3) a data storage tier that implements solutions recommended by SLEMAS. Its components are below detailed.

3.1 The Application Tier

This module receives from users a set of queries in a queue to be processed by a DBMS.

3.2 SLEMAS Tier

Once query workload is received, SLEMAS performs the following tasks:

3.2.1 Capturing of interaction among queries:

This step is performed by constructing the UQP. This construction has to take into account the Era of Big Queries. Usually, the interaction of queries is captured by the use of acyclic graph [26] (Figure 2.a. To respond to the context of Big Queries, we propose the use of hypergraph data structure that showed their efficiency in *Electronic Design Automation* (EDA) [13]. In our context the vertices and edges of the hypergraph represent respectively the join nodes and queries as shown in Figure 2-a. Since the number of nodes of our hypergraph can be very large, we adopt the same philosophy of EDA, where the hypergraph is partitioned according the interaction among queries. More precisely, each partition contains high interacted queries. To do the partitioning we use a tool widely used in EDA called *HMETIS* [13]. Figure 2-b gives an example of a such partitioning. Each partition is then transformed in a acyclic directed graph which is similar to traditional query acyclic graph. During this transformation, a cost driven approach to order the nodes is given [3]. This cost computes the processing cost of overall queries (Figure 2-c).

3.2.2 Materialized Views Selection:

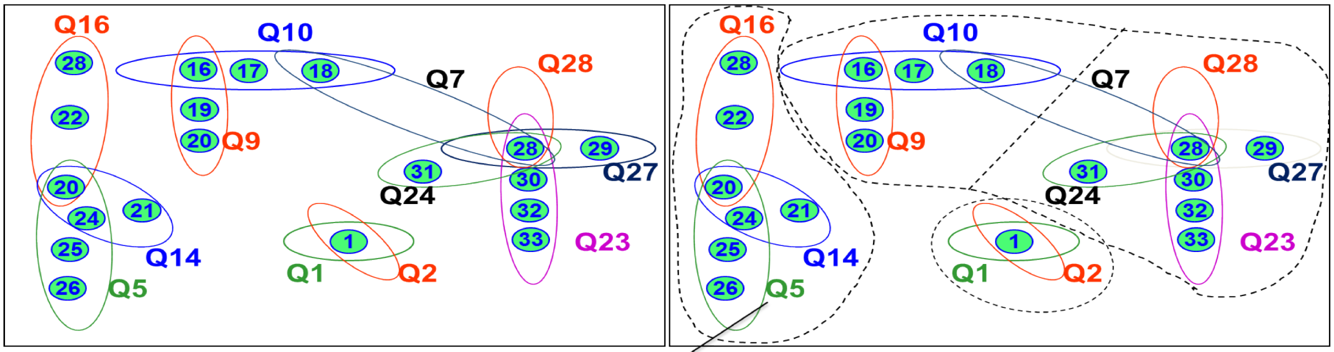
Note that all nodes of the global plan are candidate for materialization which may represent a huge number. For instance, in our experiments, we consider 1 000 queries involving 1552 join nodes. As a consequence a pruning mechanism is needed. It shall take into account the benefit of the nodes and their constraints related to their storage and maintenance. To do so, we define some functions:

- $cost_{WO}(q_i, \Phi)$: the processing cost of the query q_i without view(s).
- $cost_{WV}(q_i, V_j)$: the query processing cost of query q_i using the materialized view V_j.
- $cost_{Mat}(V_j)$: the maintenance cost of the view V_j.
- $Size(V_j)$: the cost needed to store the view V_j.

We define the benefit of a given view V_j (denoted by $Benefit(V_j)$) by:

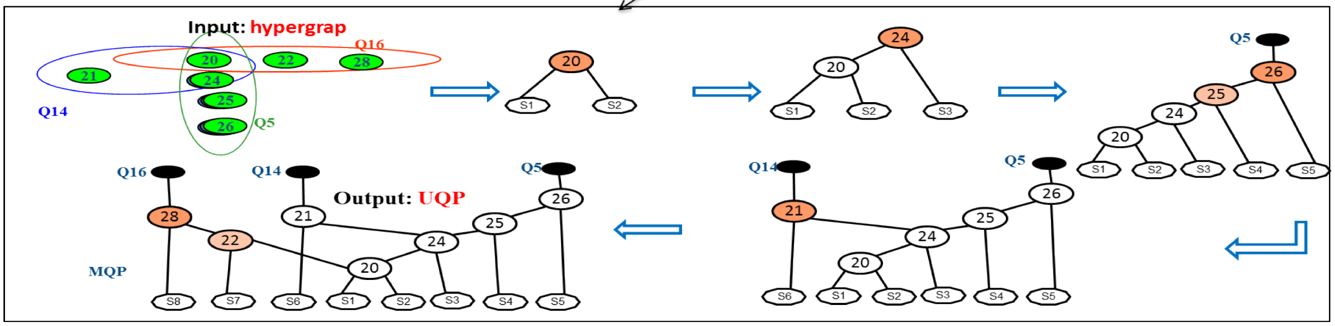
$$benefit(V_j) = cost_{WO}(V_j) - cost_{WV}(V_j) - cost_{Mat}(V_j) \quad (1)$$

where $cost_{WO}(V_j)$ and $cost_{WV}(V_j)$ represent respectively the total processing cost of queries without/with the view



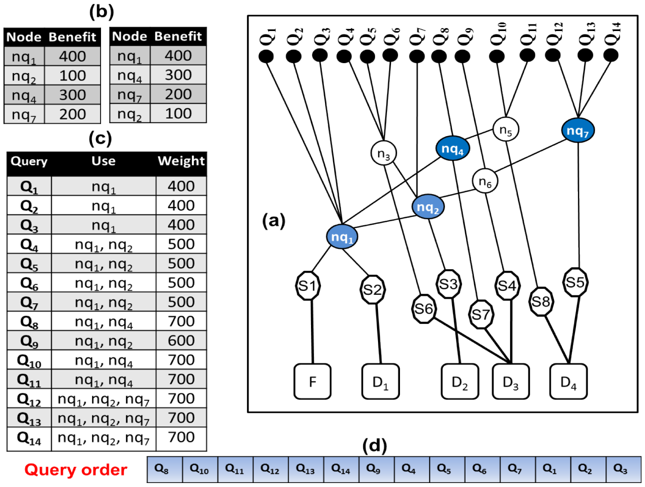
(a) Hypergraph of join nodes

(b) Hypergraph partitioning results



(c) Transformation of hypergraph to UQP

Figure 2: An Example of Query Interaction Capturing



(b)

Node	Benefit
nq_1	400
nq_2	100
nq_4	300
nq_7	200

Node	Benefit
nq_1	400
nq_4	300
nq_7	200
nq_2	100

(c)

Query	Use	Weight
Q_1	nq_1	400
Q_2	nq_1	400
Q_3	nq_1	400
Q_4	nq_1, nq_2	500
Q_5	nq_1, nq_2	500
Q_6	nq_1, nq_2	500
Q_7	nq_1, nq_2	500
Q_8	nq_1, nq_4	700
Q_9	nq_1, nq_2	600
Q_{10}	nq_1, nq_4	700
Q_{11}	nq_1, nq_4	700
Q_{12}	nq_1, nq_2, nq_7	700
Q_{13}	nq_1, nq_2, nq_7	700
Q_{14}	nq_1, nq_2, nq_7	700

(d)

Query order	Q_8	Q_{10}	Q_{11}	Q_{12}	Q_{13}	Q_{14}	Q_9	Q_4	Q_5	Q_6	Q_7	Q_1	Q_2	Q_3

Figure 3: Example of our Scheduling Approach

V_j. Instead of treating the whole search space including all candidates as in the usual approaches for materializing views, we propose to use a *divide-conquer approach*, where the search space is divided into several sub search spaces, where each one corresponds to a connected component of the global graph (Figure 1). Contrary to the existing studies where they allocate the whole storage constraint to all views candidate, our approach allocates this storage to each component to be fair. Then, each component C_k is processed individually, where its nodes are sorted according their benefits and their ability to satisfy the storage constraint. Three selection cases are possible. Let N^{C_k} be the set of nodes of C_k having a greater benefit. The top nodes satisfying the storage constraint are selected.

3.2.3 *Query scheduling:*

To avoid massively view dropping, we schedule queries. This is done by respecting the following principle: *when a view is materialized, it should optimize the maximum of queries before its dropping.* Therefore, we propose the following procedure supported by an example in which we consider a connected component with 14 queries (Figure 3-a).

1. The queries are grouped in many distinct components which the interaction (sharing of intermediate results), is important between queries inside component and negligible interaction between components. Our scheduler aims to schedule the queries in each component and the order between components.
2. The scheduler is based on maximizing the benefit of reusing nodes, so the order is guided by nodes. The materialization of a node prompts that the queries which use this node are the following to be run. So, no consideration of space constraint in query scheduler.
3. The identification of node(s) of each component with maximal benefit (called queen nodes). In our example, four *queen nodes* are selected: $\{qn_1, qn_2, qn_3, qn_4\}$ (represented by solid nodes in Figure 3-a).
4. Ordering queen nodes: Let N^{C_k} be the number of

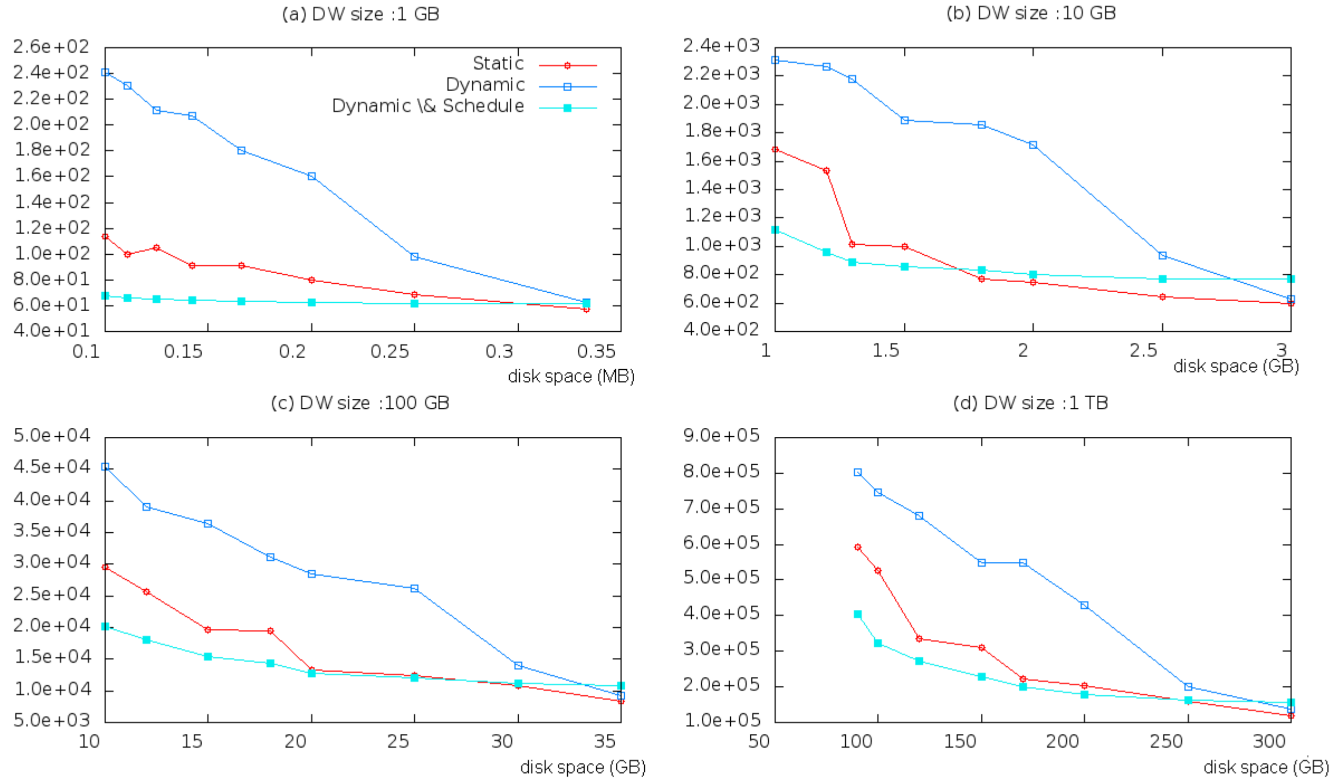


Figure 4: Advantage of dynamic materialization with query scheduling

nodes of the component C_k. Their ordering is based on their benefit. The benefit of the queen nodes are propagated to their queries. As a consequence, each query may be assigned to a weight representing the sum of the benefit of its nodes. These weight are then used to schedule the queries based on their overall benefit (Figure 3-d). The query scheduling process not consider the constraint space

Till now, materialized views candidate are identified and the order of queries. Based on the different cost models available at SLEMAS side, it can easily decide on materializing or dematerializing views by performing simulation using algorithm 1.

Algorithm 1 materializeView (mv)

```
$cost \leftarrow estimateMaintenance(mv)$;{Estimate the maintenance cost of the view using the generic cost model}
changeStat($mv$, true);{ change the stat of the view as materialized}
$diskSpace \leftarrow diskSpace - sizeOf(mv)$;
return $cost$;
```

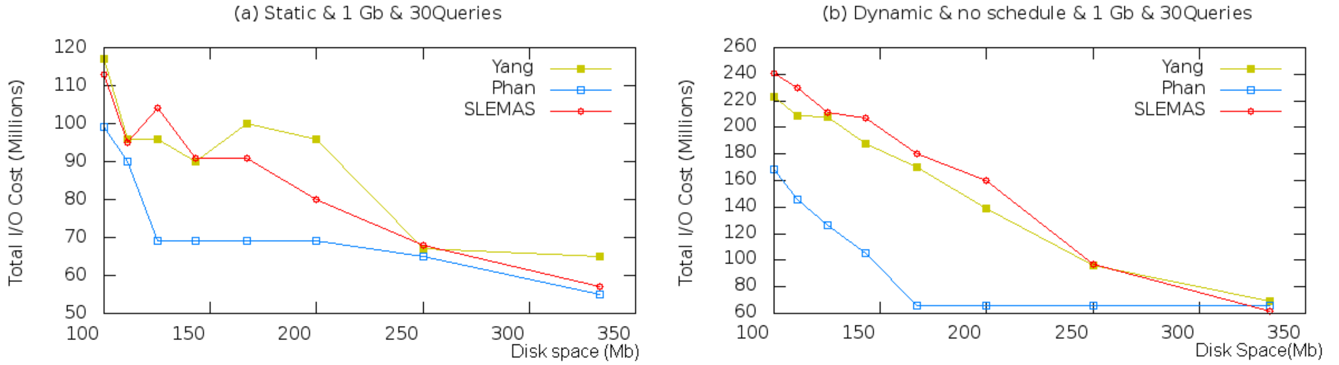
3.3 Data Storage Tier

The results obtained by SLEMAS are translated respecting the target DBMS then receives a creation and dropping scripts of materialized views and the pre-ordered queries.

4. EXPERIMENTAL EVALUATION AND ANALYSIS

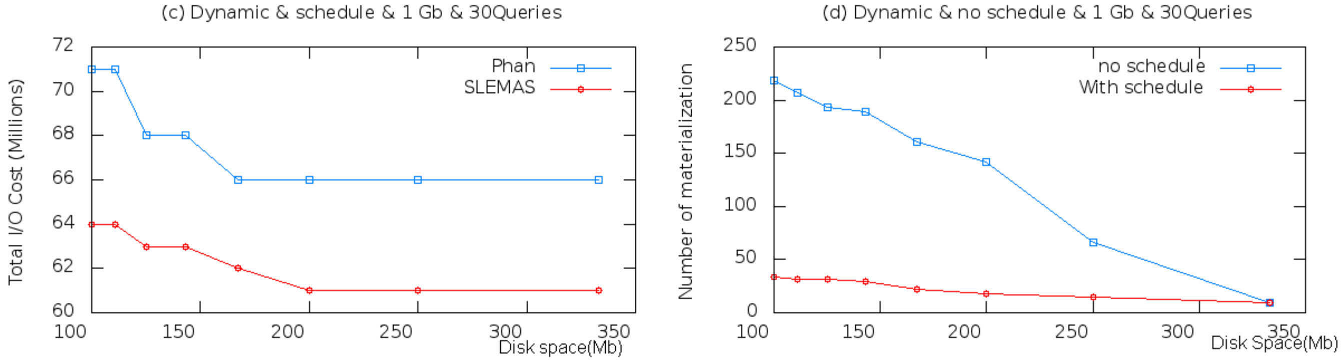
We conduct several experiments to evaluate the efficiency of SLEMAS. First of all, we develop a simulation tool using Java. It allows to get automatically the characteristics of the meta data of the data warehouse and run the three algorithms: (1) SLEMAS's approach; (2) dynamic materialization algorithm proposed by Phan et al's [21] and (3) $\mathcal{MV}$ selection algorithm proposed by Yang et al's [26] using a static formalization. To analyze the behaviors of these algorithms, we consider four scenarios: (1) using static materialization, (2) dynamic materialization, (3) with considering query scheduling and (4) without query scheduling. These scenarios are tested by varying: (a) the size of data warehouse, (b) consideration of different query workload's randomly generated using SSB query generator and (c) varying the storage space constraint. The simulated results are then deployed on Oracle 11g *DBMS*, running on a Core 2 Duo server with 2.40GHZ CPU and 32 GB of main memory. The star Schema Benchmark (SSB) with 100 GB of data containing a fact table *Lineorder* and 4 dimension tables [20]: *Part*, *Customer*, *Supplier* and *Date* is used.

In the first experiments, we test the interest of dynamic and query scheduling on optimizing queries. To perform our experiments we consider a data warehouse with different sizes (1Gb, 10Gb, 100Gb and 1 Tb) and a workload of 30 queries, the candidate nodes are selected using our approach. Three scenarios are considered: (i) naive scenario in which nodes are materialized till the saturation of storage space,



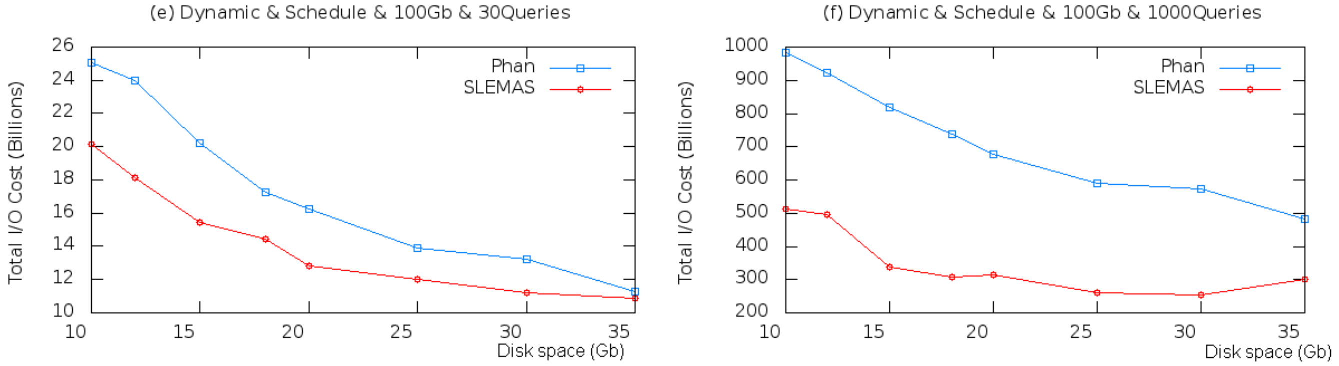
(a) Static & 1 Gb & 30Queries

(b) Dynamic & no schedule & 1 Gb & 30Queries



(c) Dynamic & schedule & 1 Gb & 30Queries

(d) Dynamic & no schedule & 1 Gb & 30Queries



(e) Dynamic & Schedule & 100Gb & 30Queries

(f) Dynamic & Schedule & 100Gb & 1000Queries

Figure 5: Performance of SLEMAS's approach

(ii) materializing without scheduling using our dynamic approach and considering the workload as pre-ordered[2] and (iii) materializing with query scheduling. The overall cost of queries in terms of inputs/outputs is then computed by varying the storage constraint. Figure 4 summarizes the obtained results. The main lesson is: the dynamic materialization with query scheduling outperforms the other scenarios whatever the size of the data warehouse. This shows the interest of incorporating the query scheduling in materializing views.

In the second experiments: we test the performance of our approach compared with two existing approaches: Phan et al.'s method [21], and Yang et al.'s method [26]. For Phan method, we have developed the following algorithms: (i) a genetic algorithm to find an optimal query permutation by emulating Darwinian natural selection of 1000 generations; (ii) algorithm to select candidate nodes which are the nodes that have greater benefit have been selected as candidates (In Phan et al.'s, they are used DB2 advisor to have those all nodes with greater benefit); (iii) pruning algorithm of candidate views using their benefit; (iv) an evaluation algorithm to estimate the total net benefit of using pruned candidate views set by query workload; (v) algorithm to manage the cache of the views (LRU). For Yang, we have developed the following functions: (i) generation of individual plan tree, (ii) generating Multiple Views Processing Plans (*MVPP*), using merging individual plans (iii) selecting materialized views using *MVPP* (iv) estimation of query MVPP using views. To show the performance of our approach three scenarios are considered:

1. **Static materialization:** we consider a data warehouse with 1Gb and a workload of 30 queries, the nodes selected by each algorithms are materialized till the saturation of storage space. As shown in the Fig-

---

[2] The query scheduling module of our approach in this case is obsolete.

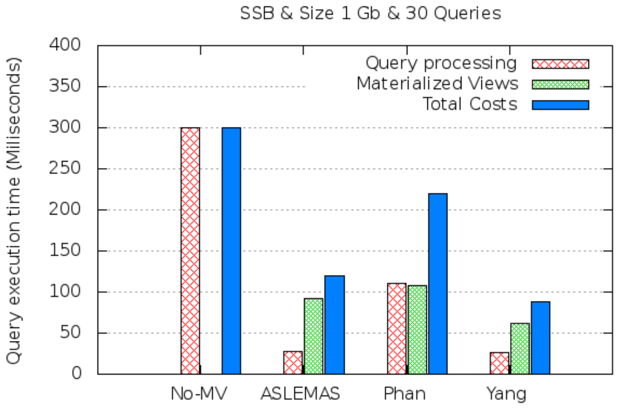


Figure 6: Oracle validation on data warehouse of 1Gb

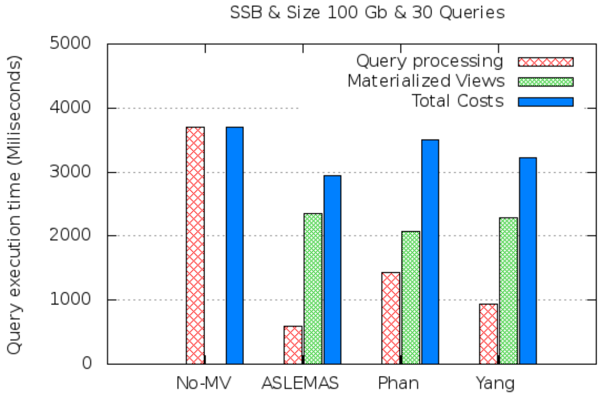


Figure 7: Oracle validation on data warehouse of 100Gb

ure 5-a, there is not a big difference between the three methods, which improve that our approach not avoids the selection of best views.

2. **Dynamic materialization without query scheduling**, we have used the same configuration as static materialization. As shown in the Figure 5-b, we show that Phan is more better than our approach. This is due to the difference of materialization/dematerialization number (Figure 5-c). Phan et al.'s method tries to find best candidate views that optimize the workload globally which minimize the dropping of the views. But the views in our approach a divided a many sub sets which each sub-set optimize some queries, which increase the probability of dropping views if the query not scheduled.
3. **Dynamic materialization with query scheduling:** we have used data warehouse with different size (1Gb and 100Gb) and a workload of 30 queries. As shown in the Figures 5-c and 5-e, our approach outperforms Phan method because the number dropping is minimal and each materialized views are used maximally to optimize the queries of the component. As shown in the Figures 5-f, our approach performs more when we use big queries (in our tests: 1000 queries).

4.1 Validation in Oracle 11g

ue to the complexity and time needed to deploy all theoretical solutions on Oracle DBMS, we propose the following: we consider a workload of 30 queries running on two data warehouses (1GB and 100GB). The disk storage is set to 400MB (39%) for the first dataset(Figure 6) and 30GB (30%) for the second dataset (Figure 7).

The obtained results described in the Figures 6 and 7 which are quite similar to those obtained by our simulator. This shows the quality of our used cost models.

5. CONCLUSION

In this paper, we addressed an important problem which is the dynamic materialized view selection by considering the query scheduling. Both problems are hard. To solve this problem, we propose a methodology supported by a generic advisor called SLEMAS. It is 3-tiers architecture included: (1) an application tier representing the query workload, (2) SLEMAS with three main components: (i) capturing of interaction among queries. To offer a scalable algorithm, we proposed the use of hypergraph largely used in the EDA domain. (ii) Generation of views candidate performed by divide-conquer approach, in which the hypergraph is partitioned into several connected components. (iii) Query scheduling allows re-ordering the query workload based on their benefit in using materialized views. (3) A data storage tier that implements solutions recommended by SLEMAS. Our approach is compared against the most important state of art works and the obtained results show the efficiency and effectiveness of our approach.

Currently, we are working in two directions: the incorporation of another optimization structure which is the indexes to SLEMAS and development of a more sophisticated tool with nice interfaces and offering several API to connect different DBMS platforms.

6. REFERENCES

[1] L. Bellatreche, A. Kerkad, S. Breß, and D. Geniet. Roupar: Routinely and mixed query-driven approach for data partitioning. In *On the Move to Meaningful Internet Systems: OTM 2013 Conferences*, pages 309–326. Springer, 2013.

[2] A. G. Bello, K. Dias, A. Downing, J. Feenan, J. Finnerty, W. D. Norcott, H. Sun, A. Witkowski, and M. Ziauddin. Materialized views in oracle. In *VLDB*, pages 659–664, 1998.

[3] A. Boukorca, L. Bellatreche, S.-A. B. Senouci, and Z. Faget. Sonic: Scalable multi-query optimization through integrated circuits. In *Database and Expert Systems Applications*, pages 278–292. Springer, 2013.

[4] L. W. F. Chaves, E. Buchmann, F. Hueske, and K. Böhm. Towards materialized view selection for distributed databases. In *EDBT*, pages 1088–1099, 2009.

[5] N. N. Dalvi, S. K. Sanghai, P. Roy, and S. Sudarshan. Pipelining in multi-query optimization. *Journal of Computer and System Sciences*, 66(4):728–762, 2003.

[6] N. Daneshpour and A. A. Barforoush. Dynamic view management system for query prediction to view

materialization. *IJDWM*, 7(2):67–96, 2011.

[7] A. Diwan, S. Sudarshan, and D. Thomas. Scheduling and caching in multi-query optimization. In *International Conference on Management of Data COMAD, Delhi, India*, 2006.

[8] A. Gupta and I. S. Mumick. Maintenance of materialized views: Problems, techniques, and applications. *IEEE Data Eng. Bull.*, 18(2):3–18, 1995.

[9] A. Gupta, S. Sudarshan, and S. Viswanathan. Query scheduling in multi query optimization. In *IDEAS*, pages 11–19, 2001.

[10] H. Gupta. *Selection and maintenance of views in a data warehouse.* Ph.d thesis, Stanford University - USA., 1999.

[11] V. Harinarayan, A. Rajaraman, and J. D. Ullman. Implementing data cubes efficiently. In *SIGMOD*, pages 205–216, 1996.

[12] V. Kantere, D. Dash, G. François, S. Kyriakopoulou, and A. Ailamaki. Optimal service pricing for a cloud cache. *IEEE TKDE*, 23(9):1345–1358, 2011.

[13] G. Karypis and V. Kumar. Multilevel k-way hypergraph partitioning. In *ACM/IEEE Design Automation Conference (DAC)*, pages 343–348, New York, NY, USA, 1999. ACM.

[14] Y. Kotidis and N. Roussopoulos. Dynamat: a dynamic view management system for data warehouses. *SIGMOD Rec.*, 28(2):371–382, June 1999.

[15] M. Lawrence. Multiobjective genetic algorithms for materialized view selection in olap data warehouses. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 699–706. ACM, 2006.

[16] C. Maier, D. Dash, I. Alagiannis, A. Ailamaki, and T. Heinis. Parinda: an interactive physical designer for postgresql. In *EDBT*, pages 701–704, 2010.

[17] I. Mami and Z. Bellahsene. A survey of view selection methods. *SIGMOD Record*, 41(1):20–29, 2012.

[18] H. Mistry, P. Roy, S. Sudarshan, and K. Ramamritham. Materialized view selection and maintenance using multi-query optimization. In *SIGMOD*, pages 307–318, 2001.

[19] T. Nykiel, M. Potamias, C. Mishra, G. Kollios, and N. Koudas. Mrshare: Sharing across multiple queries in mapreduce. *PVLDB*, 3(1):494–505, 2010.

[20] X. C. Pat O'Neil, Betty O'Neil. Star schema benchmark. June 2009.

[21] T. Phan and W.-S. Li. Dynamic materialization of query views for data warehouse workloads. In *ICDE*, pages 436–445, 2008.

[22] A. Rosenthal and U. S. Chakravarthy. Anatomy of a mudular multiple query optimizer. In *Proceedings of the International Conference on Very Large Databases (VLDB)*, pages 230–239, 2006.

[23] P. Roy, S. Seshadri, S. Sudarshan, and S. Bhobe. Efficient and extensible algorithms for multi query optimization. *ACM SIGMOD Record*, 29(2):249–260, 2000.

[24] T. K. Sellis. Multiple-query optimization. *ACM Transactions on Database Systems*, 13(1):23–52, March 1988.

[25] K. Shim, T. Sellis, and D. Nau. Improvements on a heuristic algorithm for multiple-query optimization. *Data & Knowledge Engineering*, 12(2):197–222, 1994.

[26] J. Yang, K. Karlapalem, and Q. Li. Algorithms for materialized view design in data warehousing environment. In *VLDB*, pages 136–145, 1997.

[27] D. C. Zilio, J. Rao, S. Lightstone, G. M. Lohman, A. J. Storm, C. Garcia-Arellano, and S. Fadden. Db2 design advisor: Integrated automatic physical database design. In *VLDB*, pages 1087–1097, 2004.

INDUSTRY
PAPERS

Problem Identification by Mining Trouble Tickets

Vikrant Shimpi, Maitreya Natu, Vaishali Sadaphal, Vaishali Kulkarni
Tata Research Development and Design Centre
{vikrant.shimpi, maitreya.natu, vaishali.sadaphal, vaishali.kulkarni}@tcs.com

ABSTRACT

IT systems of today's enterprises are continuously monitored and managed by a team of resolvers. Any problem in the system is reported in the form of trouble-tickets. A ticket contains various details of the observed problem. However, the knowledge of the actual problem is hidden in the ticket description along with other information. Knowledge of issues helps the service providers to better plan to improve cost and quality of operations. In this paper, we address the problem of extracting the issues from ticket descriptions. We discuss various challenges in issue extraction and present algorithms to handle different scenarios. We demonstrate the effectiveness of the proposed algorithms through two real-world case-studies.

1. INTRODUCTION

With the increasing reliance of business on IT, the health of IT systems is continuously monitored. The IT service providers manage these systems with a team of resolvers. These resolvers act on the reported problems and take corrective actions to ensure smooth functioning of IT and business. The system problems are reported to the resolvers in two ways:

- *System-generated issues:* Components such as business applications, processes, CPU, disk, and network interfaces are monitored to detect anomalies. The monitoring tools capture and report any abnormal behavior of system components in the form of alerts.
- *User-generated issues:* The users of IT systems report problems through calls, emails, or chats.

Both system-generated and user-generated issues are reported in the form of *tickets* to a team of resolvers. A ticket contains various details of the reported problem such as the reporting time, system-of-origin, severity, and other details. In addition to this, each ticket also contains a description field that contains the details of the observed problem. In case of system-generated tickets, this description is automatically generated by the monitoring and alerting tools. This description is often structured and is based on how the alerting tools have been configured to report a problem. However, in case of user-generated tickets, this description contains free-form text written by users. This description is unstructured and contains variations, ill-formed sentences, and spelling and grammar mistakes.

While the ticket description contains many details, it is important to extract the information of the actual problem referred by the ticket. We refer to the problem referred by the ticket as *issue.* For instance, consider the following ticket description

> PATROL alert for filesystem apps-orabase is 90% full on d-2hh4knn

The *issue* referred by this description is

> PATROL alert for filesystem full

Extraction of issues occurring in the IT system can provide crucial information to better understand and control the IT operations. It can assist in improving both the IT system and the human system involved in the operations. The IT system consists of the business functions, applications, and the infrastructure. The human system consists of the teams of resolvers that manage the IT systems. Below are some examples of how the knowledge of issues can assist in improving these systems.

1. *Improving the IT system:* (a) The knowledge of issues observed in the IT system helps in inferring problem trends and frequent problematic areas in the system. (b) The issues referred by the high-severity tickets can provide insights into the critical areas that cause customer unrest or business instability. These issues can be prioritized for taking corrective actions by problem management. (c) Knowledge of frequent issues observed in specific domains can assist the service providers to prepare a service catalog. A service provider uses the service catalog for knowledge acquisition, generating training plans, developing automation scripts, etc.
2. *Improving the human system:* (a) The knowledge of issues can be used to identify issues that consume maximum effort of the resolvers. These issues can be considered for full or partial automation. (b) The knowledge of frequent issues can be used to prepare training plans to train resolvers with appropriate skills. (c) The knowledge of arrival patterns of the issues can be used

The 20th International Conference on Management of Data (COMAD), 17th-19th Dec 2014 at Hyderabad, India.

to plan shifts with right number of resolvers with right skills.

Mining issue from the ticket descriptions presents several challenges. (a) The same problem is represented in different ways in different tickets. (b) The organizations customize the structure of the description according to their own preferences. (c) The descriptions are domain-dependent and contain domain-specific words. For example, Oracle issues have words such as tables, SQL query, filesystem, disk, and Oracle-specific error codes. (d) The descriptions in user-generated tickets are written in free-form text. They are often ambiguous, and contain spelling and grammar errors.

Currently, service providers lack a systematic approach to capture the knowledge of issues. They either rely on the coarse-grained information from the ticketing tools or on the intuition-driven inputs provided by the resolvers.

- Ticketing tools such as ServiceNow [1] and BMC Remedy [2] allow resolvers to classify each ticket in various categories and subcategories. These categories are often referred as Category, Type, Item, and Summary (CTIS) [3]. However, most often these tools are not configured to sufficient level of details. For instance, various disk-related issues such as *disk full*, *disk failure,* or *disk corruption* are all classified as *Category = Infrastructure, Type = Hardware, Item = Storage, Summary = Disk issue.* In addition to that, resolvers and users make mistakes in rightly classifying the issue. As a result, the analysis based simply on this classification can often lead to incorrect and insufficient knowledge of the issues.

- Another approach that is adopted to infer the issues is by analyzing the fixlogs used for ticket resolution. A fixlog of an issue is a document that contains the details of the resolution steps to act on the issue. A resolver after acting on an issue makes an entry in the ticketing tool of the fixlog used to resolve a ticket. However, there are several practical challenges faced in this approach. Often the fixlogs are not available for all issues. Many times, the fixlogs are too generic and are used for many issues. In addition to this, the resolvers often do not make an entry of the fixlog in the ticketing system.

- The most common approach used to infer issues is the manual intuition-driven approach. The service providers infer the issue information either from the monthly reports manually produced by the teams or by talking to the domain experts. This approach carries the risk of being incomplete, inaccurate, and of variable quality.

In this paper, we propose an analytics-driven approach to mine the textual descriptions of tickets to extract issues. We propose the following approach of issue extraction.

- We propose to use information retrieval techniques along with domain knowledge to extract the issues from tickets.

- We propose two different algorithms to extract issues from system-generated and user-generated tickets. There is an inherent difference in the structure and heterogeneity in the descriptions of system-generated and user-generated tickets. The system-generated tickets have structured and consistent descriptions. We propose a clustering-based technique for system-generated tickets. On the other hand, the user-generated tickets have verbose and ambiguous descriptions. We propose a keyword-discovery-based approach for user-generated tickets.

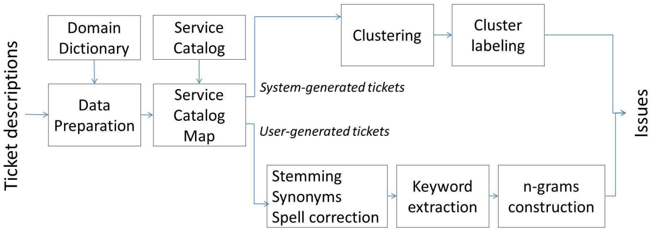


Figure 1: Functional diagram of the proposed approach.

- We capture the knowledge of IT and business domains in the form of dictionaries of include and exclude list of words and regular expressions. The knowledge consists of words and patterns pertaining to (a) the technology domain such as Oracle, Windows, and Linux, (b) business domain such as banking, retail, and finance, (c) monitoring tools such as ControlM, Autosys, and BMC Patrol, and (d) organization-specific naming conventions

- Finally, we present validation of the effectiveness of our techniques through two real word case-studies.

In Section 2, we present design rationale of the proposed issue extraction techniques. In Section 3, we present the proposed approach for extracting issues for system-generated and user-generated tickets. In Section 4, we demonstrate the effectiveness of the proposed approach using two real-world case-studies. We present related work in Section 5 and conclude in Section 6.

2. DESIGN RATIONALE

In this section, we present the proposed approach for extracting issues from ticket descriptions. Various factors need to be addressed while extracting issues from ticket descriptions.

Consider the following description:

> *ControlM Alert job=orahAAAA04 memname=orahotback.sh node=aaaa04 msg=sev3 job failed AAAA04*

The issue referred by this description is as follows:

> *ControlM Alert job failed*

The description contains other details such as name of the application *job=orahAAAA04*, name of the script *memname=orahotback.sh*, and severity level *msg=sev3*. To accurately extract the issues, it is critical to separate the factual knowledge of the issue from the specific details of its individual manifestations.

Different tickets describe the same issue in different ways. For instance, the above example can be reported as

> *ControlM Sev 3 Alert on Job Failure: job=orahAAAA04, script orahotback.sh*

In case of user-generated tickets, this problem becomes more challenging because of lack of structure, too many variations, and spelling and grammar errors. For instance, consider the following issue.

> *application not responding*

This issue is addressed by users in a variety of ways, such as,

> *application hung,*
> *application dead,*
> *No response from application since last 5 mins,*
> *application not responding for the fifth time since morning.*

While extracting issues, it is important to address such variations and generate a uniform issue for its various manifestations.

As discussed in Section 1, there is an inherent difference in the structure and heterogeneity in the descriptions of user-generated and system-generated tickets. Hence, we propose two different approaches to extract issues. Figure 1 presents the functional diagram of the proposed approach. The input is a set of ticket descriptions and the output is the issue extracted for each description.

Below we present the details of different functional elements of the proposed approach:

- *Data preparation:* The first step is to clean the descriptions, remove unwanted details, and tokenize. This step needs to be customized by knowledge of specific domains to ensure that all domain-specific words, patterns, and phrases are retained during data preparation.

- *Mapping to service catalog:* The commonly occurring issues in a domain are often captured in the form of a service catalog. For instance, service catalog of a technology domain of Oracle contains the list of commonly occurring issues in Oracle such as tablespace full, query running slow, index not built, etc. We use service catalog to mine these commonly occurring issues from the ticket descriptions. For the remaining ticket descriptions, we adopt the following approach.

- *System-generated tickets:* System-generated descriptions have a fixed structure and limited variations. Hence, we use clustering to group similar issues. We form label that best represents the descriptions in a cluster.

- *User-generated tickets:* On the other hand, user-generated descriptions demonstrate too many variations. Here, clustering is not effective. Hence, we first apply various techniques to address these variations such as use of stemming, synonym detection, and spelling corrections. We then extract keywords and group similar issues based on the commonality of keywords. We represent each group using an n-gram of keywords.

- *Knowledge cartridge:* Note that, the proposed approach is independent of any specific technology domain, and is designed to be configurable to extract issues for ticket descriptions of any domain. The effectiveness of the proposed approach can be significantly increased by domain-specific customizations of data preparation and service catalog mapping. We propose to make cartridges for technology domains (Unix, Oracle, Windows, etc.), business domains (banking, retail, finance, etc.), and tool domains (BMC Patrol, Control M, Tivolli TWS, Autosys, etc.) For each domain we maintain a dictionary of domain-specific words and patterns to include or exclude while extracting issues. We also maintain a service catalog of known issues within the domain.

In the rest of paper, we present details of these functional elements.

3. PROPOSED APPROACH

In this section, we present the approach of issue extraction from descriptions in system-generated and user-generated tickets.

3.1 Data preparation

As discussed earlier, *issue* is the problem reported in the ticket description. Ticket description contains lot of additional information such as timestamp, location, threshold, system-of-origin, severity, etc. This additional information changes as the same issue occurs at different timestamp, location, system-of-origin, etc. Though this information is important, it misleads issue extraction and hence must be separated. For example, consider a description such as

> *Unix server dt2n1g1 down.*

Here, the actual issue is

> *Unix server down.*

The name of system-of-origin *dt2n1g1* is additional information. Similarly, consider a description such as

> *filesystem 01hw4884 80% full.*

Here, the actual issue is

> *filesystem full.*

Here, the name of the system-of-origin and the current value of occupancy of space in the filesystem is additional information.

The challenge is to find the right set of words that should be removed or retained. Removing and retaining of words could have significant impact on subsequent issue extraction steps. We approach this problem by making following observations.

- *Issue is made up of dictionary words:* Consider following examples of description.

  1. *A-IM002930432.Pl reset the Unix password for login id bchho02 @ 156.5.238.69,*
  2. *27488732 CREATED unable to backup data after upgrade to windows 7,*
  3. *A-IM002971223–Need to revoke admin access,*
  4. *Domain Drive M:\ setting\ application data\SAP access denied.*

  It can be observed that issues are often described in plain English dictionary words, while all other details are non-English words. For example, in description 1 above, issue is described by

  > *reset the Unix password for login id.*

All the other words such as *A-IM002930432*, *Pl*, and *bchho02 @ 156.5.238.69* are all non-English words.

- *Some domain-specific words related to the issue are non-English:* Consider a description such as

  > *Portal Alert on appax040 at 2012-07-11 11:44:40.000 SwapPercentUsed is CRITICAL.*

  Ideally, the issue described in above description is

  > *Portal alert for SwapPercentUsed is critical.*

  But, the approach of retaining only English words identifies

  > *Portal alert for is critical*

  as the issue. Here, *SwapPercentUsed* is a non-dictionary word without which the issue is meaningless.

- *Some domain-specific words are not related to the issue but are English:* Some English words need to be excluded since they are domain-specific names of system-of-origin. Consider a description such as

  > *Portal alert on d-6x9ttk1: /export/home FileSystem is full.*

  The approach of retaining only English words identifies

  > *Portal alert on export home filesystem is full*

  as issue. In Unix, name of the *filesystem* often contains folder names such as *home*, *root*, *export*, etc. Hence, we exclude such words while extracting an issue.

Based on above observations, we provide following solution.

- We use English dictionary and mark each word as English or non-English. We separate non-English words and retain only the English dictionary words related to the issue.
- We prepare an *Include list* that contains domain-specific non-dictionary words and captures knowledge about a particular domain. The words in the *Include list* are retained in the description since they describe the issue. This list is populated by subject matter experts (SME) of respective domains. Some example words in *Include list* for Unix domain are *FSCapacity_Alarm*, *SwapPercentUsed*, etc.
- We prepare an *Exclude list* that contains domain-specific English words. Some examples of words in *Exclude list* are *export*, *home*, etc. for the domain of Unix. We make use of this *Exclude list* to remove such words.
- We also provide a lever to detect patterns in the form of regular expressions. For example, in case of Oracle, *Include list* contains patterns such as *ora** that denotes Oracle errors and *memname = *.sh* that denotes the shell script.

The pseudo code for the data preparation step is as follows.

```
1. Input and Output
    • Input: $D_r$ = List of description, $D$ = List of dictionary words, $I$ = List of include words, $E$ = List of exclude words, $S$ = List of special characters
    • Output: $D_c$ = List of cleaned description
2. Prepare a list of words, $W$, by tokenizing all descriptions in $D_r$ by space.
3. For each word in list of words, $W$,
    • Check if word is a subset of domain include list, $I$
    • If no, tokenize word by special characters, $S$ and add it to $W$
4. Prepare a list of non-dictionary words, $ND = W - (D + I)$
5. Add list of exclude words to non-dictionary words list, $ND = ND + E$
6. Remove non-dictionary words in $ND$ from descriptions $D_r$ to make a list of cleaned descriptions, $D_c = D_r - ND$
```

3.2 Service catalog mapping

Service catalog of a domain is a predefined set of issues in a domain for which resolution steps are known. It contains factual knowledge of the issues in a domain. The service catalogs are based on various technologies in IT infrastructure such as Unix, Linux, Oracle, Windows, etc. While extracting issues, we make use of this prior knowledge of known issues, by mapping cleaned descriptions, obtained by data preparation step, to the service catalog items. The service catalog makes the search guided and allows customization for each domain. Hence, many issues can be easily extracted that are otherwise difficult to extract. For example, consider a description as

> *D-8Y6TTK1 - cluster failover issue.*

Here this issue can be easily mapped to service catalog item

> *Cluster FailOver/FallBack*

in Linux service catalog.

Service catalog is structured, well defined, and contains known and finite set of issues. On the other hand, ticket descriptions are unstructured, unknown and can occur in various ways. The challenge is to map ticket descriptions with service catalog items.

Each service catalog item is defined as a two-tuple <*action object*>. For example, action = *create*, and object = *table*. However, ticket descriptions may refer to action in different ways. For example, *create table* can be referred as *make table*, *construct table*, etc. Hence, to best map service catalog item to descriptions, we construct multiple synonyms of action. Thus, for a service catalog item, if both object and action are present in the description, then we map the description to the corresponding service catalog item. The action and object keywords are either domain sensitive words or English words. For domain sensitive words, domain experts are required whereas for English words, an English thesaurus can be used for synonym creation. For example, in service *diskdetectionissue*, domain sensitive words can be *disk*, *drive*, *tape* and English words are *detect*, *recognize*, *identify*, etc. While mapping descriptions to service catalog items, following cases may occur.

1. *One service catalog item maps to multiple descriptions:* The descriptions contain many variations of the action and the object. These descriptions are mapped to a single service catalog item. For example, consider descriptions as follows:

    - *Unlock account,*
    - *Unlock password of my account,*
    - *Requesting unlocking of account.*

    All these cleaned descriptions contain <*action object*> pair <*unlock account*>. As a result, they all map to service catalog item *UnlockAccount*.

2. *One description maps to many service catalog items:* This can happen in following cases.

    - Issue is composite: Consider a description as

        > *After installing Windows 7, system is very very slow. For unlocking machine, system takes more than 20 min. Please resolve this ASAP.*

        We observe that the actual issue might be

        (a) *windows installation* problem, or
        (b) *account related unlocking* problem.

        Here, the description contains multiple <*action object*> pairs such as <*install windows*>, or <*unlock system*> which maps to multiples items in service catalog.

    - Service catalog is too detailed: Consider a description as

        > *backup failure.*

        This gets mapped to multiple service catalog items as

        (a) *Full backup failure,*
        (b) *Incremental backup failure* ,
        (c) *Differential backup failure* ,
        (d) *Cumulative backup failure* ,
        (e) *Archive log backup failure* .

Following is the pseudo code for mapping descriptions to service catalog item.

```
1. Input and Output
    • Input: $D_c$ = List of cleaned description $SC$ = Service catalog
    • Output: Mapping $< d_i, SC_j >$
2. For each cleaned description $d_i$ in $D_c$
    • For each item $SC_j$ in service catalog $SC$ with object $O_j$ and action $A_j$
        – If object, $O_j$ and action $A_j$ keywords are present, assign the corresponding service catalog item to cleaned description $d_i$
```

3.3 Issue extraction in system-generated tickets

In this section, we present the approach of extracting issues from remaining descriptions that do not map to service catalog.

3.3.1 Clustering

The information unrelated to issue is separated from the descriptions in the data preparation step. The cleaned descriptions still contain variations based on the configuration of the monitoring tools for different applications in the system. We address these variations by clustering the descriptions based on similarity. Clustering helps grouping similar descriptions together and assign dissimilar descriptions to separate groups.

Consider two descriptions as

- *BMC portal alert on d-hw6ttk1-lvmh: /var for filesystem full*, and
- *BMC portal critical alert on d-hw6ttk1-lvmh/var for filesystem full* .

After cleaning, the descriptions are

- *BMC portal alert on filesystem full*, and
- *BMC portal critical alert on filesystem full.*

These two descriptions are same except the word *critical* present in the second description. We group such similar descriptions into one cluster by computing similarity between two descriptions.

Some of the approaches to compute similarity [8] between two descriptions are Jaccard coefficient, Dice coefficient, etc. Dice coefficient gives twice the weight to common elements. Since we emphasize on commonality, we use Dice coefficient to compute similarity between two descriptions. Let A and B be sets of words in two descriptions. Dice similarity, D, between A and B is defined as follows:

$$D = \frac{2 * |A \cap B|}{|A| + |B|} \tag{1}$$

For example, if

- A = *BMC portal alert on filesystem full*, and
- B = *BMC portal critical alert on filesystem full*,

then $|A| = 6$, $|B| = 7$, $|A \cap B| = 6$ and $D = \frac{2*6}{7+6} = 0.923$.

We compute Dice similarity between every pair of clean description. We construct a similarity graph of clean descriptions in which nodes are clean descriptions. There is an edge between two clean descriptions if they are similar. We consider two clean descriptions similar if the similarity coefficient between them is greater than a predefined threshold *threshold_similarity*. We cluster clean descriptions by applying graph clustering on the similarity graph of clean descriptions. Various graph clustering techniques can be used for clustering such as cliques [4], connected components [5], graph partitioning [9], graph cuts [7], etc.. We have used cliques to identify clusters of clean descriptions. The similarity threshold can be set automatically using certain heuristics such as *median* or *mean + (k * standard deviation)* of the distribution of similarity threshold values.

3.3.2 Cluster label

A single cluster contains many variations of descriptions. The next step is to provide a label to each cluster that best represents all the members within a cluster. The set of common words from all descriptions within a cluster are probable candidate for a cluster label. If we arrange these common words in any order, then the cluster label cannot be easily understood and is not meaningful. For example, a label, *host connect unable to*

does not make sense. While we have narrowed down to the words to be used for the label, it is also important to place them correctly to form meaningful phrase. For example, the correct order of words for the previous example is

unable to connect host.

One of the criteria to compute the position of a word is based on its position in individual description. For instance, the word that occurs most frequently on 1st position is placed in the 1st position in the label.

Following is the pseudo code of clustering and labeling algorithm we have implemented.

```
1. Input and Output
    • Input: $D_c$ = List of cleaned description, $threshold\_similarity$ = similarity threshold
    • Output: Mapping $< d_i$, Issue label $>$
2. Compute dice similarity for each pair of cleaned description $< d_i, d_j >$ in $D_c$,
    • IssueAdjacencyMatrix[i,j] = Dice similarity coefficient
3. Build adjacency matrix for the issue-similarity graph
    • If (IssueAdjacencyMatrix[i,j] $\geq$ $threshold\_similarity$), IssueAdjacencyMatrix[i,j] = 1
    • else IssueAdjacencyMatrix[i,j] = 0
4. Identify a maximum clique $C_k$ in IssueAdjacencyMatrix
5. Remove issues identified in clique $C_k$ from IssueAdjacencyMatrix
6. Repeat step 4 till all issues are covered
7. For each clique $C_k$, identify a label
    • Identify set of common words $C_w$ from the set of cleaned description belonging to the clique
        – For each word $w$ in $C_w$,
            * Compute its position in set of cleaned description belonging to clique, $p$ = Mode of the position of the word $w$
            * Label of the clique = Arrangment of the words in $C_w$ according to $p$
```

3.4 Issue extraction in user-generated tickets

In this section, we present the approach of extraction of issue from user-generated tickets.

3.4.1 *Preprocessing*

User description is free-form text primarily written by users who face issues. Such descriptions contain lot of ambiguity, grammatical errors, spelling mistakes, different form of same words, etc. This results in variations in describing the same issue. For example, consider descriptions

- *job running late,*
- *job execution delayed,* and
- *abnormal delay observed on job exec.*

All these descriptions, mean the same issue, that is,

job running late,

but are written differently.

To overcome this problem, we use following different levers.

1. Stemming: Users write same words in different forms such as *lock*, *locked*, *freeze*, *freezing*, *connecting*, *connect*, etc. according to their position in description. We make use of Porter stemmer algorithm [16] and replace these words with their root such as *lock*, *freez*, *connect*, etc.

2. Spelling correction: Users often make spelling mistakes while describing issues. We identify such words and perform spelling correction [15]. For example, users often write *password* as, *passwd*, *pasword*, etc. By using spelling correction, we correct these words.

3. Synonyms: Consider examples such as (1) *remove*, and *delete*, (2) *modify*, and *change*, (3) *big* and *large*, etc. These words are often used in place of each other by the users while describing a problem. We make use of WordNet [10], which is a lexical database for English language, to automatically detect synonym words and make them consistent.

We apply these levers, to ensure that all variations in the descriptions are made consistent. By applying levers of Stemming, Spelling correction and Synonyms on above descriptions, we obtain

- *job exec delay,*
- *job exec delay,* and
- *abnormal delay observed on job exec.*

3.4.2 *Keyword extraction*

The next step is to extract issues from the processed set of descriptions. We first extract keywords from descriptions which occur frequently, such as *job*, *memory*, *filesystem*, *swap*, etc. While extracting the keywords we consider only nouns, adjectives, verbs, and adverbs. This makes the word search space limited. We consider top frequently occurring keywords to further form clusters. Another option could be to use TF-IDF for selecting top keywords. Larger the number of keywords considered, larger is the number of clean descriptions for which we can extract issue and hence a larger coverage.

3.4.3 *N-gram construction*

We next construct n-gram out of these extracted keywords. For this, we tag each of these keywords to the descriptions where they occur. For each keyword, we identify descriptions to which they are tagged and extend this keyword to form n-grams by identifying other words which appear in the descriptions. Hence, we make n-grams of these keywords such as

- *job failure,*
- *high memory utilization,*
- *filesystem backup,*

- *create swap space*, etc.

which describe the issue in a better way. Each of these n-grams represent issues and we map each of these n-grams to descriptions. The set of descriptions corresponding to each n-gram represents a cluster. Further, the n-gram is considered as the label of the cluster as well. max_n is the maximum length of n-gram considered for issue extraction and label. A longer n-gram enables a better explanation of the issue and larger correctness.

Following pseudo code describes our approach to extract issues from user-generated tickets.

```
1. Input and Output
     • Input: $D_c$ = List of cleaned description
     • Output: Mapping < $d_i$, Issue label >
2. Compute all one words and their frequency of occurrence in cleaned descriptions.
3. Select top $k$ one words as keywords.
4. For each word, construct n-gram until the issue is explained by n-gram
5. These set of n-grams represent issues. Assign all n-grams to cleaned descriptions.
```

4. CASE STUDIES

We have applied the approaches proposed in this paper on various real-world case-studies. In this section, we present two real-world case studies, one demonstrating the approach used for system-generated tickets and other for user-generated tickets.

4.1 Evaluation of the approach for system-generated tickets

In the first case-study, we applied the proposed approach on the Unix domain of a major retailer in the US. We analyzed ticket history of 6 months where 3854 tickets were produced. Out of total 3854 tickets, we were able to extract issues for 90.32% of the tickets. We validated the correctness of extracted issues by manual verification by domain expert. For each ticket, the domain expert compared ticket description with extracted issues and tagged the ticket as correct or incorrect extraction. We correctly extracted issues for 84.37 % of tickets. An example of correct issue extraction is as follows : Consider a description as

> *Patrol alert on udbax2021: STAGE N3 Filesystem udb-data etlsi01 data995_0 is not mounted on UDBAX202*

and correctly extracted issue is *Patrol alert on stage filesystem is not mounted* An example of incorrect issue extraction is as follows : Consider a description as

> *Portal Alert on bllaplx104e at 2012-11-28 23:30:58.000 the Swap Space Percent Available is CRITICAL*

and extracted issue is *Portal Alert for number of processes.* We were not able to extract issues for the 9.68 % of the tickets primarily due to insufficient ticket descriptions. These ticket descriptions did not contain any details of the ticket and only referred to the affected system-of-origin or the time of occurrence.

1. *Evaluation of data preparation:* We next present evaluation of data preparation step. As we described earlier in Section III A , we make use of domain-specific *Include list* and *Exclude list* of words.

    - *Include list:* We constructed a dictionary of frequently occurring non-English words in Unix domain such as *bmc*, *portal*, *ip*, *filesystem*, etc. We also made regular expressions such as *fscapacity_**, *vcluster_**, etc. We observed that 2970 tickets (77.06%) contained domain words and regular expressions which otherwise would not have been captured.
    - *Exclude list:* The exclude list contained English dictionary words that have specific meaning in Unix environment and should not be considered for issue extraction. For example, *home*, *export*, *root*, *etc* are folder names in filesystem. Hence these words need to be removed. 45.92% of tickets contained such exclude words.

    We translated each description to cleaned description. We generated cleaned descriptions for 90.22% of tickets. Remaining 9.68% of tickets correspond to 372 tickets. These remaining descriptions contained only non-English words and hence, no issues were generated for these tickets. For example, consider description

    > texcemdbin051d - texcmedbin052d

    After cleaning, many otherwise different looking descriptions became consistent. After cleaning 3854 descriptions, we generated 47 unique cleaned descriptions.

2. *Evaluation of service catalog mapping:* We next demonstrate the impact of using service catalog mapping. The service catalog of Unix domain contained 110 items of frequently occurring issues in Unix environment. By using proposed algorithm on cleaned descriptions, we observed a match in 21.01% of tickets. One of the most dominant match was for the service *"Address Space-Crunch"*. Note that this service catalog item matched to 5 different cleaned descriptions such as

    > BMC portal alert on dell filesystem is at x percent full,
    > BMC portal alert on filesystem is critical at x percent full.

    By using service catalog mapping algorithm, we were able to map 5 out of 47 cleaned descriptions to service catalog items. After service catalog mapping, 42 cleaned descriptions remained corresponding to 3013 tickets.

3. *Evaluation of clustering algorithm for system-generated tickets:* Many cleaned descriptions were similar and had only slight variations. We applied the proposed clustering approach to group these cleaned descriptions. We were able to group 42 cleaned descriptions into 19 groups. Figure 2 (c) shows the size of these 19 groups where size of a group refers to the number of cleaned descriptions in the group. Note that, the number of groups depends on the threshold *threshold_similarity*. We later present experiments for varying threshold values used for clustering.

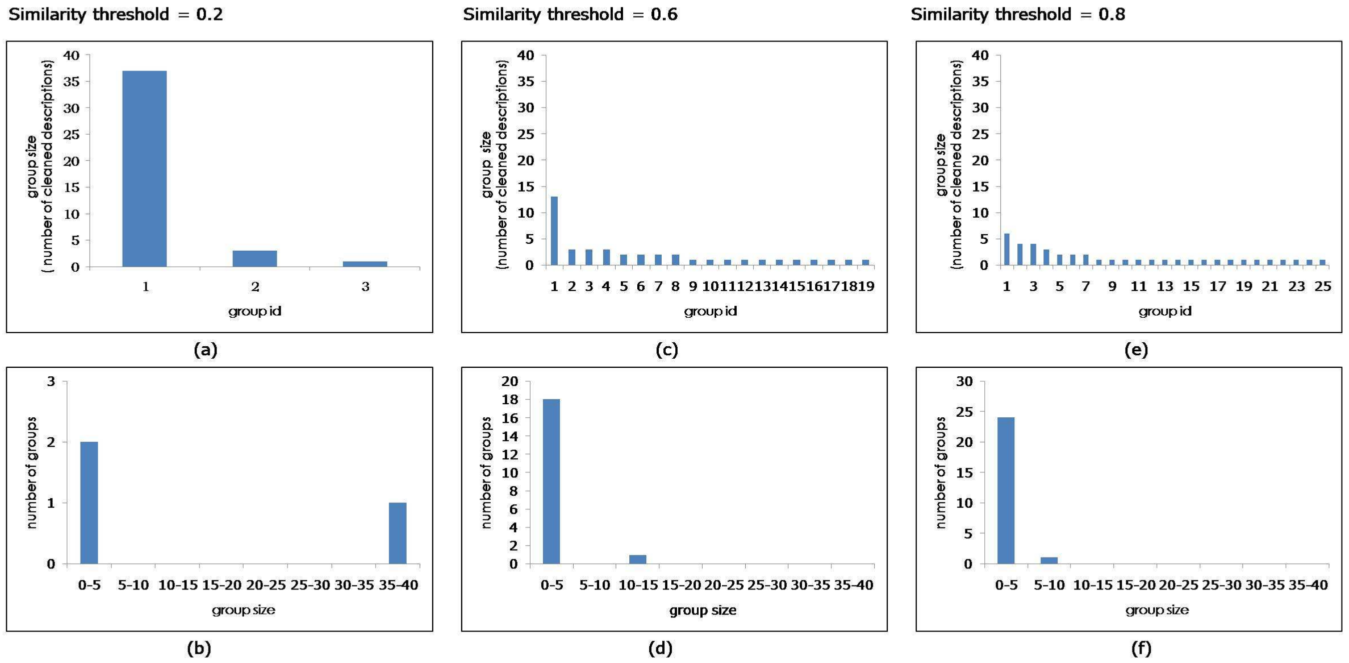


Figure 2: System-generated tickets: group id vs size of each group

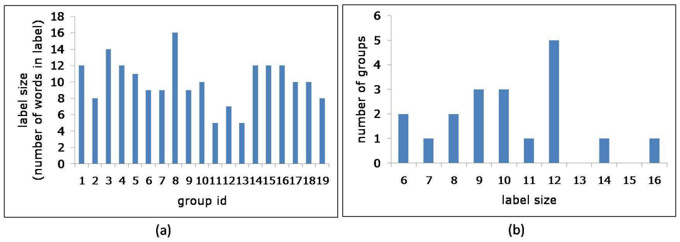


Figure 3: System generated tickets: (a) group id vs. label size. (b) label size vs. number of groups

4. *Evaluation of labels of clusters:* The next step is to assign label to these 19 clusters. As explained in Section 3, the set of common words from all descriptions within a cluster are probable candidate for a cluster label. We were able to assign long labels to most groups. Figure 3 (a) shows the label size for each of the 19 clusters. Average label size for each cluster was 10.05 words with maximum being 15 and minimum being 5 words. There is a structure and commonality in system-generated tickets. Hence we could generate long labels. Figure 3 (b) shows label size plotted against number of groups. We observe that there are 5 groups having label size of 12 words.

5. *Evaluation of change in threshold threshold_similarity:* Note that, the threshold used to compute similarity between cleaned descriptions impacts how many number of clusters are constructed. We experimented with threshold values 0.2, 0.6 and 0.8. Figures 2 (a) (c) (e) show the group ids and their size (number of cleaned descriptions) for thresholds 0.2, 0.6 and 0.8 respectively. Smaller threshold places loose constraint on similarity and hence groups many irrelevant descriptions together. Hence, the algorithm constructed only 3 clusters having average size of 13.66 cleaned descriptions with 0.2 threshold. On the other hand, a larger threshold of 0.8 places a very strict constraint on similarity and does not even group different variations of same issue. Hence, the algorithm constructed 25 clusters having average size of 1.68 cleaned descriptions with 0.8 threshold. Similarity threshold of 0.6 balances this trade-off making 19 groups with an average size of 2.16 cleaned descriptions. Figures 2 (b) (d) (f) show the group size plotted against the number of groups. We observe that as threshold increases, smaller groups are formed.

6. *Evaluation of similarity threshold against accuracy:* For measuring the impact of similarity threshold against accuracy, we selected similarity thresholds as 0.2, 0.6 and 0.8 for clustering and computed accuracy as 41.12%, 76.18% and 84.37% respectively. Accuracy increases with increase in similarity threshold because high threshold enforces more similarity and hence less accidental matching.

7. *Comparison with state of the art:* The most common approach to classify a ticket is based on CTI as discussed in [12] and [13] and explained in Section 1. We compare the accuracy of proposed approach with that of CTI. It can be seen that the accuracy of CTI is 55.34 % while that of proposed approach is 84.37 %.

4.2 Evaluation of the approach for user-generated tickets

In this case-study, we applied the proposed approach for user-generated tickets on Windows domain of a major retailer in the US. We analyzed ticket history of 8 months

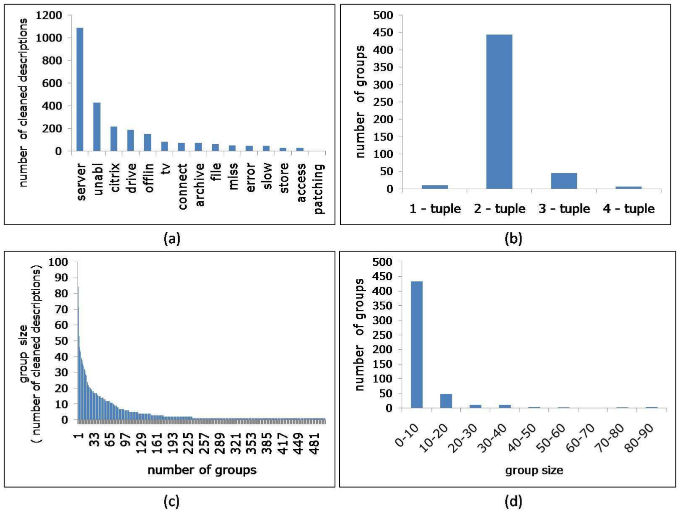


Figure 4: User generated tickets: (a) keywords Vs. number of cleaned descriptions (b) n-gram Vs. number of groups (c) number of groups Vs. group size (d) group size Vs. number of groups

where 6909 tickets were produced. Out of total 6909 tickets, we were able to extract issues for 89.34% of tickets. We verified the correction of extracted issues from domain experts. 81.63% of these issues were correctly extracted.

1. *Evaluation of data preparation:* We added 51 domain-specific words to the *Include list* such as *citrix*, *putty*, *sharepoint*, and *ip*. We observed that 4496 tickets (65.07%) contained domain words present in the *Include list*. We excluded some words such as *user*, *warning*, *etc.* We observed that 1629 tickets (23.58%) contained the words present in the *Exclude list*. We received total 6909 ticket descriptions. After data preparation step, these ticket descriptions got converted into 3460 cleaned descriptions.

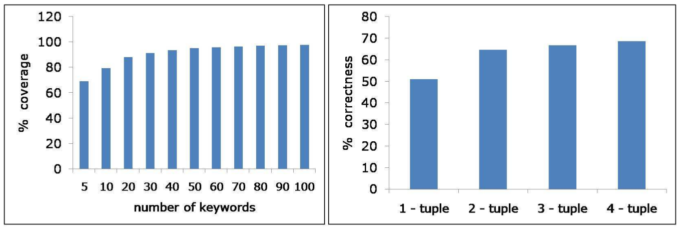
2. *Evaluation of service catalog mapping:* We next demonstrate the impact of using service catalog mapping. The service catalog of Windows domain consisted of 44 items of frequently occurring issues. We used proposed algorithm on the cleaned descriptions and observed a match of 2136 tickets (30.92%). 20 service catalog items mapped to 363 cleaned descriptions. We observed that there were two items which were dominant and covered 310 cleaned descriptions. Service catalog item of *Monitoring Disk Utilization* covered 198 different cleaned descriptions and other item of *Limited Permissions FileShare* covered 112 different cleaned descriptions. By using service catalog mapping algorithm, we were able to map 363 out of 3460 cleaned descriptions to 20 service catalog items. After service catalog mapping, 3097 cleaned descriptions remained corresponding to 4043 tickets.

3. *Evaluation of clustering algorithm for user-generated tickets:* Unlike system-generated tickets, we observe many variations in user-generated tickets. We selected 15 words as keywords from 883 words based on frequency and developed n-grams from them. Depending upon how the issue is written and what type of words are used to write the descriptions, the issues can be explained by a k-tuple. For this dataset, we considered maximum 4-tuples. We applied the proposed n-gram clustering approach to group the remaining 3097 cleaned descriptions and were able to form 505 groups. Figure 4 (a) shows 15 1-tuples with their corresponding number of tickets. 1089 (35.16%) of cleaned descriptions contained keyword *server*. We obtained 505 groups of n-grams from these 15 keywords. Figure 4 (c) shows all 505 n-grams plotted against their sizes where size of a n-gram denotes number of cleaned descriptions that are contained in this n-gram. The top 20 most frequent n-grams cover 856 out of 3097 tickets (27.64%). These 505 groups contain a mixture of 1-tuple, 2-tuple, 3-tuple and 4-tuple depending on how these tuples explain the issue. Figure 4 (b) shows n-grams plotted against number of groups. We observe that majority of the groups, that is 444 are explained by 2-tuple. Figure 4 (d) shows group size plotted against number of groups. We observed that many small groups were formed. We discovered 433 groups that have less than 10 cleaned descriptions.

4. *Effect of number of keywords on coverage of tickets:* The number of keywords used to form n-grams has a direct impact on the number of tickets for which issues are extracted by the n-grams. We now evaluate the effect of varying the number of keywords used for issue extraction. We evaluated the algorithms for varying number of keywords ranging from 5 to 100. Figure 5(a) shows the number of keywords plotted against the number of tickets for which issues were extracted. As expected, the number of tickets for which issues are extracted increases with increasing number of keywords. However, for the given data-set the increase is only marginal after 30 keywords. Our experience on various data-sets indicate that, in most cases, top 50 keywords are sufficient to extract issues of more than 95% of the tickets.

5. *Evaluation of size of n-gram on correctness of issues:* The size of n-gram impacts the level of details captured about an issue. A single-tuple often fails to give sufficient details of the issue, such as *server*, or *application*. Instead, a 2-tuple or a 3-tuple better captures an issue, such as, *server failure*, or *application slow*. We refer to the maximum value of n that is used for creating n-grams as *max_n*. For the given set of tickets, we fixed the number of keywords and extracted issues with changing value of *max_n*. We then computed the correctness of the extracted issues for each experiment. Figure 5(b) shows the effect of different values of *max_n* on the correctness of the extracted issues. The correctness of issues increases with increasing value of *max_n*. However, for the given dataset we were able to extract correct issues for most of the tickets with 2-tuples. Only few tickets required the 3-tuples or 4-tuples to completely explain the issue. Hence, increasing the value *max_n* to 3 and 4 provided only marginal increase in correctness.

6. *Comparison with state of the art:* The most common approach to classify a ticket is based on CTI as dis-



(a) (b)

Figure 5: User generated tickets: (a) number of keywords vs % coverage (b) n-gram vs %correctness

cussed in [12] and [13] and explained in Section 1. We compare the accuracy of proposed approach with that of CTI. It can be seen that the accuracy of CTI is 28.56 % while that of proposed approach is 81.63 %.

5. RELATED WORK

Many researchers in the past have worked on performing various types of analysis on alerts and tickets based on various objectives. Some of these objectives are

1)*Automated problem inference*: Authors in [17] have proposed a system that aims to do automated problem inference for network trouble tickets.

2)*Resolver analysis*: Authors in [14] provide statistics-based discovery algorithms to identify experts, specialists and highly experienced people for specific service tasks.

3)*Problem occurrence pattern*: Authors in [12] develop a subgroup discovery technique and demonstrate its use to identify expensive problems in an IT Infrastructure Support (ITIS) organization. Authors in [13] propose statistics-based discovery algorithms to answer specific business questions related to ITIS operations such as identifying heavy hitter problems, identifying problems facing unusually high SLA violations and identifying problems suitable for automation, etc.

All above work requires knowledge of issues. Our work falls in the category of extracting issues from text. Ticketing tools such as ServiceNow [1], Remedy [2], etc. provide drop-down menus for classification of issues in classes such as Category, Type, Item and Summary (CTIS). However, the classification is often coarse grained, as the quality of classification is dependent upon the customization by the organizations. Further, the resolvers and users make mistakes in categorizing the issue. As a result, the analysis based on this classification is often misleading.

6. CONCLUSION

In this paper, we address the problem to mine textual descriptions in tickets to extract issues. We propose use of informational retrieval techniques along with domain knowledge to extract issue. We capture the knowledge of IT and business domains in the form of include and exclude lists of words and regular expressions. We propose two different algorithms to extract issues for system and user-generated tickets. In system-generated tickets, the algorithm is based on clustering descriptions based on similarity, while for user-generated tickets, we propose keyword based approach to cluster similar descriptions.

We have applied the proposed ideas on several real-world case-studies. The proposed algorithms were able to derive issues that were otherwise difficult to extract. We demonstrate the proof-of-concept of proposed approach using two real-world case-studies. One for system-generated tickets and other for user-generated tickets. In system-generated tickets, by using the proposed approach we were able to correctly extract the issues for 76.16% of tickets. Whereas, in user-generated tickets, we were able to correctly extract the issues for 81.63% of tickets.

7. REFERENCES

[1] `http://www.servicenow.com/products/it-service-automation-applications.htm%l`.

[2] `http://www.bmc.com/it-solutions/it-service-management.html`.

[3] `http://www.itil-officialsite.com/aboutitil/whatisitil.aspx`.

[4] Immanuel M Bomze, Marco Budinich, Panos M Pardalos, and Marcello Pelillo. The maximum clique problem. In *Handbook of combinatorial optimization*, pages 1–74. Springer, 1999.

[5] Fan Chung and Linyuan Lu. Connected components in random graphs with given expected degree sequences. *Annals of combinatorics*, 6(2):125–145, 2002.

[6] Kenneth Ward Church and Patrick Hanks. Word association norms, mutual information, and lexicography. *Computational linguistics*, 16(1):22–29, 1990.

[7] Daniel Freedman and Tao Zhang. Interactive graph cut based segmentation with shape priors. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 755–762. IEEE, 2005.

[8] Anna Huang. Similarity measures for text document clustering. In *Proceedings of the sixth new zealand computer science research student conference (NZCSRSC2008), Christchurch, New Zealand*, pages 49–56, 2008.

[9] George Karypis and Vipin Kumar. Metis-unstructured graph partitioning and sparse matrix ordering system, version 2.0. 1995.

[10] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[11] Makoto Nagao and Shinsuke Mori. A new method of n-gram statistics for large number of n and automatic extraction of words and phrases from large text data of japanese. In *Proceedings of the 15th conference on Computational linguistics-Volume 1*, pages 611–615. Association for Computational Linguistics, 1994.

[12] Maitreya Natu and Girish Keshav Palshikar. Interesting subset discovery and its application on service processes. In *Data Mining for Service*, pages 245–269. Springer, 2014.

[13] Girish Keshav Palshikar, Harrick M Vin, Mohammed Mudassar, and Maitreya Natu. Domain-driven data mining for it infrastructure support. In *Data Mining Workshops (ICDMW), 2010 IEEE International Conference on*, pages 959–966. IEEE, 2010.

[14] Girish Keshav Palshikar, Harrick M Vin, V Vijaya Saradhi, and Mohammed Mudassar. Discovering experts, experienced persons and specialists for it

infrastructure support. *Service Science*, 3(1):1–21, 2011.

[15] Joseph J Pollock and Antonio Zamora. Automatic spelling correction in scientific and scholarly text. *Communications of the ACM*, 27(4):358–368, 1984.

[16] Martin Porter. Snowball: A language for stemming algorithms, 2001.

[17] Rahul Potharaju, Navendu Jain, and Cristina Nita-Rotaru. Juggling the jigsaw: Towards automated problem inference from network trouble tickets. In *NSDI*, pages 127–141, 2013.

Supporting Math Trails on Property Graphs

Sai Sumana P[+]
IIT Guwahati
Guwahati 781039
Assam, India
s.pagidipalli@iitg.ernet.in

Veena S Kambi, Sudharani R Nakati
Sarada Research Labs
56/2 ‘Anugraha’ Seshadri Dr, Benson Town
Bangalore 560046, India
080-23545856
veena.kambi,sudha.nakati@saradaresearchlabs.org

Jagannathan Srinivasan
Oracle Corporation
One Oracle Drive
Nashua, NH 03063, USA
jagannathan.srinivasan@oracle.com

ABSTRACT

pg-MathTrails is an application developed to assess math skills of school students. It includes i) *topic hierarchy* to categorize the questions, ii) *question template repository* using which teachers can add different types of questions, iii) *question play module* to play questions, iv) *student evaluation module* to evaluate student performance, and v) *recommender* that recommends videos residing in a *video repository*. A key feature is that each interactive student session is treated as a *walking trail* in the *property graph* formed by *topic* and *question* nodes, and student evaluation involves operations on the property graph. The application is being used in the after-school program of *Sri Sarada Academy of Excellence, Bangalore*.

1. INTRODUCTION

pg-MathTrails is a student assessment application built for school children. It is inspired by the fitness trails equipped with exercise stations (see Figure 1 [1]) typically found in parks and college campuses. Using the *pg-MathTrails* application, a student undertakes a *math trail* aimed to evaluate or strengthen his *math fitness* with respect to a particular *topic* and as he proceeds he is presented with exercises varying in levels of difficulty. The entire trail along with his attempts at the exercises is recorded in a *property graph* [2], which can be subsequently used by him or his teacher to analyze his performance (hence the name *pg-MathTrails*). The application records the trail, the order in which exercises are attempted, and for each exercise it includes whether correct answer was obtained, number of wrong attempts, and the time taken. Like a fitness trail, our application allows user to quit in middle.

Of course, our application also has some variations when compared to fitness trails. Two students can simultaneously be on the same trail attempting the same exercise at any given point of time. In addition, unlike a fitness trail where exercise stations are fixed, we have a choice of rearranging the exercises within a single level of difficulty so that there is a variation in the order in which questions are presented to different students.

Furthermore, the application monitors the student performance continuously and elevates him to challenging exercises if he is performing well.

Figure 1. A fitness trail with 20-station exercises at Woods Hole, MA, USA; top-right showing station 8.

On the contrary, if he is having difficulty with a particular exercise, our application encourages the student by giving a suitable hint. It also has an option to preempt the student from continuing if we find that his performance is below par. He is then presented with videos on the topic that he can watch to further learn that topic after which he can re-attempt that trail. Also, his complete performance at various exercises is available for him to review, made available by the courtesy of the underlying property graph.

We chose property graph as the basis for this application, because of its versatility. Specifically, property graphs allow associating one or more *properties* as *key-value* pairs with both *nodes* and *edges*. Furthermore, it has the flexibility in that two nodes can have different set of properties. For example, in our case *topic* nodes and *exercise* nodes differ in their properties. In addition, the nodes of single type, for example *exercise* nodes, can differ in their properties. An *exercise* node, which is skipped, would not have *wrongAnswers*, and *timeTaken* properties specified.

The 20th International Conference on Management of Data (COMAD),
17th-19th Dec, 2014 at Hyderabad, India.

[+] This work was done as part of a summer internship at Sarada Research Labs, Bangalore.

We also benefit from the ability to associate *key-values* pairs with *edges*. For math trails, we associate for edges connecting nodes, a key/value pair for key *autoSkip* that indicates if an optional question was skipped as student was performing well (See Section 2.3 for more details).

Challenging aspects in development of the application are:

- Support variation in the trails, including supporting auto skipping of questions for student performing well, and forcefully exiting students who are performing poorly.
- Support multiple students to simultaneously walk the same topic's trail, with each student's information independently maintained.
- Support comparing two trails of a particular topic belonging to the same student or different students.
- Support student to suspend and resume a trail.
- Support adding questions of various types under a topic hierarchy.
- Support evolution of a trail pertaining to a particular topic, for example, addition, update, and/or removal of exercises.
- Support linking exercises to instructional videos.
- Even though developed for Math, the application should be usable for any other subject.
- The application should be driven by a light-weight custom property graph implementation that does not require interfacing with graph databases (such as Neo4j[3], Dex [4]) and yet allows interactive response time.

The complete application is developed as a database-centric web application using Oracle Application Express 4.0.2 (APEX) [5], which is a rapid application development tool, and uses Oracle Database 11gR2 Express Edition [6] as the backend database.

The application is being used in the after school program of Sri Sarada Academy of Excellence, Bangalore for children from grade 1 to grade 10. Also, to illustrate its working we present math trail data based upon the use of this application by children in the after school program.

The key contributions of this work are:

- The use of property graph to represent the question repository and student's math trails.
- A simple and customizable criterion for skipping questions as well as for pre-empting the user if his performance is below par.
- Providing detailed analysis of student's performance on a particular topic's math trail, including comparison of two trails on the same topic, and computing overall student's participation.
- A usability evaluation in an after school program.

The rest of the paper is organized as follows. Section 2 presents the key concepts. Section 3 covers the design and implementation. Section 4 gives a tour of the application. Section 5 reports the usability study. Section 6 covers the related work and Section 7 concludes the paper and outlines the future work.

2. KEY CONCEPTS

This section discusses the key concepts.

2.1 Terminology

A *Property Graph* is key*/value-based, directed, multi-relational, labelled* graph [2]. Specifically, it consists of a set of vertices, and a set of edges like directed labelled graph with the following characteristics:

- Each vertex has a unique identifier, a set of outgoing edges, a set of incoming edges, and a collection of properties defined by a map from key to value.
- Each edge has a unique identifier, a head and tail vertex, a label that denotes the type of relationship between its two vertices, and a collection of properties defined by a map from key to value.

The *multi-relational* aspect refers to the ability to associate multiple edges for same pair of vertices. This along with the ability to associate a set of *key/value properties* with both vertices and edges makes the property graph model suitable for our assessment application.

2.2 Basic Model for Question Repository

The questions are categorized under a topic hierarchy. The questions and the topic nodes together form the default graph. Each set of questions under a topic is referred to as the *topic default trail* and it consists of multiple groups of questions varying in their level of difficulty. Each group itself consists of a *mandatory set of questions* followed by zero or more *optional questions* that gets skipped automatically if the performance of the student is satisfactory.

Figure 2 shows a default topic trail for *Math→Arithmetic→Addition* topic, consisting of 9 questions organized under three levels. The optional questions shown in sand color appears towards the end of each level. The questions are connected to the leaf topic via *hasQuestion* edge. The leaf topic also connects to the first question by the *nextQuestion* edge. In addition, a question is connected to next question through the *nextQuestion* edge. The key/values pairs showed in bold (corresponding to properties *nm*, *lvl*, *opt*, *et*) are the key/values for the default topic trail.

2.3 Representing a Student Trail

The actual student assessment of a topic forms a *student trail*, which also is stored in the property graph. The trail is formed by questions visited by the student during the session. The questions are connected to the topic and to the next question node by the label *nextQuestion*. For each question node, the key/value properties include the ones for default topic trail as well as values for additional properties (*wa, ca, tt, tid*). The *tid* (refers to trail id) is not shown in Figure 2 to save space. A unique *tid* is generated for each student trail and associated with each of its question nodes. The student trail shown in Figure 2 in red indicates that the student only visited questions *q1*, *q2*, *q4*, *q5*, *q6*, and *q7*. The trail illustrates the concept of *auto_skip* that allowed him to skip optional questions, *q3*, *q8*, and *q9* as he was performing very well.

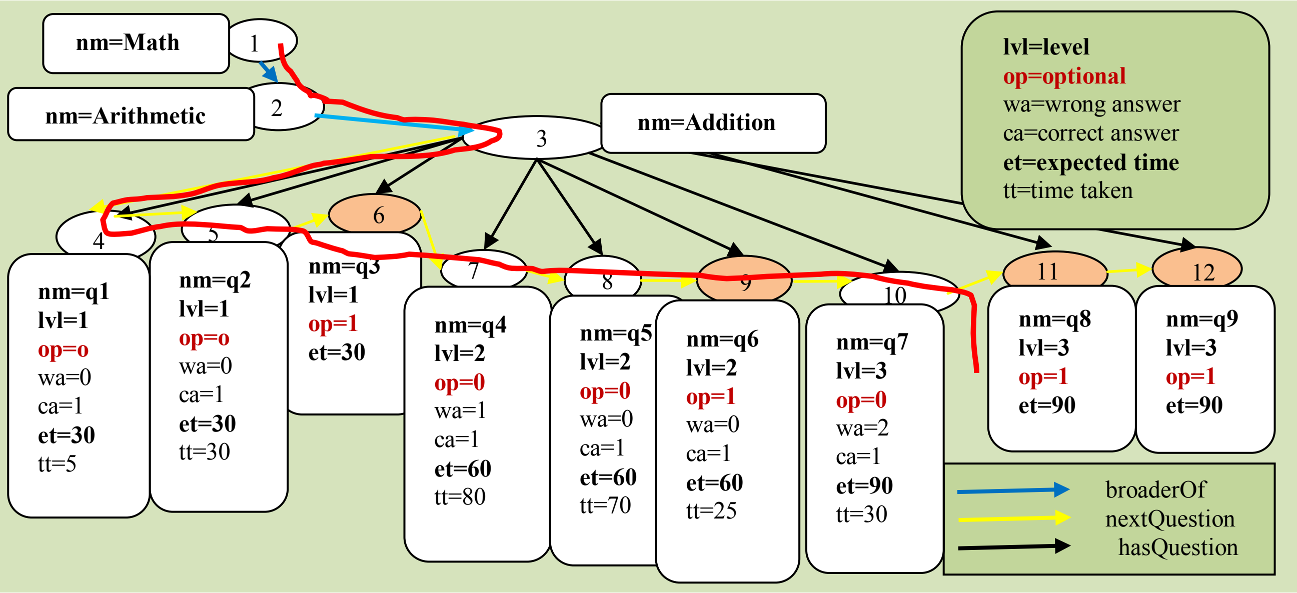


Figure 2. A Math topic trail with 9-station exercises; the actual trail undertaken by student shown in red.

The *auto_skip* criterion is defined as ability for student to skip optional questions if he has so far answered all questions visited correctly within the cumulative expected time.

Let a student topic trail be made of a set of nodes $Q=\{Q_1, Q_2, \ldots\}$ and each pair of consecutive *nodes* (Q_i, Q_{i+1}) are connected by *nextQuestion* edge. Each question Q_i itself has a set of properties (attributes), which can be referenced using '.' operator. For example, $Q_1.ca$ represents the value of *ca* attribute for question node Q_1. The *auto_skip* criterion can be defined by the following formula:

Skip to next level

WHEN $\forall i\, Q_i.ca=1$ for $i=1$ to cq and

$$\sum_{i=1}^{cq} Q_i.tt <= \sum_{i=1}^{cq} Q_i.et \text{ and } Q_{cq+1}.op=1$$

where cq is the current question.

For example of trail Figure 2, the student was able to skip question *q3*, because it was optional, and prior to that he had answered all questions correctly and the time taken (5s+30s=35s) was less than cumulative expected time (30s+30s=60s).

Note that for the edge with label *nextQuestion*, we associate the key *autoSkip (as)* to indicate optional questions were skipped. For example of trail in Figure 2, the edge *q2-nextQuestion*→*q4*, the property *as=1* is associated. In addition, student gets credit for auto skipped questions.

In addition to *auto_skip* feature, we also support *forced_exit* feature. This feature is used to terminate the student session if he is performing poorly. Specifically, if the total time taken for questions so far exceeds twice the cumulative expected time, we terminate the student trail.

This *forced_exit* criterion can be defined by the following formula:

Stop the lesson

WHEN

$$\sum_{i=1}^{cq} Q_i.tt >= 2 \times \sum_{i=1}^{cq} Q_i.et$$

where cq is the current question.

Also, we support *self_exit* feature, where the student can himself terminate the trail. Note that the *auto_skip* and *forced_exit* criteria can be further refined if needed. The key point is that our choice of storing the trail as an annotated graph, allows us to enforce these behavior easily when student is taking the trail.

2.4 Student Performance on a Math Trail

To keep things simple, raw score for mandatory and optional questions is reported separately. The raw score for mandatory questions can be specified as

$$Score = 100 \times \frac{\sum_{i=1}^{nq} (Q_i.ca \times Q_i.lvl)}{\sum_{i=1}^{nq} Q_i.lvl} \text{ and } \forall i\, Q_i.op=0$$

where nq is number of questions in the trail. For the trail shown in Figure 2,

$$Score = 100 \times \frac{((1 \times 1+1 \times 1)+(1 \times 2+1 \times 2)+(1 \times 3))}{(1+1+2+2+3)}$$

$$=100\%$$

Similar formula is used to report score for optional questions based upon questions attempted or credited because of *auto_skip* feature. Also, total points contributed by auto skipped questions will be maintained separately. At present the wrong attempts are not used in computing raw score. Our goal is to encourage students to try and get answers right. The wrong attempts, and time take for each question, are provided separately.

2.5 Comparing Math Trails

We support comparing performances on same trail. The comparison becomes interesting because of occurrences of *auto_skip*, *forced_exit*, or *self_exit* due to which the two performances may differ in the questions visited.

Raw scores for each trail is compared using formula presented in Section 2.4. For the actual trail comparison, we show wrong attempts, and time taken for *all* the questions (the union of questions visited by the two trails). Our approach allows for comparing two trails even when the questions were visited in different order. However, we assume the order of questions can only be changed within questions of same level.

2.6 Computing Student Participation

Since student performance for each trail is stored as a separate graph, the overall student participation is readily available. The student or teacher can examine his performance by topic. Specifically, the *topic tree annotated with trails* undertaken by student capture the student participation. For a topic trail, a single trail performance can be examined, or two trails can be compared to see his progress over time.

3. DESIGN

This section discusses the design and implementation of applications.

3.1 Overview of Application

The application is made up of three sub-applications, which operate on a single consolidated backend database (Figure 3):

- *Video Repository*: This application manages the instructional videos, which are downloaded from Internet (for example, Khan Academy [7]) organized under a *topic hierarchy*. It also links to assessment module that can be used as a follow-up after watching a video on a particular topic.
- *Assessment Module*: This application handles all aspects of assessment including maintaining a topic hierarchy, adding different types of questions based on a question template repository, playing questions, and recording and evaluating student's performance (math trails). In addition, it also includes a recommender to recommend relevant videos.
- *Supplementary School Program Info*: This application handles the supplementary school program information, including student enrolment, and class composition in terms of videos screened and the assessments undertaken by students.

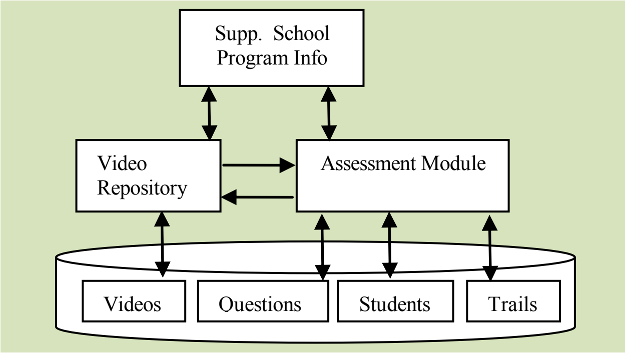


Figure 3. An overview of applications.

3.2 Database Schema

The key tables are shown in Figure 4. A student can have many trails and each trail itself is represented as graph of nodes and edges. A node in the trail refers to a topic or a question. Trail is for a particular leaf topic. Each topic has many questions, and videos. In present application, we currently have one-to-one relation from `M_QDB_TOPICS` to `T_QDB_TRAILS`. However, in future, we plan to extend this to one-to-many relationship that will allow a single leaf topic to have one or more trails. Thus, the student can select one of the trails available for the particular topic.

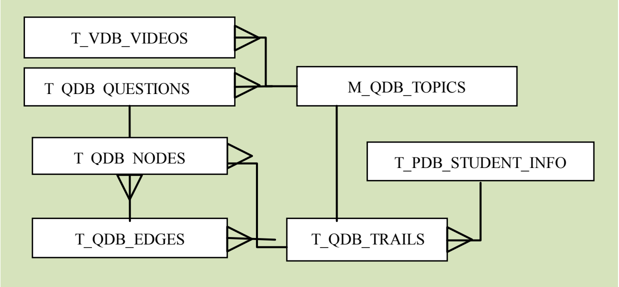
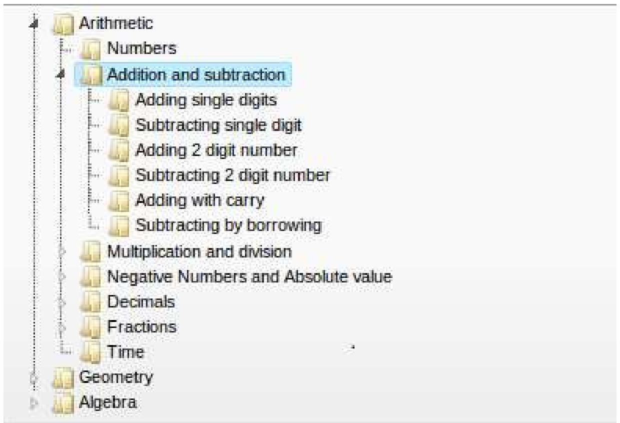


Figure 4. Database Schema (Key tables).

We have used the approach of creating custom property graph tables. That is, instead of generic nodes and edge tables, we use custom tables (`T_QDB_NODES` and `T_QDB_EDGES`), where the node/edge properties are represented as additional columns in node/edge tables. Specifically, `T_QDB_NODES` includes columns for representing *nm*, *lvl*, *opt*, *et, wa, ca, tt,* and *tid* properties. Similarly, `T_QDB_EDGES` includes a column for representing *autoSkip* property. This approach leads to efficient implementation.

3.3 Topic Hierarchy

Topics data is collected from Khan Academy [7] and are ordered under main topics Arithmetic, Geometry, Algebra (Figure 5). There are 32, 36, and 62 leaf subtopics respectively under these main topics. These are arranged using APEX Tree object in the order of their level. Users can take assessment in those topics which are leaf nodes.



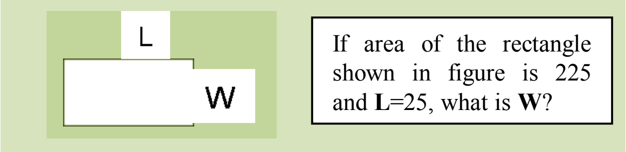
- Arithmetic
  - Numbers
  - Addition and subtraction
    - Adding single digits
    - Subtracting single digit
    - Adding 2 digit number
    - Subtracting 2 digit number
    - Adding with carry
    - Subtracting by borrowing
  - Multiplication and division
  - Negative Numbers and Absolute value
  - Decimals
  - Fractions
  - Time
- Geometry
- Algebra

Figure 5. A portion of topic hierarchy

3.4 Topic Question Template and Question Repository

Specifically there are four types of question templates:

1. *Single Answer*: Solution to the question is a number or a fraction.
2. *Multiple Choice single correct answer*: A problem statement and four options will be given to the user out of which only one option is correct.
3. *Multiple Choice multiple correct answers*: A problem statement and four options will be given to the user out of which there may be two or more correct options
4. *True or False:* A statement would be given and the user has to identify whether it is true or false.



If area of the rectangle shown in figure is 225 and **L**=25, what is **W**?

Figure 6. Supporting questions with figure.

3.5 Student Session and Trails

Each student signs-up prior to starting the trail, browses the topic hierarchy and makes a selection, which determines the start of the trail.

The trail itself is represented as a graph involving topic and question nodes. The question nodes are organized in increasing level of difficulty. Also, for each level, optional questions are maintained towards the end, which may be automatically skipped, if the student is performing well. Figure 2 represents one sample trail.

3.6 Playing Questions and Recording Trails

The algorithm for playing questions and recording trail is presented below. In the algorithm, step 1 initializes Cumulative Correct Answer (*cca*), Cumulative Wrong Answer (*cwa*), Cumulative Time Taken (*ctt*) and Cumulative Expected Time (*cet*) to `0`.

Algorithm : Play Questions in Trail

```
Input: Leaf Topic -- Chosen Topic Trail
1. Initialize a New Trail and cumulative values
   cca:=0; cwa:=0; ctt:=0; cet:=0;
2. Insert nodes of type 'TOPICS' from Root to Leaf Topic;
3. Insert edges for the nodes in topic path;
-- Start assessment
4. Insert node of type 'QUESTION' $Q_1$; cq:=1;
5. Update meta-data (nm, op, et) of question $Q_{cq}$;
6. Insert edge from Leaf Topic node to question node;
-- Attempting question
7. WHILE cq>0 LOOP – more questions –
     autoSkip:=False;
7.1 IF (correct_answer=True) THEN
       $Q_{cq}$.ca:=1, $Q_{cq}$.wa:=0, $Q_{cq}$.tt:=time taken;
       cca:=cca+1;ctt:=ctt+$Q_{cq}$.tt;cet:=cet+$Q_{cq}$.et;
       IF (cwa=0 AND ctt < cet AND $Q_{cq+1}$.op=1) THEN
         autoSkip:=True;
         cq=nq -- nq is firstQ in next level or 0 (if no more)
       ELSE – go to nextQ in sequence or 0 (if no more)
         cq:=cq+1;
       END IF;
       nextQ:=True;
     ELSE -- wrong answer
       $Q_{cq}$.wa:=$Q_{cq}$.wa+1; cwa:=cwa+1; nextQ:=False;
       IF (skip_question_by_user) THEN
         $Q_{cq}$.tt:=time taken;
         ctt:=ctt+$Q_{cq}$.tt;cet:=cet+$Q_{cq}$.et;
         – go to nextQ in sequence or 0 (if no more)
         cq:=cq+1; nextQ:=True;
       END IF;
     END IF;
     – too slow (forced_exit) or done assessment
7.2 IF (ctt > 2 x cet) OR done_assessment=True THEN
        EXIT;
     END IF;
7.3 IF (cq>0) and nextQ=True THEN
     – next question exists
         Insert node of type 'QUESTION' $Q_{cq}$;
         Update properties (nm, op, et) of question $Q_{cq}$;
         Insert edge from previous question node to $Q_{cq}$;
        IF (autoSkip=True) THEN
          Set current edge property as to 1;
        END IF;
     END IF;
   END LOOP;
8. Set Record trail end time;
```

In step 2, student trail gets initiated with root topic to leaf topic path, and in step 3, corresponding edges are created. As assessment starts, node type and current question (*cq*) is set to `'QUESTION'` and `1` respectively and Q_{cq} node properties (*nm, op, et*) are updated to respective values from the question node of default trail (steps 4-5). Step 6 connects the leaf topic node to the question node.

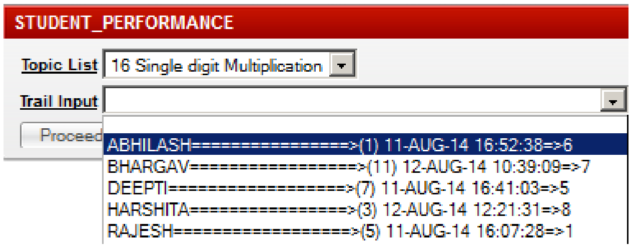
Steps 7 iterates over each question, which is presented to the student in the trail. In step 7.1, the question is attempted and based on student's performance (correct or wrong answer) the relevant question node keys are updated, along with the cumulative measures (*ctt, cet, cwa, cca*). For correct answer the *auto_skip* condition is also evaluated and if met the next question is set to the first question in the next level.

In step 7.2, the *forced_exit* condition is met or it is the case of *self_exit*, the trail is terminated. In step 7.3, as student moves to next question, a new question node is created with default values for properties, and an edge is created to connect to prior question node. If *auto_skip* feature is exercised then the edge property *as* is set to 1. Step 8 records the end of the trail.

3.7 A Student Trail Summary and Details

Presenting student performance translates to traversing the recorded property graph for the trail and printing its properties. Students can review their performance by selecting the topic name in Topic list (Figure 7).

Trail Summary. For the selected trail, it shows student id, student name, topic name, trail id (*Trail)*, trail date (*T Date*), total time taken (*T Time*) by student, total expected time of that trail ((*E Time*), raw score of mandatory *(Raw score M Q)* and optional questions (*Raw score O Q*), and percentage.



Topic List: 16 Single digit Multiplication

ABHILASH================>(1) 11-AUG-14 16:52:38=>6
BHARGAV=================>(11) 12-AUG-14 10:39:09=>7
DEEPTI==================>(7) 11-AUG-14 16:41:03=>5
HARSHITA================>(3) 12-AUG-14 12:21:31=>8
RAJESH==================>(5) 11-AUG-14 16:07:28=>1

Figure 7. Trail selection.

The summary (Figure 8) is shown using a dynamically created HTML region followed by a SQL report.

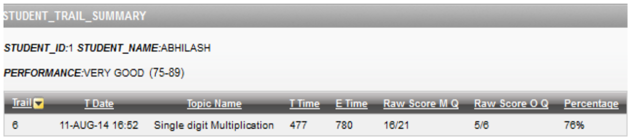
```
STUDENT_TRAIL_SUMMARY:
htp.p ('<p>'||'<em><b>STUDENT_ID:</b></em>'
            || V_STUDENT_ID||' '
            || '<em><b>STUDENT_NAME:</b></em>'
            || V_STUDENT_NAME||'</p>');
PRC:=nvl(PRC, (N_R1/N_TQ)*100);
IF (PRC<25) THEN
htp.p('<p>'||'<em><b>TOPIC_NAME:</b></em>'
           ||V_TOPIC_NAME||' '
           ||'<em><b>PERFORMANCE:</b></em>'
           ||'BAD'||'</p>');
END IF;
...
```

The `STUDENT_TRAIL_SUMMARY` contains the SQL statement which generates the report shown in Figure 8:

```
SQL_STMT:= 'SELECT '||P_TRAIL_ID||' TRAIL, '''
 || S_DATE||''' T_DATE, '
 || V_CUMULATIVE_TT ||' T_TIME, '
 || V_CUMULATIVE_ET|| ' E_TIME, '''
 || N_R1||'/'||N_TQ||''' RAW_SCORE_M_Q, '''
 || N_O||'/'||N_TQ_O||''' RAW_SCORE_O_Q, '''
 || ROUND(PRC)||'%'||'''PERCENTAGE '
 ||' FROM DUAL ';
```

This function will be called in report as shown below:

```
  DECLARE
  SQL_STMT VARCHAR2(1000);
  BEGIN
   SQL_STMT:=STUDENT_TRAIL_SUMMARY(:P8_TRAIL);
   RETURN(SQL_STMT);
 END;
```


STUDENT_TRAIL_SUMMARY

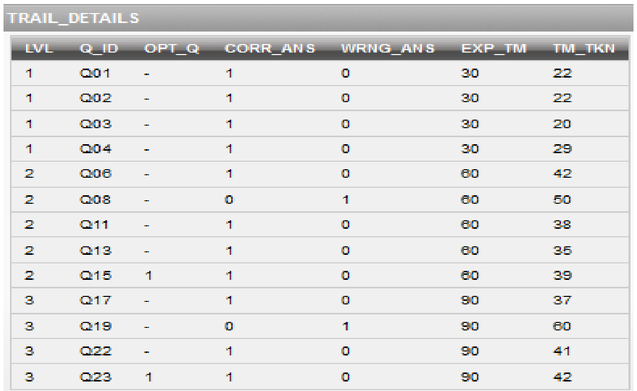
STUDENT_ID:1 ***STUDENT_NAME:***ABHILASH

PERFORMANCE:VERY GOOD (75-89)

Trail	T Date	Topic Name	T Time	E Time	Raw Score M Q	Raw Score O Q	Percentage
6	11-AUG-14 16:52	Single digit Multiplication	477	780	16/21	5/6	76%

Figure 8. Trail summary.

Trail Details. This report (Figure 9) shows the details of the selected trail by querying `T_QDB_TRAILS` and `T_QDB_NODES`. It shows level of question (*LVL*), question id (*Q ID*), optional question (*OPT Q*), correct answer count (*CORR ANS*), wrong answer count (*WRNG ANS*), expected time for question (*EXP TM*), and time taken by student (*TM TKN*).


TRAIL_DETAILS

LVL	Q_ID	OPT_Q	CORR_ANS	WRNG_ANS	EXP_TM	TM_TKN
1	Q01	-	1	0	30	22
1	Q02	-	1	0	30	22
1	Q03	-	1	0	30	20
1	Q04	-	1	0	30	29
2	Q06	-	1	0	60	42
2	Q08	-	0	1	60	50
2	Q11	-	1	0	60	38
2	Q13	-	1	0	60	35
2	Q15	1	1	0	60	39
3	Q17	-	1	0	90	37
3	Q19	-	0	1	90	60
3	Q22	-	1	0	90	41
3	Q23	1	1	0	90	42

Figure 9. Trail details.

Correct and Wrong Answer Comparison Chart. An APEX Flash Chart region is created to compare correct and wrong answer counts, shown in blue and maroon respectively (Figure 10). For the trail, student got 11 correct and 2 wrong. The chart data was generated using a SQL query shown below. For computing cumulative counts, Oracle's built-in Analytic function SUM is used.

```
SELECT null link, rownum label,
  SUM(nd_q_ca) OVER(ORDER BY nd_q_id) value1
FROM(
 SELECT loj.nd_q_id, t.nd_q_ca FROM(
 (SELECT  nd_q_id
  FROM  "VAP"."T_QDB_NODES"
  WHERE nd_name='QUESTION' AND
        nd_trail_id=:P8_TRAIL) loj
 LEFT OUTER JOIN
 (SELECT  nd_q_id, nd_q_ca
  FROM  "VAP"."T_QDB_NODES"
  WHERE nd_name='QUESTION' AND
        nd_trail_id=:P8_TRAIL) t
  ON loj.nd_q_id=t.nd_q_id)
  UNION ALL
  SELECT  0,0 FROM dual
ORDER BY nd_q_id)
```

Post-Trail Report. This report is shown after completion of a topic trail. Student can end the trail normally, or leave in the middle, or have a *forced_exit*. This report is similar to the one shown in Figure 8. We have used the same function `STUDENT_TRAIL_SUMMARY` to generate this report.

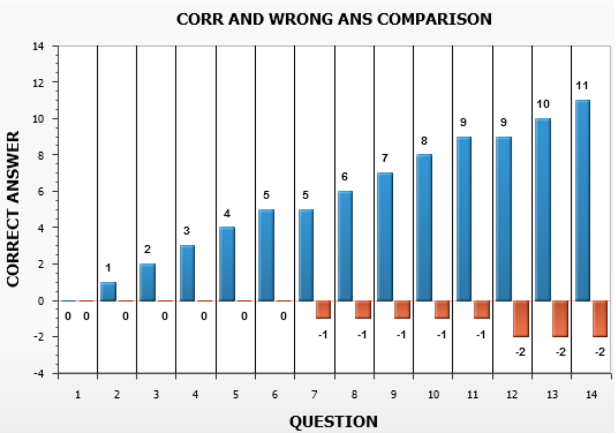
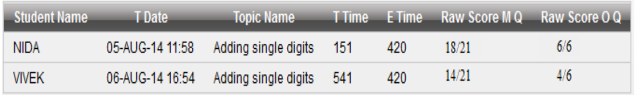


Figure 10. Correct and wrong answer comparison.

3.8 Comparing Trails

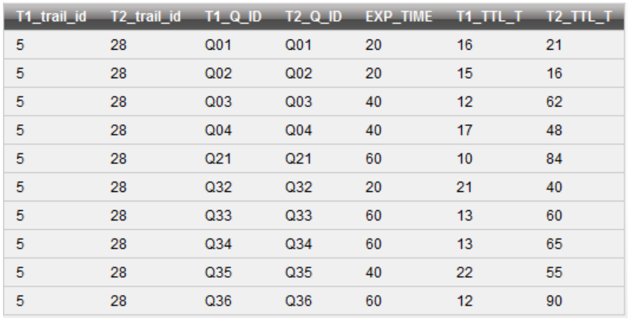
Compare trail translates to comparing the respective property graphs for the trails. Two trails on the same topic belonging either to a single student taken at different times or to two students can be compared. Input for the trails comparison is *trail_ids* of two different trails and output will show up as summary (Figure 11) and detail region (Figure 12).



Student Name	T Date	Topic Name	T Time	E Time	Raw Score M Q	Raw Score O Q
NIDA	05-AUG-14 11:58	Adding single digits	151	420	18/21	6/6
VIVEK	06-AUG-14 16:54	Adding single digits	541	420	14/21	4/6

Figure 11. Trail comparison (Summary).

The summary report compares the two trail details. Cumulative expected time may not be same as one of the students can exit the trail in middle or he is able to skip questions due to *auto_skip* feature.



T1_trail_id	T2_trail_id	T1_Q_ID	T2_Q_ID	EXP_TIME	T1_TTL_T	T2_TTL_T
5	28	Q01	Q01	20	16	21
5	28	Q02	Q02	20	15	16
5	28	Q03	Q03	40	12	62
5	28	Q04	Q04	40	17	48
5	28	Q21	Q21	60	10	84
5	28	Q32	Q32	20	21	40
5	28	Q33	Q33	60	13	60
5	28	Q34	Q34	60	13	65
5	28	Q35	Q35	40	22	55
5	28	Q36	Q36	60	12	90

Figure 12. Trail comparison (Detail).

The report region is formed by dynamically generating SQL statement using function mentioned below. This function is similar to the function `STUDENT_TRAIL_SUMMARY` described in Section 3.7. Instead of creating single row of summary it is creating 2 rows of summary report taking 2 *trail_ids* as inputs.

```
COMPARE_TRAIL_SUMMARY (p_trail_id NUMBER,
                           q_trail_id NUMBER)
RETURN VARCHAR2
```

The detailed trails comparison Report (Figure 12) presents the individual nodes of the trails, representing number of questions attempted, whether question is answered correct or wrong, and the time taken. This helps us understand the performance of the student in comparison with another student or with a prior trail taken by him.

This report is generated by using left outer join to connect node values for the two trails, union operator to have total questions attempted in both the trails.

The graph (Figure 13) presents cumulative expected time (shown in Green) for the two trails along with the individual cumulative time taken to complete trail by each student for his attempted questions. In this graph, Nida's trail is shown in Blue and Vivek's trail is shown in Red.

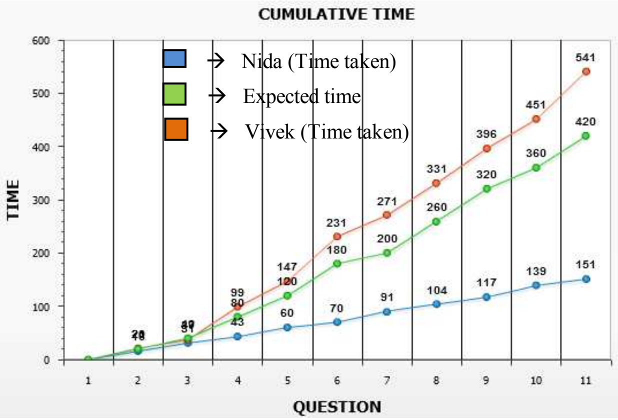


Figure 13. Cumulative time graph.

The cumulative expected time is computed by SQL query shown below.

```
SELECT null link, ROWNUM label,
    sum(ND_Q_ET) over (ORDER BY ND_Q_ID) value1
   FROM
  ((
   (SELECT ND_Q_ET , ND_Q_ID
    FROM "VAP"."T_QDB_NODES"
   WHERE ND_NAME='QUESTION'AND
         ND_TRAIL_ID=:P5_TRAIL_INPUT_1
   UNION
   SELECT ND_Q_ET , ND_Q_ID
   FROM "VAP"."T_QDB_NODES"
   WHERE ND_NAME ='QUESTION' AND
         ND_TRAIL_ID =:P5_TRAIL_INPUT_2)
   UNION ALL
         SELECT  0,0 FROM dual)
   ORDER BY ND_Q_ID)
```

The cumulative expected time is formed by adding expected time of the each node using Oracle's analytic function `SUM`, which are ordered by `nd_q_id`. Similarly, we get trails lines by `'LEFT OUTER JOIN'` join of cumulative expected time query with cumulative total time query of individual trails, which is combined with `'SELECT  0,0 FROM dual'` to start the trail from zero.

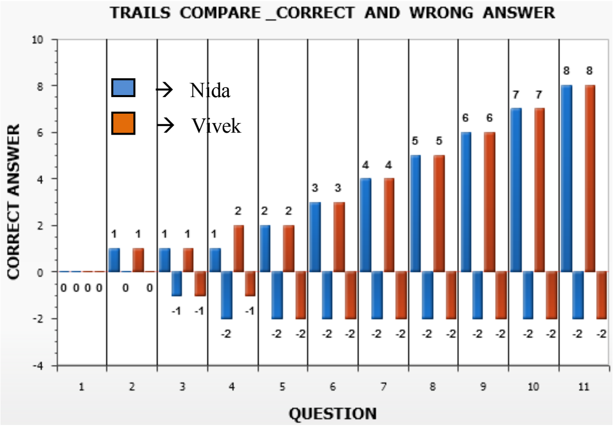
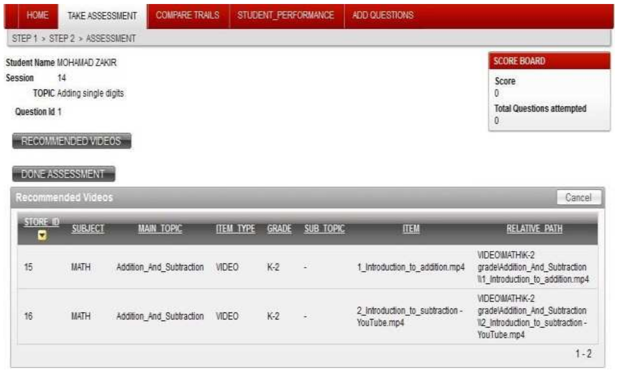


Figure 14. Cumulative correct and wrong answer counts.

Similarly, we have a bar graph representing the difference in two trails with respect to number of correct and wrong attempts (Figure 14).

3.9 Recommending Videos

Video recommender in the *Assessment Module* acts like a helper for the student taking assessment. On clicking `‘Recommended Videos’` button, a report of videos relevant to the current topic is shown (Figure15).



Student Name MOHAMAD ZAKIR
Session 14
TOPIC Adding single digits
Question Id 1

SCORE BOARD
Score
0
Total Questions attempted
0

Recommended Videos

STORE_ID	SUBJECT	MAIN_TOPIC	ITEM_TYPE	GRADE	SUB_TOPIC	ITEM	RELATIVE_PATH
15	MATH	Addition_And_Subtraction	VIDEO	K-2	-	1_Introduction_to_addition.mp4	VIDEO\MATH\K-2 grade\Addition_And_Subtraction \1_Introduction_to_addition.mp4
16	MATH	Addition_And_Subtraction	VIDEO	K-2	-	2_Introduction_to_subtraction - YouTube.mp4	VIDEO\MATH\K-2 grade\Addition_And_Subtraction \2_Introduction_to_subtraction - YouTube.mp4

1 - 2

Figure 15. Recommended videos for a topic.

This report is produced using the function:

```
recommend_video(f_topic VARCHAR2)
RETURN VARCHAR2
IS
sql_stmt VARCHAR2(512);
WORDS_ARRAY APEX_APPLICATION_GLOBAL.VC_ARR2;
MYWORDS VARCHAR2(256);
BEGIN
sql_stmt:='SELECT Store_id S_id,
       Subject Sub,
       Main_topic M_topic,
       Item_type I_type,
       Grade,
       Sub_topic S_topic,
       Item,
       Link RELATIVE_PATH
       FROM T_VDB_VIDEO_STORE WHERE ';
WORDS_ARRAY := APEX_UTIL.STRING_TO_TABLE(f_topic,'
');
FOR z IN 1..WORDS_ARRAY.count LOOP
  MYWORDS:='';
  FOR x IN 1..z LOOP
    MYWORDS:= MYWORDS||WORDS_ARRAY(x)||' ';
END LOOP;
IF z< WORDS_ARRAY.count THEN
  sql_stmt:=sql_stmt||
'UTL_MATCH.JARO_WINKLER_SIMILARITY(Upper(main_topi
c), upper('''||MYWORDS||'''))> 75 OR  ';
ELSE
  sql_stmt:=sql_stmt||
'UTL_MATCH.JARO_WINKLER_SIMILARITY(Upper(main_topi
c), upper('''||MYWORDS||'''))> 75';
END IF;
END LOOP;
RETURN(sql_stmt);
END;
```

where `f_topic` is the particular topic name.

In the report region of recommended videos the ‘Link’ column is not shown which represents the HTML hyper link for the individual ITEM of the respective topic. The link is formed by concatenating ‘Relative path’ to the Application Item `‘F106_VROOT’` from `Video Repository` which is the root path of the video store.

When the button `RECOMMENDED VIDEOS` is clicked a confirm message `‘Do you want to end the active trail? ’` is presented. On confirmation, the application will terminate the trail and present recommended videos. The topic name of the assessment is compared to the similar topic present in the *Video Repository*, which is similar to searching for a relevant video for a topic. Since topics may not completely be similar, we consider the topic as a phrase (`‘WORDS_ARRAY’`) and expand it into multiple phrases (`‘MYWORDS’`) formed by variation of keywords.

These generated phrases `‘MYWORDS’` are used in approximate string comparison with `‘Main topic’` of `Video Repository` table. The approximate string comparison via *Jero-Winkler* metrics helps reduce the *false negatives*.

4. A TOUR OF pg-MathTrails

This section describes the features and functioning of *pg-MathTrails* assessment module.

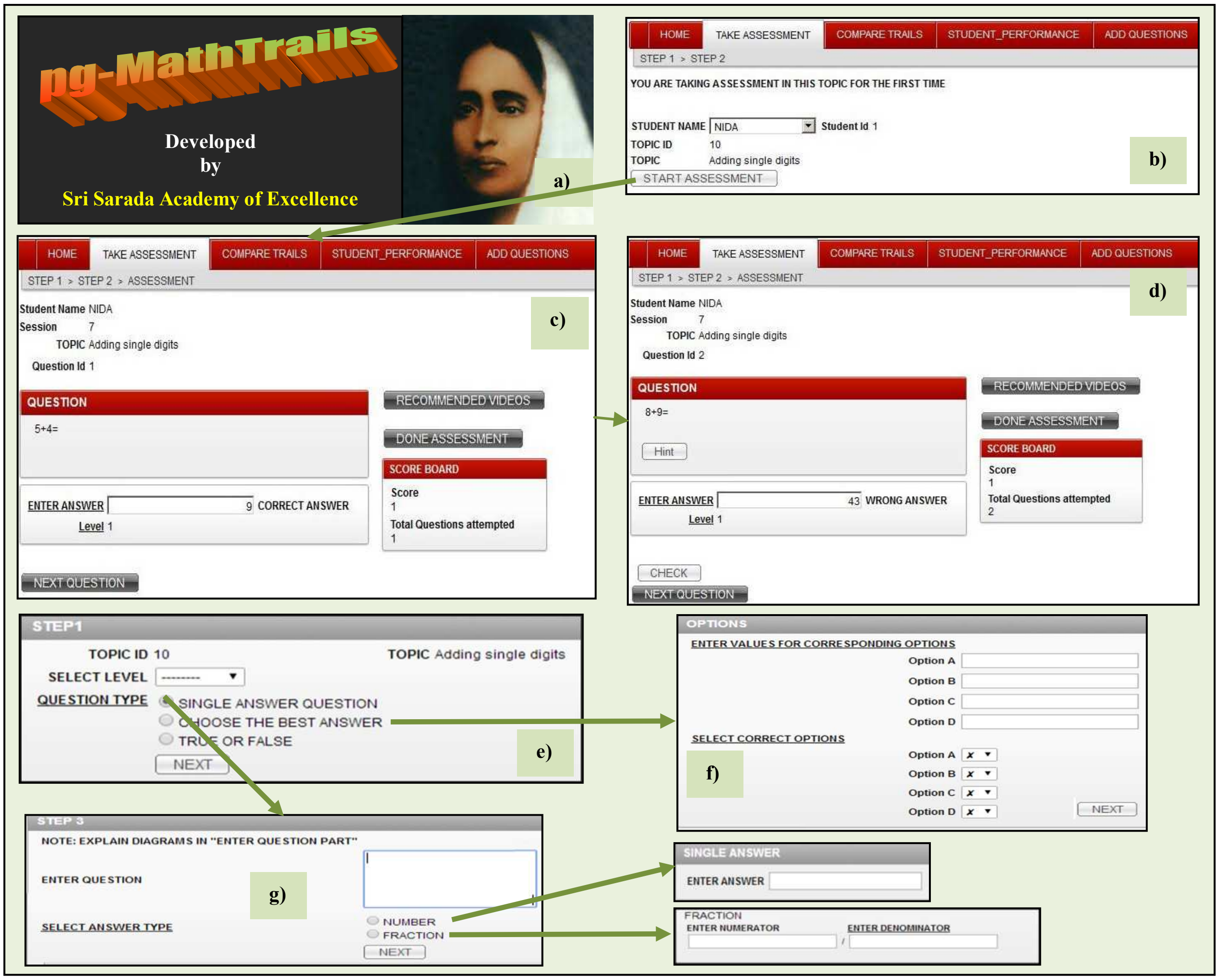
4.1 Take Assessment

This module allows user to login, select a topic, and take an assessment (Figure 16-b and c). On entering the answer, the answer is checked, and the score is updated, and user can move to the next question.

On the next question, if the user makes a mistake the system displays ‘wrong answer’ as well as an option to click the ‘hint’ (Figure 16-d). The scoreboard is updated to include the wrong attempt.

4.2 Student Performance and Compare Trails

The student performance and compare trails have already been covered in Section 3, where individual student performance can be examined as well as two performances of same or different students on a topic trail can be compared.



pg-MathTrails

Developed
by
Sri Sarada Academy of Excellence

STEP 1 > STEP 2

YOU ARE TAKING ASSESSMENT IN THIS TOPIC FOR THE FIRST TIME

STUDENT NAME NIDA Student Id 1
TOPIC ID 10
TOPIC Adding single digits
START ASSESSMENT

STEP 1 > STEP 2 > ASSESSMENT

Student Name NIDA
Session 7
TOPIC Adding single digits
Question Id 1

QUESTION
5+4=

ENTER ANSWER 9 CORRECT ANSWER
Level 1

SCORE BOARD
Score
1
Total Questions attempted
1

STEP 1 > STEP 2 > ASSESSMENT

Student Name NIDA
Session 7
TOPIC Adding single digits
Question Id 2

QUESTION
8+9=

ENTER ANSWER 43 WRONG ANSWER
Level 1

SCORE BOARD
Score
1
Total Questions attempted
2

STEP1

TOPIC ID 10
TOPIC Adding single digits
SELECT LEVEL --------
QUESTION TYPE
○ SINGLE ANSWER QUESTION
○ CHOOSE THE BEST ANSWER
○ TRUE OR FALSE

OPTIONS

ENTER VALUES FOR CORRESPONDING OPTIONS
Option A
Option B
Option C
Option D

SELECT CORRECT OPTIONS
Option A x
Option B x
Option C x
Option D x

STEP 3

NOTE: EXPLAIN DIAGRAMS IN "ENTER QUESTION PART"

ENTER QUESTION

SELECT ANSWER TYPE
○ NUMBER
○ FRACTION

SINGLE ANSWER
ENTER ANSWER

FRACTION
ENTER NUMERATOR / ENTER DENOMINATOR

Figure 16. ***pg-MathTrails*** **a) Home Page b) Start Assessment, c,d) Play Question, e to g) Add Questions.**

4.3 Add Questions

This functionality is meant for teachers or instructors to add questions. (Figure 16-e) shows adding a question with single answer and multiple answer choices. For single answer (Figure 16-g), we provide the option of entering a single number or a fraction specified through two number fields.

For multiple choice questions (Figure 16-f), we allow user to specify the choices as answer. Note for the multiple choice question, we support both single choice and multiple choices as correct answer. The true or false type question is similar to a multiple choice question, except that user has to only specify either the true or false.

5. EXPERIENCE

This section describes our experience of using this application.

5.1 Usability Study

Objective. The objective of the Usability Study was twofold: i) To report the students experience of using the application, and ii) To study the concurrent use of the application by multiple clients.

Setup. Three computers were setup with a wireless LAN, one as the server as well as a client, and two other laptops as clients. We created two topic trails: i) for `'multiplication by single digit'`, and ii) for `'adding negative numbers'`. Both trails had questions categorized into levels 1, 2, 3, and each trail had a total of 12 questions, of which 9 were mandatory and 3 were optional. The trails were taken by children in the after school program of Sri Sarada Academy of Excellence, Bangalore (SSAE). The first and second topic trail online test was administered to 3rd to 6th and 7th to 9th grade students respectively. Individual students from each group took the assessment.

Observations. User interface looked attractive to students and the performance was satisfactory in the wireless LAN setup. Students were very enthusiastic as they never had taken online test like this before. In the first session, students were struggling with mouse and keyboard as they came from underprivileged families having limited exposure to computers.

The concurrent use of application worked correctly (no interference). The response time was reasonably well, meeting the interactive response time limit of 2000 ms.

Challenges faced by students.

- Cursor was not set on the answer field.
- The answers entered by prior students were appearing as the browser form history was getting cached.
- Hint button was reappearing in the next question after correcting the first question. This was a bug.
- 7th and 8th students were struggling in using `('-', '+')` signs as they were taking test on `'adding negative numbers'`.

Overall performance. In the first session (1st August 2014, time: 3.30 to 4.45pm), students from 3rd and 4th grade performed very well as they had taken video based lessons on the `'multiplication'` topic in prior classes. These students attempted all the questions and took less time for questions in the beginning of the trail, and they were rewarded with *auto_skip*, which allowed them to skip optional questions. Students from 5th and 6th grade were little uncomfortable and slow while taking the same trail. This could be attributed to the fact that they were rusty, that is, they had covered this topic a while back. So as those students took more time to attempt initial questions, they were forced to end the trail (via the *forced_exit* feature).

Solution to problems faced.

- We made sure the students became familiar with the system and comfortable with mouse and keyboard.
- We fixed the application to have the cursor focus on the answer field automatically after each question.
- We configured the browser to not to cache form history.
- We fixed the bug, to clear the hint when next question button is pressed.

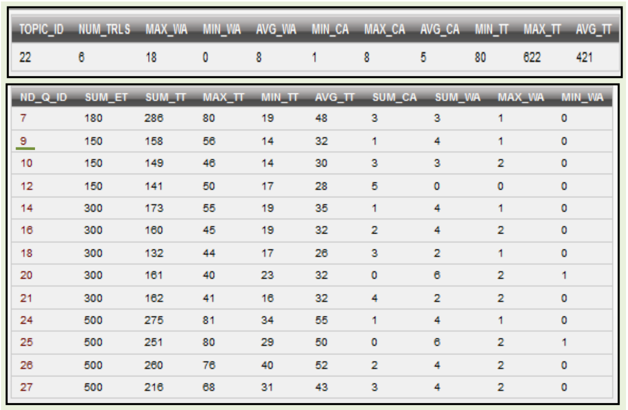
In the second session (7th August 2014, time: 3.30 to 5.00 pm), the students were more confident and comfortable with user interface due to changes made as well as they were more familiar. Similar to first session, two new trails were created for the same two topics. Students performed better when compared to the prior session and as one student got all the mandatory questions correct with no wrong attempts plus was able to skip optional questions.

Challenges faced. We noticed that using *Backspace* button on the keyboard multiple times was moving the user to the previous page and was repeatedly recording multiple similar nodes.

We resolved this problem by configuring the browser to not map *Backspace* key event to browser *Back* event.

The student trail performance was recorded and the application automatically generated `'STATISTICS OF TRAILS'` reports. For inputs, the `TOPIC`, `START TIME`, and `END TIME` of the session, we got the trails statistics (shown in Figure 17 top) representing total trails have been taken by all students, and the count of minimum, maximum, and average correct attempts, wrong attempts, and total time taken.

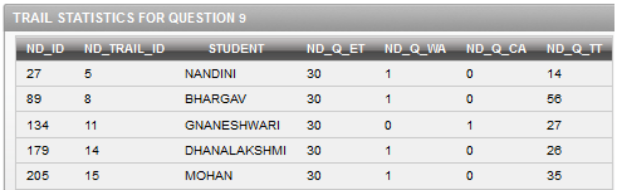
In addition, the application automatically generates the trail statistics by Question (Figure 17 bottom), with the drill-down ability. By clicking any question listed under `ND_Q_ID`, the detailed summary for the selected question is shown (Question 9 shown in Figure 18). The 9th question was attempted only by 5 students. The 6th student did not attempt this question as she exited from the trail after 1st question because of the time constraints.



TOPIC_ID	NUM_TRLS	MAX_WA	MIN_WA	AVG_WA	MIN_CA	MAX_CA	AVG_CA	MIN_TT	MAX_TT	AVG_TT
22	6	18	0	8	1	8	5	80	622	421

ND_Q_ID	SUM_ET	SUM_TT	MAX_TT	MIN_TT	AVG_TT	SUM_CA	SUM_WA	MAX_WA	MIN_WA
7	180	286	80	19	48	3	3	1	0
9	150	158	56	14	32	1	4	1	0
10	150	149	46	14	30	3	3	2	0
12	150	141	50	17	28	5	0	0	0
14	300	173	55	19	35	1	4	1	0
16	300	160	45	19	32	2	4	2	0
18	300	132	44	17	26	3	2	1	0
20	300	161	40	23	32	0	6	2	1
21	300	162	41	16	32	4	2	2	0
24	500	275	81	34	55	1	4	1	0
25	500	251	80	29	50	0	6	2	1
26	500	260	76	40	52	2	4	2	0
27	500	216	68	31	43	3	4	2	0

Figure 17. Overall and by question trail statistics.



TRAIL STATISTICS FOR QUESTION 9

ND_ID	ND_TRAIL_ID	STUDENT	ND_Q_ET	ND_Q_WA	ND_Q_CA	ND_Q_TT
27	5	NANDINI	30	1	0	14
89	8	BHARGAV	30	1	0	56
134	11	GNANESHWARI	30	0	1	27
179	14	DHANALAKSHMI	30	1	0	26
205	15	MOHAN	30	1	0	35

Figure 18. Trails statistics for question 9.

5.2 Additional Comments

Currently, we allow creating a single trail for each leaf topic, which will be extended to support multiple trails. Also, we will present a bird's eye view of the trail using the recorded property graph as the student makes progress, which he can navigate to review and make changes at the end of the trail completion.

Adding question in this application has simple and easy steps. We can update and make changes in questions, and time of the particular question by directly updating underlying tables through APEX table browser. For lower grades, for topics such as 'Numbers', which has very basic questions, it might be difficult to give any hints. For those questions, we may hide the hint option.

Based on our experience, we may provide an option where *forced_exit* and *auto_skip* feature can be turned on or off. Students taking assessment have time constraints. For example, first level question has 30sec time to answer, 2nd level has 60sec, and 3rd level has 90sec. If student spends more time on any question then the *forced_exit* condition can be trigger termination of the trail. In both the class sessions, we found students were unaware of the time constraint and spent longer while answering initial questions and their trail was terminated. So we are thinking of introducing 'starting buffer time' to provide extra time for initial questions.

An important reason for developing *pg-MathTrails* was that it runs in a wireless LAN without requiring Internet. The benefits are that it has interactive response time and there is no instability due to fluctuating speed of Internet or the site hosting the online assessment application. Furthermore, students also stay focused as they are not tempted to browse other Internet sites.

The approach of storing each individual student performance on a trail as a separate graph opens interesting possibilities. For example, this can be used to conduct a Math Challenge contest, where the progress of various participants can be observed live.

Our vision is to expand *pg-MathTrails* to *e-Learning Trails*, which will allow creating learning trails for all subjects. The present application is well suited for Math as answers to Math questions tend to be precise and numeric. However, the current application is not suitable for question such as proving theorem. Despite this we feel Math is a good fit since the concepts and logic of the math theorems gets exercised in solving numeric problems.

Currently, the following question templates are supported 1) `'Single answer'`, 2) `'Choose the best answer'`, and 3) `'True or False'`. These question templates can be used to form questions for all subjects, including Science (Physics, Chemistry, and Biology), Social Science and Languages (English). However, the current set of question templates do not provide direct way to form questions like `'Fill in the blanks'`, `'Match the following'`, `'Odd man out'` and `'Paragraph based group of questions'`. We can handle these using current templates in the following way:

- By using `'Single answer'` template, we can form `'Fill in the blanks'`, example, `1, 2, _, 4` and `'Odd man out'` question, example, `1, 3, 8, 5`.
- Similarly by using `'Choose the best answer'` we can form `'Match the following'` question:

```
1. 2 + 3       a) 10
2. 2 x 5       b) 5
3. 15/5        c) 14
4. 16 - 2      d) 3
Ans: A. (1-a), (2-b), (3-c), (4-d)
     B. (1-b), (2-a), (3-d), (4-c)
     C. (1-b), (3-a), (4-d), (2-c)
     D. (3-b), (4-a), (1-c), (2-d)
```

For questions with numeric answers, checking the answer is easy. However, for question requiring answer in the form of phrase or sentences is difficult to match. These kinds of questions are common in subjects like Science and English. For example,

```
'Sunday is Raju's favourite day of the week. He likes it because on Sunday, he watches football. On other days, he also gets to watch football but not all day.'
```

The question `'Which is Raju's favourite day?'` can be answered either as `'Raju's favourite day is Sunday'`, or, `'Sunday'`, or `'Sunday is Raju's favourite day'` making it difficult to match the answer.

6. RELATED WORK

There are quite a few online assessment modules available (Table 1). Khan Academy [7] is one of the popular free educational sites, having good collection of videos and assessments on various topic categorized by grades. We utilize the videos provided by them for our students. LearnNext [13] is somewhat similar to Khan Academy but meant for primary and high school students with Maths and Science videos and online tests. IXL [16] is another good site that covers the complete math syllabus for K-12th grades with good assessment pattern, but charges membership fee. emathematics.net [11] is another good site for students to practice.

Math-Drills [10] is an interesting site, which provides free Math Worksheets on a variety of math topics for classes 1 to 12. However, the sheet has to be downloaded and printed and does not have online testing capability.

Math Trail [12] tests geography and math skill together, where student has to find the location on the map and then proceed to answer the question. The idea of the trail is similar to our application. However, in our application, the topics and the questions form the trail as opposed to real geographic locations.

Table 1: Comparison of Math Assessment Software

SOFTWARE	***Q. in the form of Games***	***Q. in increasing level of difficulty***	***Links to videos***	***Hint for Q.***	***100% Syllabus Coverage***	***Detailed student report***	***Multiple Subjects***	***Free***	***Off line***	***Alexa+ Website rank***
Khan Academy	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	No	**Yes**	**Yes**	No	3,308
IXL	**Yes**	**Probably**	No	No	**Yes**	**Probably**	No	No	No	8,169
Math-Drills	No	No	No	No	**Yes**	No	No	No	**Yes**	93,774
LearnNext	**Yes**	No	**Yes**	No	**Yes**	**Yes**	No	**Yes**	**Yes**	92,118
Math Trail	**Yes**	No	No	No	No	No	**Yes**	No	No	1,006,101
emathematics.net	No	No	No	No	**Probably**	No	No	**Yes**	No	1,892,116
pg-MathTrails*	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	No	**Yes**	**Yes**	-

**auto_skip*, *forced_exit*, and comparing two trails are supported in *pg-mathTrails* only.
+Website rank tends to change. The current ranks reported are based on 6th October 2014.

Also, our application differs in that it is built on property graphs, which enable in-depth analysis of student's performance on the trail, allows comparing two trails, and provides overall student participation. Plus, our application supports features such as *auto_skip*, or *forced_exit*, which we have not seen in these sites mentioned above.

Property graphs [2] have recently become popular for handling large scale connected graph like data. Property graphs are supported by graph databases such as Neo4j [3], Sparksee [4] and Infinite Graph [14]. Graph databases are used to build social networking and scientific applications [9]. Companies such as Facebook, and Twitter have developed their own graph databases social graph and interest graph respectively.

We are using a custom property graph to store the default trail as well as student trails in our Math assessment module. Our approach is similar to defining typed graph classes to enable a statically type query language for property graphs [8]. That is, instead of defining a generic structure such as Entity-Attribute-Values (EAV) [15] to hold properties of nodes and edges, we define custom node and edge tables where properties are implicitly stored in additional columns.

7. CONCLUSIONS AND FUTURE WORK

The paper presented *pg-MathTrails*, an Math student assessment module built using property graphs. The distinguishing aspect of this application is to store topic trail as well as student performance on the trail as individual graphs that are readily available for performing in-depth analysis of student performance on individual trails, as well as for comparing two trails of the same student or two students. Instructors can create topic trails by making use of pre-existing question templates. It also features a video recommender that allows linking with relevant videos in a video repository.

The application has been tried out in an after school program with positive feedback. The application also features *auto_skip* and *forced_exit* capabilities to make the math trail interesting and challenging for students.

In future, we plan to handle evolution of a trail pertaining to a particular topic, and support 'Suspend and Resume' option for a trail, which will be useful for long trails. We also plan to extend the tool to support e-learning trails on other subjects as well.

Acknowledgments: We thank Khan Academy [7] for having such a wonderful site with a huge collection of instructional videos and making them available free to download. We thank the children in the after school program of Sri Sarada Academy of Excellence, Bangalore for participating in the usability study. We thank Vani S Kambi for proofreading the paper, and Priya Das of Sarada Research Labs, Bangalore for testing the application. We thank all reviewers for their useful suggestions.

8. REFERENCES

[1] WHOI Fitness Trail, http://www.whoi.edu/generalinfo/fittrail

[2] Property Graph Model · tinkerpop/blueprints Wiki · GitHub, https://github.com/tinkerpop/blueprints/wiki/Property-Graph-Model

[3] Neo4j Graph Database, www.neo4j.com

[4] Sparsity Technologies -- Powering Extreme Data, http://www.sparsity-technologies.com

[5] Oracle Application Express, www.oracle.com/technetwork/developer-tools/apex

[6] Oracle Database Express Edition 11g Release 2, www.oracle.com/technetwork/database/express-edition

[7] Khan Academy, https://www.khanacademy.org

[8] Tausch, N., Philippsen, M.,Adersberger, J.: A statically typed query language for property graphs. IDEAS 2011: 219-225

[9] Buerli , M., The current state of graph databases, 2012 http://www.cs.utexas.edu/~cannata/dbms/Class%20Notes/09%20Graph_Databases_Survey.pdf

[10] Math-Drills. www.math-drills.com

[11] emathematics.net, www.emathematics.net

[12] Math Trail. mathtrail.heymath.com

[13] LearnNext, www.learnnext.com

[14] Objective, www.objectivity.com/infinitegraph

[15] Nadkarni, P. M., et al., Organization of Heterogeneous Scientific Data Using the EAV/CR Representation. JAMIA 6(6): 478-493 (1999)

[16] IXL, http://www.ixl.com

POSTER PRESENTATIONS

Event Processing across Edge and the Cloud for Internet of Things Applications

Nithyashri Govindarajan[1], Yogesh Simmhan[2], Nitin Jamadagni[3] and Prasant Misra[4]
[1]Birla Institute of Technology and Science, Pilani
[2]Supercomputer Education and Research Centre, Indian Institute of Science, Bangalore
[3]National Institute of Technology, Surathkal
[4]Robert Bosch Centre for Cyber Physical Systems, Indian Institute of Science, Bangalore
f2011746@pilani.bits-pilani.ac.in[1], simmhan@serc.iisc.in[2],
nitin.jamadagni@gmail.com[3], prasant.misra@rbccps.org[4]

ABSTRACT

The rapid growth of sensing devices has opened up complex event processing (CEP) for real-time analytics in Internet of Things (IoT) Applications. While CEP has traditionally been centralized, the increasing capabilities of edge devices like smart phones, and the operational needs of low latency and privacy makes it desirable to use both edge and the Cloud for distributed CEP, the former often serving as event sources. This paper motivates the need for real-time analytics across edge and the Cloud, formalizes an optimization problem for bi-partitioning a CEP query pipeline based on IoT application needs, and proposes an initial solution.

1. INTRODUCTION

The rapid proliferation of sensing elements, both in the physical space (such as RFID tags, sensors, smart phones) and virtual space (crowd-sourced data, social networks, virtual agents) is generating immense volumes of data. Often, these data sources are continuous in nature, contributing to the velocity aspect of Big Data. This is best exemplified by the fast evolving Internet of Things (IoT) domain [1] where 50 billion or more devices are expected to be interconnected through the Internet. Extracting meaningful information from these rapid streams of data requires both existing analytics models, architectures and platforms to be extended, and novel ones to be defined. In this regard, Complex Event Processing (CEP) is a key data processing approach that facilitates easy specification of event patterns, and their fast detection on high traffic event streams, to derive actionable intelligence in real-time [1, 6].

CEP engines accept continuous queries that are defined over event tuples and detect patterns within and across events that arrive on a single or on multiple event streams. They help translate low level information collected from heterogeneous sources at high rates into "complex events" that identify situations of interest [8]. Traditional research and development of CEP engines has been on single-machine, shared memory platforms [5, 7]. There, event sources are often moved centrally into Cloud data centers where integrated processing of CEP queries takes place [2] [3].

In the new generation of IoT applications, the event sources are often on edge devices like wireless sensors, smart power meters, and smart phones, many of which are starting to have non-trivial processing capabilities (e.g., Arduino and Raspberry Pi boards for sensor network gateways, Android Smart Phones). Given the need for rapid decision-making on event pattern detection and the operational constraints , it makes sense to leverage the processing capabilities of the edge devices rather than rely solely on centralized clouds. Such requirements are often seen in healthcare applications and public utilities in Smart Cities [10].

The focus of our work is to enable real-time analytics on distributed systems that span across the edge and the Cloud. As such, there is growing interest on distributed CEP [2]. Like others [3], our aim is to model the distributed query processing of blended streams across dynamic systems, without centralization on the Cloud. Similarly, our goal is to minimize the end-to-end latency for executing the query pipeline by minimizing the overheads in the entire system, and not just on the edge devices or the Cloud.

However, we have several unique aspects. One is the use of a distributed IoT infrastructure with heterogeneous resources with varied computational capabilities for executing CEP pipelines. Secondly, CEP engines available on the diverse resources have varying query expressivity that limits their ability to handle specific query predicates. Lastly, privacy constraints are enforced on event streams. These varied Quality of Service (QoS) requirements and IoT infrastructure conditions are taken into consideration while minimizing the latency for detecting patterns.

In the rest of the paper, we describe an IoT scenario for water analytics and an architecture for distributed CEP across a smart phone platform and the Cloud (§ 2). Further, we motivate the need for intelligent partitioning of CEP query pipelines to meet constraints unique to the IoT domain, and formalize it as an optimization problem (§ 3). We

---

[1] #IoTH: The Internet of Things and Humans, April 16, 2014, http://radar.oreilly.com/2014/04/ioth-the-internet-of-things-and-humans.html

[2] Amazon AWS Kinesis, http://aws.amazon.com/kinesis/

The 20th International Conference on Management of Data (COMAD),
17th-19th Dec 2014 at Hyderabad, India.

propose initial solutions for this optimization problem, using both brute force and dynamic programming approaches (§ 4), and then discuss future extensions to this work (§ 5).

2. CEP ACROSS EDGE AND THE CLOUD

Our motivating scenario is a Campus Smart Water IoT infrastructure where water level, flow and quality sensors (~ 200) are deployed on water tanks, reservoirs and mainline pipes, and connected to the Internet through Android-based wireless gateway devices and smart phones. The sensors emit events periodically ($\frac{1}{sec} \sim \frac{1}{min}$) and report observations such as the water level in the tank, quality parameters such as total dissolved solids, chlorine content, temperature, and the volume of water flow. Analytics using CEP queries are defined usually as a dataflow pipeline where data pre-processing and quality checks precede aggregations or detection of interesting situations, e.g., water overflow and underflow due to pump operations, leakage of water due to input-output flow imbalance, presence of contaminants, and excess usage relative to campus average. Then, notifications are send to operations managers or water consumers for sustainable water usage.

Given the current scale, the CEP queries can be processed either on the edge devices (here-on called *edge*) or centrally on the Cloud without significant performance impact. But as we scale the system, e.g., to a town or city wide deployment, or by including more number and types of sensors, such as air quality and electricity usage, there is a need to intelligently partition the CEP query execution across the edge and the Cloud to meet these requirements. As we consider location-based sensing of consumer activity, data ownership and security become concerns, and data generators may impose restrictions on moving raw data to the Cloud.

As a proof of concept of executing CEP pipelines across edge and the Cloud, we implement a light-weight CEP engine, *CEPLite*, on the Android platform, complemented by a full-featured *Siddhi* CEP engine on the Cloud [9]. Incoming sensor events on the edge device (Android) are first categorized based on their sensor source and placed into distinct input queues of an EventBus [3] Publish-Subscribe (PubSub) system. Subscribers, in this case the CEPLite engine, register a subscription with the PubSub broker and are notified asynchronously when events arrive. The subscribed events are placed in input streams from which CEPLite Processing Units (PU) poll for events. Each PU is responsible for one query submitted to CEPLite, and operates over specific input event streams in a SIMD manner. The PU's query capability is limited to filter, sequence, count windows and simple aggregation, and it forwards the generated output (complex) events over a stream to the next PU or the Cloud having the next CEP query in the pipeline for processing.

Users submit their CEP query pipeline as a dataflow graph (Fig. 1). A query planner distributes this pipeline across the edge and the Cloud based on several factors, as discussed in § 3. These query segments are registered with CEPLite on the edge device, and Siddhi on the Cloud. The communication between the edge device and the Cloud is done using a REST or a CoAP [4] service on the Cloud. CoAP is intended for resource-constrained IoT devices to communicate over the Internet with low overhead, using UDP and optional multicast, and allows integration with traditional HTTP web services.

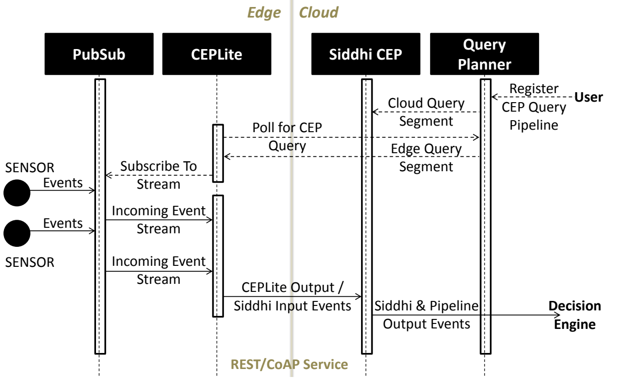


Figure 1: Sequence diagram of interactions between edge and Cloud

3. PROBLEM FORMALIZATION

Preliminaries. The pipeline of queries executing over event streams is represented as a *directed acyclic graph (DAG)* of vertices and edges: $\mathcal{G} = \langle \mathbb{Q}, \mathbb{S} \rangle$. $\mathbb{Q} = \{q_i\}$ is the set of *CEP queries* that are the vertices of the DAG, and $\mathbb{S} = \{s_i \mid s_i = \langle q_j, q_k \rangle,\ q_i, q_j \in \mathbb{Q}\}$, is the set *event streams* that connect the output of query q_i to the input on the next query q_j, and form the *directed* edges of the DAG. Edges indicate the data dependency between the queries, and events produced from the output of a query pass along the stream as input to the next query.

The queries that receive the initial input event streams into the DAG are called the *source queries*, and are characterized by having no incoming edges (though implicitly, an edge indicating the input stream is incident on the queries). The set of source queries is given by:

$$\mathbb{Q}_{\succ} = \{q_{\succ} \mid \nexists\ s_j = \langle q_i, q_{\succ} \rangle \in \mathbb{S}\ \forall\ q_i \in \mathbb{Q}\}$$

The queries that emit the final output event streams from the DAG are called the *sink queries*, and these have no explicit outgoing edges. The set of sink queries is given by:

$$\mathbb{Q}_{\prec} = \{q_{\prec} \mid \nexists\ s_j = \langle q_{\prec}, q_i \rangle \in \mathbb{S}\ \forall\ q_i \in \mathbb{Q}\}$$

For simplicity, we assume in this paper that there can be multiple source queries, i.e., $|\mathbb{Q}_{\succ}| \geq 1$, but there is only one sink query so that the optimization goals of the pipeline are scoped to outputs from this sink.

We define *paths* in the DAG which represent the logical flow of the events through the streams, from the source queries to a unique sink query. Since we assume a single sink query, we denote its path as p. This path offers an alternative, edge-centric view of the DAG. The following production rules generate the path from *path segments*, where $s_\star \in \mathbb{S}$ and the '$\multimap$' symbol indicates a query that connects streams on either of its sides. The path for the sink query is also the *maximal unique path* for the DAG – it has every stream in the DAG appearing exactly once (maximal) and we assert a canonical ordering for comma-separated parallel streams (unique). By definition, the path is acyclic for a DAG.

$$\begin{aligned}
Source &: \varnothing \multimap s_i \\
Sink &: s_i \multimap \varnothing \\
Sequential &: s_i \multimap s_j \\
Parallel &: Predecessor \multimap (Segment, Segment+)
\end{aligned}$$

---

[3] http://greenrobot.github.io/EventBus/

$$\begin{aligned} &\multimap \mathit{Successor} \\ \mathit{Predecessor} &: \mathit{Source} \mid \mathit{Sequential} \mid \mathit{Parallel} \mid \varnothing \\ \mathit{Successor} &: \mathit{Seqential} \mid \mathit{Parallel} \mid \mathit{Sink} \\ \mathit{Segment} &: \mathit{Seqential} \mid \mathit{Parallel} \\ p &= \underset{\mathit{Segment}}{\arg\max} \left(\textsc{Length}(\mathit{Segment})\right) \end{aligned}$$

Queries execute on specific *computing resources* and the set of resources available within the IoT infrastructure is given by $\mathbb{R} = \{r_i\}$. We consider two classes of computing resources, *edge* devices such as smart phones and micro-controllers and *Cloud* virtual machines (VMs), each with a specific computing capability as defined later. The mutually exclusive set of edge devices and Cloud VMs are given by $\mathbb{R}^E$ and $\mathbb{R}^C$, respectively. Thus $\mathbb{R} = \mathbb{R}^E \cup \mathbb{R}^C$ and $\mathbb{R}^E \cap \mathbb{R}^C = \varnothing$. A *resource mapping* function indicates the resource on which a query executes:

$$\mathcal{M} : \mathbb{Q} \rightarrow \mathbb{R}$$

We define a *privacy function*:

$$\mathcal{P} : \langle \mathbb{S}, \mathbb{R} \rangle \rightarrow \{True, False\}$$

that determines if a stream is allowed on a resource. If the privacy for a stream $s_i = \langle q_j, q_k \rangle$ on resource r_l is $False$, this means that neither of the queries q_j or q_k, emitting or consuming the stream, can be executed on resource r_l, i.e.,

$$\mathcal{P}(s_i\langle q_j, q_k\rangle, r_l) = False \implies \mathcal{M}(q_j) \neq r_l \;\&\; \mathcal{M}(q_k) \neq r_l$$

Further we define $\mathbb{A} \subseteq \mathbb{Q}$ as the set of *anonymization queries* such that after a stream is processed by one of them, successive streams along the path have been anonymized and the privacy of such streams on any resource is $True$. For a stream $s_k = \langle q_l, q_m \rangle$ in the path p such that $q_m \in \mathbb{A}$, then $\forall s_j \textit{ downstream of } s_k, \mathcal{P}(s_j, \star) = True$, i.e., the streams after the current one up to the sink have been anonymized and their privacy on any resource is $True$.

A *selectivity* function is associated with each query of the pipeline as a statistical measure of the fraction of incoming events satisfying the query and generated as output events. The *selectivity* is denoted by $\gamma(q)$ and is the average number of output events generated per input event. Output events generated by a query are duplicated on all output streams.

The *stream rate* Ω defines the number of events per unit time on a stream. The *incoming stream rate* entering each input query of the pipeline is denoted by $\Omega_\succ$. Knowing the incoming stream rates and the selectivity per query in the pipeline, the rate at its output stream s_j generated by query q can be calculated recursively as $\Omega(q) = \gamma(q) \times \sum_i \Omega(s_i)$, where s_i are the input streams to query q. This can then be used to estimate the *outgoing stream rate*, $\Omega_\prec$, from the output query for the entire pipeline.

Latency $\mathcal{L}$ is defined as the time for an event to move from one resource to another, over a stream connecting queries present on those resources. Latency is taken to be a constant value for any two resource-pairs at a given maximum event rate Ω^L_{max}. Latency, rather than bandwidth, is often the limiting factor for event processing on commodity networks, and once bandwidth starts becoming the bottleneck, we assume that the pipeline cannot scale on those resources, and, for simplicity, set $\mathcal{L} \rightarrow \infty$. Formally:

$$\begin{aligned} &\mathcal{L}^{\Omega}(r_i, r_j) = x \mid x \in \mathbf{R}^+ \cup 0 && \text{if } \Omega \leq \Omega^L_{max} \;\&\; i \neq j \\ &\mathcal{L}^{\Omega}(r_i, r_j) = \infty && \text{if } \Omega > \Omega^L_{max} \;\&\; i \neq j \\ &\mathcal{L}^{\Omega}(r_i, r_j) = 0 && \text{if } i = j \end{aligned}$$

where $\mathbf{R}^+$ is the set of positive Real Numbers and $r_i, r_j \in \mathbb{R}$.

CEP engine implementations may be limited by the resource platform. If a CEP query (i.e., all its predicates) can be evaluated by an engine on a resource, then the resource's *expressivity* is said to exist for the query. If a query is not supported on a resource, then the expressivity is absent.

Like latency, the *compute time* $\mathcal{C}$ for a query is a constant for a given query q_i on a resource r_j for all $\Omega \leq \Omega^C_{max}$. This *max* value can be empirically obtained. Beyond this threshold, the compute resources get saturated and the query cannot scale. Since events arrive continuously, the pipeline is unsustainable once saturated. Expressivity of a resource is related to the computational time of the query, since for a query that cannot be express on a resource, its $\mathcal{C} \rightarrow \infty$.

$$\begin{aligned} &\mathcal{C}^{\Omega}(q_i, r_j) = x \mid x \in \mathbf{R}^+ && \text{if } \Omega \leq \Omega^C_{max} \;\&\; q_i \text{ is expressive on } r_j \\ &\mathcal{C}^{\Omega}(q_i, r_j) = \infty && \text{if } \Omega > \Omega^C_{max} \;\&\; q_i \text{ is expressive on } r_j \\ &\mathcal{C}^{\Omega}(q_i, r_j) = \infty && \text{if } q_i \text{ is } \textit{inexpressive} \text{ on } r_j \end{aligned}$$

The total *makespan* $\mathcal{T}$ for the path in the pipeline is the time taken to generate a single event from the output query after processing input events by each query on the resource it is mapped to, and transferring events between resources. For a mapping $\mathcal{M}$ and input rate $\Omega_\succ$, the makespan for path p is computed using its rate, latency and compute functions:

$$\mathcal{T} = \sum_{q_i \in p} \mathcal{C}^{\Omega(q_i)}\big(q_i, \mathcal{M}(q_j)\big) + \sum_{q_i \in p} \mathcal{L}^{\Omega(q_i)}\big(\mathcal{M}(q_i), \mathcal{M}(q_j)\big)$$

As discussed before, edge devices are often collocated with the incoming event sources, while the eventual decision making happens on the Cloud. An *edge cut* is a bi-partition of the path into two disjoint path segments. It denotes the logical partitioning of the queries between edge and Cloud resources. Queries in the first path segment execute on the edge, queries in the other path segment execute on the Cloud, and stream(s) at the split move events across these resources. The mapping $\mathcal{M}$ captures this. For edge cuts to form a consistent bi-partition, they should appear on a *Sequential*, *Source*, or *Sink* Segment, or on each sequential segment present in a *Parallel Segment*, recursively.

Optimization Problem Definition. *Given* the query pipeline and its path, the selectivity and anonymization of each query, the resources in the IoT system, the compute time on each resource for a query, the latency between resources, and a given incoming event rate, the optimization problem is to find a edge cut that consistently bi-partitions the path and maps it across different resources such that the makespan time for the pipeline is *minimized*, while ensuring that the privacy *constraint* is satisfied.

$$\begin{aligned} \text{Given: } & \mathbb{G}, p, \gamma, \mathbb{A}, \mathbb{R}, \mathcal{C}, \mathcal{L}, \Omega_\succ \\ \text{Minimize: } & \mathcal{T} \\ \text{Subject to Constraint: } & \mathcal{P} \\ \text{Output: } & \mathcal{M} : \mathbb{Q} \rightarrow \mathbb{R} \end{aligned}$$

In the Campus Smart Water project, a sample CEP pipeline performs three queries: data cleaning, temporal aggregation, and detecting water overflow. This forms the dataflow graph, $\mathbb{G}$. The resource r_e is the Android board reporting the water level observations, $\mathcal{L}$ is the network latency in moving this data stream to the Cloud r_c for processing.

Algorithm 1 SplitBruteForce

```
procedure SplitBF(EdgeCut $EdgeCuts[\,]$, $p$, $\Omega_\succ$)
    $T_{min} = \infty,\ LHS = 0,\ RHS = 0$
    for $s_x \in EdgeCuts[\,]$ do                                   ▶ Test each cut
                                          ▶ Calculate LHS and RHS costs for cut
        for $s_k = \langle \varnothing, q_\succ \rangle$ to $s_x \mid \{s_k = \langle q_i, q_j \rangle \in p\}$ do
            $LHS = LHS + \mathcal{L}^{\Omega(q_i)}(r_e, r_e) + \mathcal{C}(q_j, r_e)$
        end for
        for $s_k = s_{x+1}$ to $\langle q_\prec, \varnothing \rangle \mid \{s_k = \langle q_i, q_j \rangle \in p\}$ do
            $RHS = RHS + \mathcal{L}^{\Omega(q_i)}(r_c, r_c) + \mathcal{C}(q_j, r_c)$
        end for
        $T = LHS + RHS + \sum_{\langle q_x, \star \rangle \in s_x} \mathcal{L}^{\Omega(q_x)}(r_e, r_c)$
        if $T < T_{min}$ then                                ▶ Found a better cut?
            $T_{min} = T,\ Split = s_x$
        end if
    end for
    return $\langle Split, T_{min} \rangle$
end procedure
```

Algorithm 2 SplitDynamicProgramming

```
procedure SplitDP(EdgeCut $EdgeCuts[\,]$, $p$, $\Omega$)
    $n =$ SizeOf($EdgeCuts$), $T[\,] = float[1..n]$
    $T[1] = \sum_{q_i \in p} \mathcal{C}(q_i, r_c)$                      ▶ Run fully on Cloud
    $prev = 1$
    Define Update($prev$) : $T[i] = T[prev] - \mathcal{C}(q_j, r_c) +$
    $\mathcal{C}(q_j, r_e) + \mathcal{L}^{\Omega(q_j)}(r_e, r_c)$                 ▶ Incremental update
    for $i = 1\ to\ n$ do
        $s_i = \langle q_j, q_k \rangle \in EdgeCuts[\,]$
        if IsSeqSeg($s_i$) then                          ▶ Cut Sequential
            Update($prev$)                             ▶ Move $q_j$ to edge
        else                ▶ Cut Parallel
            if $s_i and s_p rev have same parent stream$ then
                Update($prev$)
            else
                $ancestor =$ Find($EdgeCuts$,
                        Parent stream sharing suffix edge cuts)
                Update($ancestor$)
            end if
        end if
        $prev = i$
    end for
    return $\langle EdgeCuts[$IndexOf(Min($T$))$]$, Min($T$)$\rangle$
end procedure
```

4. SOLUTION APPROACH

Edge cuts are locations in the path where the queries can be split between the edge and the Cloud resources. The EdgeCutDetection algorithm (omitted for brevity) scans every stream in the path and decide if it is a candidate for a split. It recursively identifies consistent edge cut candidates in the path, starting with a cut that can happen prior to the source queries (i.e., the entire pipeline is executed fully on the Cloud) to a cut after the sink query (pipeline runs entirely on the edge). The algorithm further limits cuts to those that satisfy the privacy constraints on the streams, by forcing anonymized queries to be on the edge (left hand side, or LHS) or the Cloud (RHS) as needed. Each edge cut includes one or more streams, and the cuts are stored in *EdgeCuts*[ ], an array of linked lists, when every item in the linked list represents a stream in the edge cut. If the linked lists has only one item, it is a cut of a sequential path segment, and multiple items indicate cuts across parallel segments, one item per parallel segment.

We propose two algorithms SplitBruteForce (Alg. 1) and SplitDP (Alg. 2) to obtain the optimal split that minimizes the *makespan*. SplitBruteForce computes $\mathcal{T}$ at all candidate edge cuts by calculating each of their compute times on the edge and the Cloud, and the latency to transfer output events from the edge to the Cloud. It then picks the cut with the least $\mathcal{T}$. While tractable for short pipelines, its time complexity is $O(|\mathbb{S}|^2)$ for even sequential pipelines.

SplitDP uses *Dynamic Programming* to compute the $\mathcal{T}$ assuming the pipeline fully runs in the Cloud, and incrementally moves one query at a time to the edge. It uses memoize to store previously computed values of $\mathcal{T}$ for edge cuts, and reuses it to truncate the computation for incrementally longer cuts. This has a time complexity of $O(|\mathbb{S}|)$.

5. FUTURE WORK

As part of future work, we propose to extend the formalization by allowing multiple edge devices and Cloud VMs for a single pipeline, and using an edge device's power level and efficiency as a QoS factor. We can also introduce *ad hoc* anonymization queries for better privacy control, and allow fine-grained partitioning of a single CEP query rather than between CEP queries. We will also explore additional solutions to the optimization problem, including those that allow dynamic repartitioning based on changing conditions and user needs. The proposed solutions need to be evaluated for practical feasibility and scalability through empirical benchmarks.

6. REFERENCES

[1] A. Adi, D. Botzer et al. Complex event processing for financial services. In *IEEE Services Computing Workshops*, 2006.

[2] M. Akdere, U. Çetintemel, and N. Tatbul. Plan-based complex event detection across distributed sources. *Proc. VLDB Endow.*, 1(1):66–77, Aug. 2008.

[3] W. Z. B. Chandramouli, S. Nath. Supporting distributed feed-following apps over edge devices. *PVLDB*, 6(13), 2014.

[4] C. Bormann, A. P. Castellani, and Z. Shelby. Coap: An application protocol for billions of tiny internet nodes. *IEEE Internet Computing*, 2012.

[5] G. Cugola and A. Margara. Complex event processing with T-REX. *Journal of Systems and Software*, 85(8):1709–1728, 2012.

[6] G. Cugola and A. Margara. Processing flows of information: From data stream to complex event processing. *ACM Computing Surveys*, 44(3), June 2012.

[7] A. Demers, J. Gehrke, and P. Biswanath. Cayuga: A general purpose event monitoring system. In *CIDR*, pages 412–422, 2007.

[8] R. Mayer. Real-time distributed complex event processing for big data scenarios. In *Distributed Event-Based Systems (DEBS) Ph.D. Forum*, 2013.

[9] S. Suhothayan, K. Gajasinghe et al. Siddhi: A second look at complex event processing architectures. In *ACM GCE Workshop*, 2011. http://siddhi.sourceforge.net.

[10] Y. Simmhan, S. Aman et al. Cloud-based software platform for data-driven smart grid management. *Computing in Science and Engineering*, 2013.

Exploratory Data Analysis Using Alternating Covers of Rules and Exceptions

S. Saikia*, G. Shroff*, P. Agarwal*, A. Srinivasan+, A. Pandey*, G. Anand*
sarmimala.saikia, gautam.shroff, puneet.a, aditeya.pandey, gaurangi.anand@tcs.com; ashwin.srinivasan@iiitd.ac.in
*TCS Research, +Indraprastha Institute of Information Technology Delhi

ABSTRACT

During exploratory data analysis in a variety of domains we have found it useful to summarize the available data via sets of rules and their exceptions. Often, a practitioner is interested in exploring more than one such summary before settling on any single explanation. It is important for each summary to be succinct, cover a significant fraction of the data as well as highlight rules enjoying high confidence, support and lift. It is often difficult to achieve all these objectives simultaneously, and retain a meaningful dialogue with the user. To help achieve these goals, we have found it useful to combine some existing tools for data analysis: association rule-mining to find broad patterns in the data; clustering to group these patterns into similar sets; exception-mining to find deviations from the patterns; and a coverage-based extraction of explanations for the data. We demonstrate how a combination of these methods can be used for exploratory data analysis on both synthetic and industrial data sets for which we derived interesting insights.

1. INTRODUCTION

During exploratory analysis we often seek the causes or effects of specific situations, such as the kind of consumers who often use a particular product feature, the situations in which a vehicle heats up, or how an engine behaves when under heavy load. In such cases rule-based explanations are easy to understand and therefore desirable. Practitioners often want:

Alternatives: To automatically identify alternative descriptions for a subset of data; (or at least, different patterns that roughly describe the same subset of data);
Comprehensibility: To ensure that the descriptions found are meaningful, given what is already known;
Explanations: To extract from the descriptions, one or more combinations that can adequately "explain" the data;
Deviations: To understand outliers to these explanations.

With rules as a representation language, most coverage-based methods do not allow a user access to alternative descriptions. Instead, a greedy procedure is usually adopted, in which the next best rule is found, the corresponding data explained ("covered") is removed from further consideration, and the process is repeated. This results in a single explanation of the data. So, while they satisfy the "Explanation" requirement by constructing one explanation for the data, they typically do not present any solutions to finding alternatives or deviations. Comprehensibility may follow if the rules do not represent overly complex concepts.[1]

Although association rule mining [1] is particularly a widely used approach for finding patterns in data, by itself it will not satisfy any of the requirements of a practitioner, as described above, since it is simply aimed at finding patterns that exist in the data. Nevertheless, in conjunction with other techniques, some useful headway can be made. For example, clustering of association rules has been proposed to group similar sets of rules [7, 8, 11, 16, 18, 20], thus addressing the requirement of "Alternatives" above. Exception mining in conjunction with association rules has been proposed as a way of identifying comprehensible association rules and deviations from them [4, 5, 12, 17, 20].

It would appear that a combination of the methods described may result in a tool that could satisfy each of the requirements listed. In this paper we describe a tool of such a kind. ACRE (**A**lternating **C**overs of **R**ules and **E**xceptions), uses a combination of association-rule mining, clustering, and exception-mining to allow us to address the "Alternatives", "Comprehensibility" and "Deviations" requirement. In addition, it allows multiple rule-sets to be extracted, using a coverage-based approach (the "Explanation" need).

The basic procedure is as follows. ACRE first computes a rule set via association rule mining. Many of these rules usually overlap with each other, i.e., they cover many of the same data instances. ACRE clusters the rules based on their mutual overlap. Each cluster of rules thus roughly covers the same subset of the data. Whenever an explanation is sought for the entire dataset, a rule is selected in turn from each cluster until no further rules are required, or possible. By altering the choice of the rule chosen from each cluster, ACRE is able to provide alternative explanations for the same dataset. For each rule, ACRE is also able to identify exceptions in the data. These data instances can then be analyzed by ACRE in the same manner as before, namely using association rule-mining, clustering, explanation and, if needed, further levels of exception-mining.

From the user's viewpoint, we expect ACRE to be used

The 20th International Conference on Management of Data (COMAD),
17th-19st Dec, 2014 at Hyderabad, India.

---

[1] The work of Srinivasan et al [15] is an example of a covering procedure that also highlights deviations.

as follows. The user identifies a data set for exploratory analysis. Data analysis is selection of rules iteratively from clusters found by ACRE, and examination of exceptions until a satisfactory set of patterns are found.

In the sections that follow we summarize the related work on clustering of association rules as well as exception mining and then describe ACRE. We present our results qualitatively using our previous technique [14] for visualizing interactively, rules and exceptions as well as quantitatively in terms of F1-measure, both for ACRE as well as standard rule-based classifier (CN2, [3]).

2. RELATED WORK

Rule Summarization: Pruning and grouping association rules using clustering was introduced in [18]. A rule cover is a subset of rules that covers almost all data instances as the original set. In [16] a normalized distance metric (different from ours) is used to cluster association rules along with self organizing feature maps. These two approaches are similar to our approach, however, they do not augment the rules with exceptions. Rule-templates [6] can be used to select interesting rules from a larger set, based on user-defined criteria. [11] prunes a rule set by removing the insignificant ones, and then finds a special subset called Direct Setting (DS) rules to form a summary; however, DS-rules do not include rule-exceptions as we do. A technique similar to ACRE was used in [19] for summarizing a collection of frequent itemsets, using K representatives by clustering them to create a pattern profile representing sets of similar frequent itemsets. [8] introduces a technique for clustering association rules using a geometric-based algorithm. However, both these approaches filter rules based on the pattern of the clustered frequent itemsets, rather than coverage.

Exception Mining: Deviation analysis was used in [12] to identify interesting exceptions and explore reliable ones; [5] used information theory measures and evaluated exceptions based on the common sense rules, negative association rules were used in [4] to find exceptions and [17] used an autonomous probabilistic model to find out exceptions and common sense rule pairs with high confidence. However, mining of exceptions, and pruning of association rules have always been separate areas of research. We have combined both these techniques so as to derive a hierarchical summary that alternates between small sets of covering rules and their exceptions to provide a succinct summary for a desired consequent of interest. An approach similar to ours has also been used in [20] to find useful association rules and exceptional association rules for each of them, however, they do not cluster rules or exceptions to offer choices as we do. Finally, [13, 10] find 'unexpected' rules that are exceptions to *user-supplied* existing knowledge about the domain.

3. ACRE ALGORITHM

Each data instance is called a transaction that contains one or more items from a set of items $I = \{i_1, i_2......., i_n\}$: For example each survey response is a transaction, in which items are the customer's responses to each question asked. In multi-sensor data, each time step is a transaction with the individual values of different sensors forming items, after suitable discretization. A subset of I is referred to as an *itemset*; frequent itemsets are those that occur more often than others. Each frequent itemset, say $\{X, y\}$, may form a rule r for a pre-determined 'consequent of interest' (COI) y, with the subset of items (X) as antecedent, i.e., $X \rightarrow y$.

The *support of a rule* $S(r)$ is the percentage of transactions that contain all items in $(X \cup y)$. The *confidence of a rule* $C(r)$ is $P(y|X)$. Further, *lift* of a rule is a measure of its interestingness and is the ratio of its confidence and the probability of consequent, i.e., $L(r) = P(y|X)/P(y)$.

Since confidence is not usually equal to one, we also attempt to find patterns in the subset of transactions satisfying the antecedents of a rule. Such rules, i.e., $X, Z \rightarrow \neg y$, are called *exceptions* of the main rule $X \rightarrow y$.

The *coverage of a rule* is $\rho(r) = P(X \cup y)/P(y)$, indicating the percentage of transactions where the rule is satisfied, out of those that contain the consequent of interest y. However, rules can overlap in the data, so the coverage of multiple rules may be far less than the sum of their coverages.

Given a set of rules $R = \{r_1, r_2, ..., r_N\}$ having common consequent (y), we define *Rule cover* $R_{co} = \{r_1, r_2, ..., r_k\}$ as a subset of R, which cover almost the same set of transactions as covered by R. We quantify the *degree of overlap* O_{ij} between two rules $\{r_i, r_j\}$ as $O_{ij} = \frac{S(r_i \bigcap r_j)}{min(S(r_i), S(r_j))}$. A *distance measure* d_{ij} between a pair of rules $\{r_i, r_j\}$ is defined as $d_{ij} = \frac{1}{(O_{ij}+\kappa)}$ ($\kappa = .01$ is a small constant).

ACRE: consists of data processing steps that are performed for data understanding at two levels, to generate alternative yet 'coverage equivalent' rule-exception covers.

Data Processing:

Step-1: Rule Generation, we generate a set of rules R for a COI with support $> \tau_s$, and confidence $> \tau_p$ using frequent itemsets derived by PFP-growth algorithm [9].

Step-2: Rule Cover: We find a rule cover from R, such that the cumulative coverage of the rule cover R_{co} is at-least $\tau_r\%$. For this we scan the rules in R in descending order of support, and add them to rule cover R_{co} until $\tau_r\%$ of the transactions containing COI are covered, or top-K rules are included in R_{co}, whichever comes first.

Step-3: Rule Clustering: The rules in R_{co} are clustered based on overlap using the distance measure d_{ij} using DBSCAN (optimizing parameters using gradient descent).

Multilevel Data Understanding:

At **Level 1**, steps 1-3 are performed on the entire set of transactions, yielding a set of rules R. Using an interactive rule visualization such as VARC [14], the user *chooses one rule from each cluster.* At **Level-2**, we repeat steps 1-3 for the set of transactions covered by the antecedents of each rule in R, however, this time we use $\neg y$ as the COI. This yields a set of exceptions for every rule in R. Note that the confidence threshold used for mining exceptions for rule $r : X \rightarrow y$ is $\tau_e = (100 + \Delta c - C(r))$, where Δc is called the confidence *gap*: For example, consider using a gap of 20% - if r has 85% confidence, 15% of the time we anyway expect $\neg y$ when X is true, so we would consider any exception implying $\neg y$ as significant if it had a confidence of even 35%.

4. RESULTS AND EVALUATION

We distinguish between two ways of using ACRE: in **batch mode**, at each level, we choose the highest support rule from each cluster, in **interactive mode**, the user *interactively chooses alternative rules from each cluster (at each level)* thereby allowing many equivalent alternating rule-exception covers to be examined. We expect batch mode to be more useful for data sets that have relatively few items. In higher

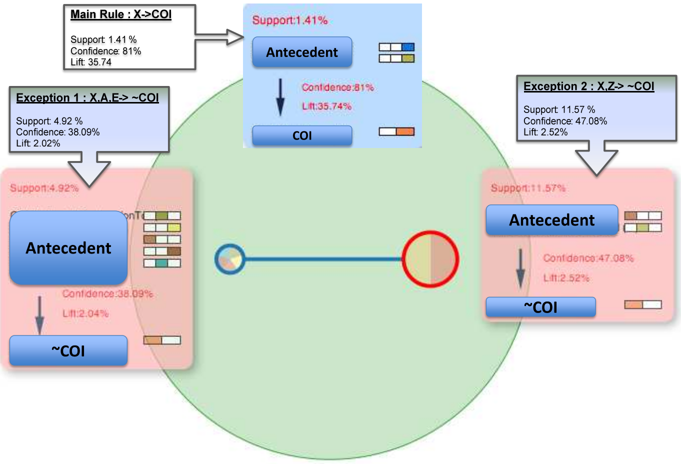


Figure 1: VARC depiction of a Sensor-I rule

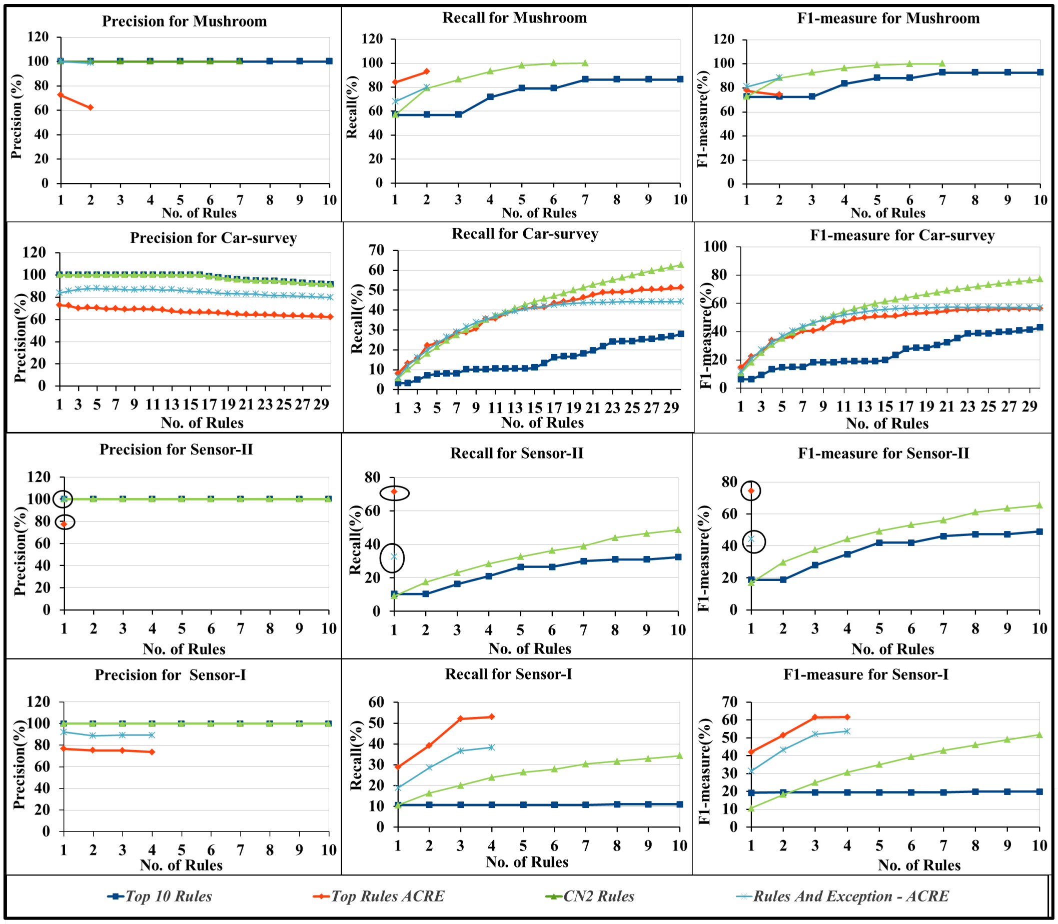


Figure 2: Precision, Recall and F1-measure graph for Mushroom, Car-Survey, Sensor-I, and Sensor-II datasets

dimensional spaces, an interactive mode is expected to be more useful because one is more likely to find relatively disjoint rules that nevertheless overlap in the data. However, in such cases frequent itemset mining itself has performance challenges (which, to our knowledge, remains an open problem). Here we present the results for batch mode on relatively narrow data.

We used the standard public Mushroom dataset [2] (poisonous vs non-poisonous mushrooms), as well as three real-life datasets: Car-survey, which captures customer driving patterns as well as usage of special vehicle features, and two vehicular sensor datasets: Sensor-I and Sensor-II. The Car-survey dataset has 1,153 transactions with 17 attributes while Sensor-I has 1,603 transactions with 18 attributes and Sensor-II has 54,565 transactions with 14 attributes.

ACRE requires input parameters such as minimum support threshold τ_s, minimum confidence threshold τ_p, coverage percentage τ_r, top-K rules K, and Δc. We used τ_p as 70% in general, unless the class-probability of the COI is greater, in which case $\tau_p = COI_{classProbability} + 10$. For top rules $\tau_r = 90\%$ and $K = 200$ in case if the required coverage is not achieved. However, for discovering exceptions we set $\tau_r = 75\%$ and $\Delta c = 20\%$.

Figure 1 depicts one of the rules discovered for Sensor-I data using the VARC visualization [14]. Our real-life datasets are proprietary, so the actual items are masked. Using 15 car-survey rules we described 40% of customers who used a particular enhanced mode based on their driving. Four Sensor-I rules described the effect of highly loading an engine, and one Sensor-II rule (with four exceptions) causally explained high-temperature situations.

Even though our goal is data-understanding rather than classification accuracy; still we can use F1-measure for quantitative evaluation from an IR perspective, i.e., balancing precision vs. recall of the consequent of interest. Figure 2 shows the F1-measure obtained by ACRE, both for top-level rules alone as well as rules+exceptions. These are compared with the top association rules as well as rules obtained using CN2 [3]. (Rule-sets are ordered similarly for a fair comparison, i.e., the best n rules in terms of F1-measure are chosen.)

Notice that ACRE has been always as good as CN2 (for Mushroom and Car-survey), at least for the succinct explanation it provides. Further, ACRE provides a markedly better F1-measure than CN2 for Sensor-I and Sensor-II. Next, notice how exceptions improve precision while reducing coverage somewhat. Thus, if coverage is more important top-level ACRE rules can be used without exceptions if their precision is deemed adequate. At the same time, their exceptions are always available for a deeper explanation that also improves precision, albeit affecting recall to some extent.

5. CONCLUSIONS

We have described ACRE, a technique for computing a set of rules and their exceptions that achieves high coverage of a desired consequent of interest as well as reasonable precision. ACRE clusters association rules at two levels to minimize inter-rule overlaps. Experimental results show that ACRE provides a succinct explanation with an adequately higher F1-measure, and is found to be as good or better than CN2 especially on our real-life data. Further, ACRE naturally enables qualitatively alternative yet quantitatively equivalent rule-exception covers by allowing for choosing different rule sets from each cluster of rules, at each level, thus allowing for more interactive and effective use in practice.

6. REFERENCES

[1] R. Agrawal, T. Imieliński, and A. Swami. Mining association rules between sets of items in large databases. In *SIGMOD 1993*.

[2] C. Blake and C. J. Merz. UCI repository of machine learning databases. 1998.

[3] P. Clark and T. Niblett. The CN2 induction algorithm. *Machine learning*, 1989.

[4] O. Daly and D. Taniar. Exception rules mining based on negative association rules. In *ICCSA 2004*.

[5] F. Hussain, H. Liu, E. Suzuki, and H. Lu. Exception rule mining with a relative interestingness measure. In *Knowledge Discovery and Data Mining*. Springer Berlin Heidelberg, 2000.

[6] M. Klemettinen, H. Mannila, P. Ronkainen, H. Toivonen, and A. I. Verkamo. Finding interesting rules from large sets of discovered association rules. In *CIKM 1994*.

[7] S. Kotsiantis and D. Kanellopoulos. Association rules mining: A recent overview. *GESTS Intl Trans on Computer Science and Engineering*, 2006.

[8] B. Lent, A. Swami, and J. Widom. Clustering association rules. In *ICDE 1997*.

[9] H. Li, Y. Wang, D. Zhang, M. Zhang, and E. Y. Chang. Pfp: Parallel fp-growth for query recommendation. In *RecSys 2008*.

[10] B. Liu, W. Hsu, and S. Chen. Using general impressions to analyze discovered classification rules. *KDD 1997*.

[11] B. Liu, W. Hsu, and Y. Ma. Pruning and summarizing the discovered associations. *KDD 1999*.

[12] H. Liu, H. Lu, L. Feng, and F. Hussain. In *Methodologies for Knowledge Discovery and Data Mining*. Springer Berlin Heidelberg, 1999.

[13] B. Padmanabhan and A. Tuzhilin. A belief-driven method for discovering unexpected patterns. *KDD 1998*.

[14] G. Sharma, G. Shroff, A. Pandey, P. Agarwal, and A. Srinivasan. Interactively visualizing summaries of rules and exceptions. In *EuroVis Workshop on Visual Analytics*, 2014.

[15] A. Srinivasan, S. Muggleton, and M. Bain. Distinguishing exceptions from noise in non-monotonic learning. In *Proc of 2nd Inductive Logic Programming Workshop*, 1992.

[16] A. Strehl, G. K. Gupta, and J. Ghosh. Distance based clustering of association rules. In *ANNIE 1999*.

[17] E. Suzuki. Autonomous discovery of reliable exception rules. *KDD 1997*.

[18] H. Toivonen, M. Klemettinen, P. Ronkainen, K. Htnen, and H. Mannila. Pruning and grouping discovered association rules. In *ECML 1995*.

[19] X. Yan, H. Cheng, J. Han, and D. Xin. Summarizing itemset patterns: A profile-based approach. *KDD 2005*.

[20] A. Zhou, L. Wei, and F. Yu. Effective discovery of exception class association rules. *Journal of Computer Science and Technology*, 2002.

sv(M)kmeans - A Hybrid Feature Selection Technique for Reducing False Positives in Network Anomaly Detection

Shubham Saini*
Undergraduate Student
shubham.saini2010@vit.ac.in

Shraey Bhatia*
Undergraduate Student
shraey.bhatia2010@vit.ac.in

I. Sumaiya Thaseen
Advisor
isumaiyathaseen@vit.ac.in

School of Computing Science and Engineering
Vellore Institute of Technology, India

ABSTRACT

Feature Selection in large multi-dimensional data sets is becoming increasingly important for several real world applications. One such application, used by network administrators, is Network Intrusion Detection. The major problem with anomaly based intrusion detection systems is high number of false positives. Motivated by such a requirement, we propose sv(M)kmeans: a two step hybrid feature selection technique. The proposed technique applies classification on false-positives and true positives; and on false-positives and true-negatives after an initial round of clustering. Specifically, SVM-RFE is applied on the results obtains from MK-Means. sv(M)kmeans is evaluated for its real world applicability using the benchmark NSL-KDD data set. We show the feature subset to significantly reduce the false positive rate and increase the accuracy in network anomaly detection.

Categories and Subject Descriptors

I.5.2 [**PATTERN RECOGNITION**]: Design Methodology—*Feature evaluation and selection*

Keywords

feature selection, false-positive, intrusion detection

1. INTRODUCTION

Feature selection is an active area of research in the domains that require the use of data-sets with tens or hundreds of features. In his work "Statistical Modeling: The Two Cultures" [1], Breiman refers to the Rashomon Effect derived from a Japanese movie in which four different witnesses, from different vantage points, report the same facts in court but with different stories of what happened. The same is true for features in a data-set. The objective of feature selection is to extract a faster and more relevant feature subset that gives a better understanding of the data generation process, with little or no reduction in the overall predictive performance.

With the use of advanced data mining techniques, anomaly based intrusion detection systems (IDS) are able to prevent intrusions with little human intervention. However, high number of false positives wastes a lot of time and effort of network administrators.

False Positives or False Alarms are legitimate activities that IDS detects as malicious. The primary role of IDS is to detect a substantial percentage of intrusions, with false positive rate at an acceptable level. In real environments IDS throw an abundance of alerts, most of them being false positives. High number of false positives wastes a lot of time and effort of network administrators.

Most clustering and classification algorithms suffer from problems such as computational complexity, over-fitting etc. Ensemble learning is a popular approach to reduce the effect of these common problems. Ensemble approaches work on a simple intuitive idea of a group of experts working together on a problem. A group of experts with diverse experience in solving a problem will have a higher probability of arriving at an acceptable solution than a single expert. We used the same idea to experiment with a hybrid approach for feature selection using two distinct approaches.

To address these problems, we present sv(M)kmeans: a two step hybrid feature selection technique. The proposed technique applies classification on false-positives and true positives; and on false-positives and true-negatives after an initial round of clustering.

The key contributions of this paper are:

- sv(M)kmeans helps in reducing the false positive rate in anomaly based network intrusion detection systems.
- The reduced feature subset obtained through sv(M)kmeans helps in faster execution of MK-Means algorithm.
- Validation of the proposed technique using the benchmark NSL-KDD data set.

2. BACKGROUND AND RELATED WORK

Several studies involving hybrid clustering and classification approaches have been conducted previously. Vibha et al.[8] proposed a combined classification based on clustering for multispectral LANDSAT images for soil mining. In the

---

*This work was completed while the authors were full-time undergraduate students.

The 20th International Conference on Management of Data (COMAD),
17th-19st Dec, 2014 at Hyderabad, India.

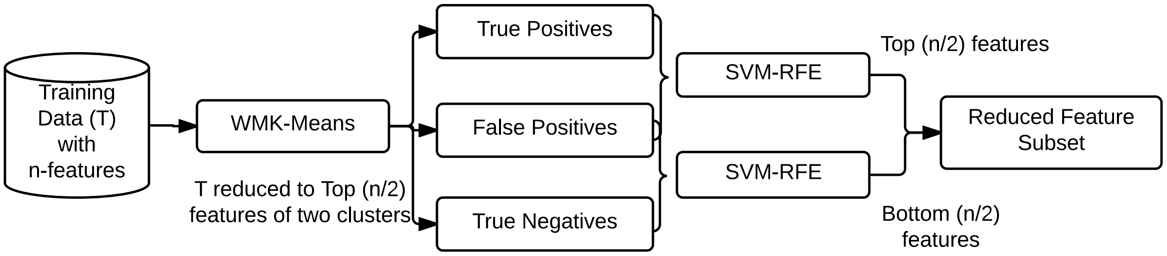


Figure 1: sv(M)kmeans: A two step hybrid feature selection technique

context of intrusion detection, Singh et al.[6] used a hybrid approach with a collective goal of reducing false positive rate and detection of previously unknown attacks. More recently, Li et al. [4] presented a clustering, ant-colony and SVM hybrid to incrementally remove features for intrusion detection. Ravale [5] proposed a similar K-Means - SVM hybrid feature selection by training a SVM classifier on the results of clustering.

We propose a two step hybrid methodology making use of MK-Means[2] and SVM-RFE[3] feature selection algorithms that helps to reduce the false positive rate in an IDS. Our work is significantly different from previous studies in several ways. The primary objective of our work is substantial reduction in the false positive rate of intrusion detection systems. Algorithms used to reduce false positive rate generally increase the false negative rate, there by having little improvement in the accuracy. sv(M)kmeans is experimentally shown to have little effect on the false negative rate, hence improving the overall accuracy.

2.1 MK-Means

K-Means is a machine learning algorithm used to partition a data-set into K clusters. The objects assigned to the resulting clusters have the smallest distance with respect to the centroid of their clusters. The Euclidean distance is the most popular measure used with K-Means. A more general measure is the Minkowski distance which can be considered as a generalization of both the Euclidean distance and the Manhattan distance.

$$d(x,y) = (\sum_{v=1}^{V} +|x_v - y_v|^p)^{1/p} \tag{1}$$

Amorim et al. makes use of the Minkowski measure to calculate the feature weights for each cluster. In order to make use of the calculated feature weights, the K-Means criteria is modified as:

$$W_p(S,C,w) = \sum_{k=1}^{K} \sum_{i \in S_k} \sum_{v=1}^{V} w_{kv}^{p} |y_{iv} - c_{kv}|^p \tag{2}$$

Feature weights are calculated using EQ (3). This follows from the idea that features with a small relative dispersion within a cluster are given a higher weight.

$$w_{kv} = \frac{1}{\sum_{u \in V} [D_{kv}/D_{Ku}]^{1/(p-1)}} \tag{3}$$

2.2 SVM-RFE

Support Vector Machines - Recursive Feature Elimination gives a ranking of features for a problem trained by SVM using a linear kernel function. After each iteration the feature with lowest ranking criteria is removed. SVM-RFE uses the weight magnitude as the ranking criterion.

1. Train an SVM on the training set
2. Order features using weights of the resulting classifier
3. Eliminate features with the smallest weight
4. Repeat the process with the training set restricted to the remaining features

3. SV(M)KMEANS

We now present sv(M)kmeans - our two step hybrid feature selection technique. Figure 1 gives a summary of the feature selection process. In order to reduce occurrence of false positives, we use a two step hybrid methodology making use of MK-Means and SVM-RFE feature selection algorithms. Algorithm 1 illustrated below is used to obtain a feature subset that we show to be better predictors and help reducing the false positive rate.

Algorithm 1 Feature selection using MK-Means and SVM-RFE

```
1: GIVEN Dataset $D = \{(x_i, c_i) | x_i \in R^n, c_i \in \{0,1\}\}$
2: Features F = 1,2,...n
3: Take Training Data $T^f$ where $T \subset D$ and $f = F$
4: $[U, W] \leftarrow MKMeans(T^f, 2, Beta, p)$
5: $F_2 \leftarrow$ Top (n/2) features of $W_1 \cup$ Top (n/2) features of
   $W_2$
6: Compare U and $c_i$
7: $T_{FP}^{F_2} \leftarrow \{(xi, ci) | xi \in false-positives\}$
8: $T_{TP}^{F_2} \leftarrow \{(xi, ci) | xi \in true-positives\}$
9: $T_{TN}^{F_2} \leftarrow \{(xi, ci) | xi \in true-negatives\}$
10: $F_{FP-TP} \leftarrow SVM-RFE(T_{FP}^{F_2}, T_{TP}^{F_2})$
11: $F_{FP-TN} \leftarrow SVM-RFE(T_{FP}^{F_2}, T_{TN}^{F_2})$
12: $F_{final} \leftarrow$ Top (n/2) features of $F_{FP-TP} \cup$ Bottom (n/2)
    features of $F_{FP-TN}$
```

The algorithm requires a labeled data-set D, with class labels 0 and 1 for normal and anomaly records respectively. Each record in the data-set is a n-dimensional real vector. In line 3 we take a subset of data set as training data. We apply MK-Means function as given in Line 4 on the n-dimensional

Table 1: Summary of the Performance Metrics with Generic Anomalies

Stage	#Features	Avg FPR(%)	Avg Accuracy(%)	Min FPR(%)	Max Accuracy(%)	Execution Time (s)
Initial	41	15.40	83.29	12.90	83.59	5
After MK-Means	39	16.29	85.19	12.32	83.92	5
After SVM-RFE	31	4.10	85.90	3.92	86.99	3

Table 2: Summary of the Performance Metrics with Neptune Anomaly

Stage	#Features	Avg FPR(%)	Avg Accuracy(%)	Min FPR(%)	Max Accuracy(%)	Execution Time (s)
Initial	41	5.88	96.10	4.01	97.99	4
After MK-Means	40	5.76	96.17	4.03	97.78	4
After SVM-RFE	36	3.36	97.46	2.24	98.67	3

data T, specifying the number of clusters (K) the data has to be divided into, the weight exponent (Beta) and the distance exponent (p). The result is the clustering detail (U), and weights of all features in each cluster (W). In Line 5, we find out top (n/2) features of each cluster and their union is stored as reduced feature subset (F_2). The training data is reduced to the dimensionality of F_2. In line 6, we find out the records which are detected as false positives, false negatives, true positives and true negatives from U. In lines 7-9, we store the records detected as false positives, true positives and true negatives respectively. In lines 10-11, we run SVM-RFE to find the strongest features that distinguish false positive from true positives and false positive from true negative respectively. Finally, we take the union of top (n/2) from the result of SVM-RFE on false positive and true positive and bottom (n/2) from the result of SVM-RFE on false positive and true negative. The rationale behind this being that we obtain the features that strongly differentiate false positives from true positives, and at the same time remove the features that differentiate false positives from true negatives.

4. EXPERIMENTAL SETUP

4.1 Data Set

KDDCUP'99 is the mostly widely used data set for anomaly detection. To solve some of its inherent problems Tavallaee et al.[7] suggested the NSL-KDD data-set. The NSL-KDD data set has several advantages over the original KDD data set such as no redundant records in the train and test sets, the number of selected records from each difficulty level group inversely proportional to the percentage of records in the original KDD data set. Table 3 gives a summary of the data set. The NSL-KDD data includes 41 features and 5 classes; normal and 4 types of attacks: Dos, Probe, R2L and U2R. We measure the performance of our proposed technique using NSL-KDD data-set.

4.2 Analytical Implementation

Experiments were done using R language for statistical computing version 3.0 on Linux 64-bit platform running on Intel Core i3 processor 2.4Ghz with 3GB RAM. Amorim has provided MATLAB implementation of MK-Means algorithm [1] which was ported to R language for experimentation. R implementation of SVM-RFE is available at the Data Mining group webpage [2] of Universidad Catlica de Crdoba. A hybrid MK-Means SVM-RFE algorithm was implemented by making use of the aforementioned pieces of codes.

Before applying the proposed algorithm to the dataset all attributes were normalized to the range 0-1.

Class	Attack Type	Number of Instances	
		Training	Test
Normal	-	67343	9711
DOS	Back, Land, Neptune, Pod, Smurf, Teardrop, Mailbomb, Processtable, Udpstorm, Apache2, Worm	45927	7456
Probe	Satan, IPsweep, Nmap, Portsweep, Mscan, Saint	11656	2421
R2L	Guess_password, Ftp_write, Imap, Phf, Multihop, Warezmaster, Xlock, Xsnoop, Snmpguess, Snmpgetattack, Httptunnel, Sendmail, Named	995	2756
U2R	Buffer_overflow, Loadmodule, Rootkit, Perl, Sqlattack, Xterm, Ps	52	200

Table 3: Overview of the NSL-KDD data set

5. RESULTS

Table 1 summarizes the performance metrics for data containing 25000 normal and 25000 generic anomaly class records. Further, Table 2 summarizes the test performed using 25000 records of specific anomaly type (Neptune class) with 25000 normal class records. Initially we start with 41 features in

---

[1] http://renatocamorim.com/2012/11/21/minkowski-weighted-k-means/

[2] http://www.uccor.edu.ar/paginas/seminarios/Software/SVM_RFE_R_implementation.pdf

the data-set, which are reduced to F_2 after running MK-Means. Finally, we have the feature subset F_{final} obtained using SVM-RFE. After each stage, we find the performance metrics by running MK-Means clustering algorithm on the reduced feature subset.

The results of MK-Means depend on the initial cluster centroids and number of centroids. We therefore ran each test 20 times. The average and best results are presented. It is evident from the experimental results that applying just MK-Means for feature selection did not yield a significant improvement. It was after the application of SVM-RFE that we notice reduction in false positive rate. It may also be noted that the decrease in false positives was accompanied by an increase in the overall prediction accuracy; thereby having little ill-effect on the false negative rate.

The computation time at each iteration is proportional to the product of the number of entities and the dimension of the feature space. The execution time for running the MK-Means algorithm is presented (rounded to the nearest integer). As evident from the table, the reduced feature space helps in reduction of the execution of the MK-Means algorithm, hence an added advantage.

6. CONCLUSION

In this paper, we presented sv(M)kmeans: a two step hybrid methodology for feature selection that allows for reduction in false positive rate in anomaly based network intrusion detection systems. It is evident from our experiments that the sv(M)kmeans is effective in reducing the false positive rate; from 12.9% to 3.92% in case of generic anomaly data and from 4.01% to 2.34% in case of specific anomaly data for the best case scenario. Interestingly, the technique reduced the false positive rate along with an increase in the overall prediction accuracy, thus having little effect on the false negative rate. This aspect needs further analysis to establish if it is true across diverse data sets. Apart from increasing the prediction accuracy, the reduced feature subset also helps reducing the execution time of MK-Means algorithm.

7. REFERENCES

[1] L. Breiman et al. Statistical modeling: The two cultures (with comments and a rejoinder by the author). *Statistical Science*, 16(3):199–231, 2001.

[2] R. Cordeiro de Amorim and B. Mirkin. Minkowski metric, feature weighting and anomalous cluster initializing in k-means clustering. *Pattern Recognition*, 45(3):1061–1075, 2012.

[3] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Machine learning*, 46(1-3):389–422, 2002.

[4] Y. Li, J. Xia, S. Zhang, J. Yan, X. Ai, and K. Dai. An efficient intrusion detection system based on support vector machines and gradually feature removal method. *Expert Systems with Applications*, 39(1):424 – 430, 2012.

[5] U. Ravale, N. Marathe, and P. Padiya. Attribute reduction based hybrid anomaly intrusion detection using k-means and svm classifier. *International Journal of Computer Applications*, 82(15):32–35, November 2013.

[6] V. Singh and P. Kaur. Adaptive distributed intrusion detection using hybrid k-means svm algorithm.

[7] M. Tavallaee, E. Bagheri, W. Lu, and A.-A. Ghorbani. A detailed analysis of the kdd cup 99 data set. In *Proceedings of the Second IEEE Symposium on Computational Intelligence for Security and Defence Applications 2009*, 2009.

[8] L. Vibha, G. HarshaVardhan, S. Prashanth, P. Deepa Shenoy, K. Venugopal, and L. Patnaik. A hybrid clustering and classification technique for soil data mining. In *Information and Communication Technology in Electrical Sciences (ICTES 2007), 2007. ICTES. IET-UK International Conference on*, pages 1090–1095. IET, 2007.

Removing Noise Content from Online News Articles

Jayendra Barua
Indian Institute of Technology,
Roorkee, Uttarakhand
India 247667
jbarudec@iitr.ac.in

Dhaval Patel
Indian Institute of Technology,
Roorkee, Uttarakhand
India 247667
patelfec@iitr.ac.in

Ankur Kumar Agrawal
Indian Institute of Technology,
Roorkee, Uttarakhand
India 247667
ank70pec@iitr.ac.in

ABSTRACT

A typical news web page consists of news articles. Along with the news article content tags, it also contains tags of navigation links, privacy & copyright information and advertisements. These tags are called as noise tags. Given an online news article in html form, existing works extract articles by discovering informative tags using various heuristic techniques. In this paper, we follow an alternate approach that removes noise tags from the web page. In particular, we define two types of noise contents, namely *static noise content* and *dynamic noise content*, to be removed from a web page. Static noise content is the content which is present in all the article web pages of a same news website, whereas, the dynamic noise contents are advertisements and irrelevant hyperlinks which keep changing from one article web page to another web page. We present a two-stage approach for identification of static and dynamic noise content. After identifying tags with *noise content*, we remove these tags from the news webpage and thus only tags with article content will be left out. Experimental studies is done on news web pages extracted from 11 different news websites. Comparison with three well known open source content extraction techniques our approach suggest around 6% improvement in recall value with fair F1-score and precision value compared to best performing content extraction technique.

1. INTRODUCTION

Along with increase in publication of online news articles, many news aggregators, news recommendation and search systems such as Google News, Yahoo News etc. has been evolved which crawls various news websites and then extract the news content from the crawled news web pages. However, news content extraction from a web page is not a simple task because, along with the article content, news web pages also contain "*Noise*" in the form of irrelevant hyperlinks, advertisements, copyright information, etc. Yi et al. in [1] have also shown that noise can seriously harm the accuracy of various data mining algorithms that are performed on news article content. Given online news article in a form of an html form, existing works extract articles by discovering informative tags using various heuristic techniques [2] [4] [7]. In particular, these heuristic techniques calculate various statistics for each tag present in the web page and then identify the informative tags. Since, news aggregators crawl new websites multiple times for durable archival of contents, many news articles from the same news website are accumulated. In such scenario, it is possible to learn how a website publishes its own news articles. In particular, by analyzing a set of news articles published by a single news website, we can easily distinguish the noisy and informative content. Existing work [1] and [5] also uses the multiple news articles for efficient content extraction, but none of these technique leverages static and dynamic nature of the noise content in web pages.

Given a set of web pages *W* that are sampled from a single news web site N, we develop a document object model (DOM) tree based solution to remove the noise contents of each web page in *W*. Each web page in *W* is parsed using HTML parser and is represented as a DOM tree. Figure 1 shows a DOM tree representation of a news web page (some details omitted) where each node is an HTML tag. We have some observations based on DOM tree as follows: (1) Some nodes in DOM tree have same content in all the news web pages of *W*, for example Home, name of news website, copyright, print, follow us, contact us and many more. These content are not relevant to article text. We called these types of nodes as *Static Noise Tags*. These nodes may be either text nodes or hyperlink nodes. (2) Other noise nodes in the DOM tree are hyperlinks. Here some nodes are relevant to the article text content, but most of the nodes are irrelevant. Many of the irrelevant hyperlinks are generally listed under nodes such as Recent articles, Mostly Read articles, related news, etc. We call these types of nodes as *Dynamic Noise Tags*. We term these tags as "Dynamic" because information under these tags is not static across all news web pages *W*.

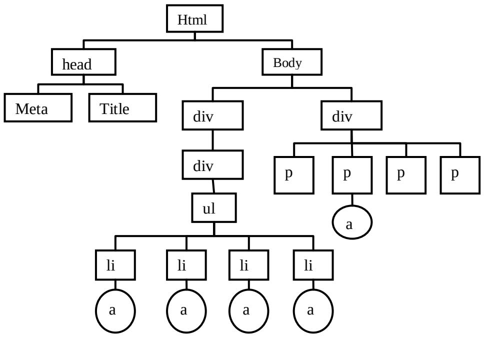


Figure 1: DOM tree of a news web page

In this paper, we use a two step approach, namely StaDyNoT, to remove *Static and Dynamic Noise Tags*. In the first step, we mark *Static Noise Tags* by using set of web pages extracted from the same website. In the second step, we mark *Dynamic noise Tags* by applying the common ancestor technique on hyperlink nodes of DOM tree of the given news web page. Finally, we remove all noise marked tags and thus only tags with article content will be left on the web page.

The 20th International Conference on Management of Data (COMAD),
17th-19th Dec, 2014 at Hyderabad, India.

Moreover, existing content extraction techniques output unformatted text from input web page. These methods cannot be used to extract structured data from the web pages since there output looses text formatting such as font-size, bold and italic text, etc. In contrast, instead of unformatted text, our technique outputs cleaned HTML web pages which retains text formatting. If needed, only unformatted text can also be easily extracted from these HTML web pages**.** Based on exprimental analysis, StaDyNoT achieves an overall average score of 88%, 96% and 91% for Precision, Recall and F1 respectively, which is remarkable in comparison with existing NReadability[1], Boilerplate[8] and *jusText*[3], content extraction techniques.

2. RELATED WORK

Retrieval of noiseless news content has attracted many researchers[1][2][3][4][5][6][7][8]. In order to retrieve noiseless news content, majority of the existing approaches searches for the article content in the web page and extract it. Compared to this, our method searches for noise content and remove it.

Yi et al. in [1] introduces a Site Style tree to capture common-styles element tags in web pages of a website. Then by identifying common-style they remove noise element tags from web pages. But the noise content on news web pages may not follow the common style always. Fan et al. in [2] proposes the detection of "print-link" on the web pages to extract article content. But many variations of the print-link representations makes print-link detection a non-trivial problem. A set of rules are used in *jusText* [3] that classify text blocks as "good" or "bad". The classification is done by applying context-free rules followed by context-aware rules. The work in [4] uses text density to extract the informative content. Sluban et al. in [5] introduces URL tree created using of stream of HTML web pages to extract the main content. In [9] a heuristic approach based on text-density and visual importance features of a web page is proposed by Song et al. for extraction of article content. Bhu et al. in [10] proposed a heuristic based approach which uses "fuzzy association rules" and "sliding window" to extract main text content from web pages. Peters et al. in [6] introduces tag attribute values as new block feature to improve article extraction. ECON method by Guo et al. in [7], works on principle that informative content contains more number of punctuation marks than noise content. Kohlschutter et al. in [8] proposes the detection of noise (termed as boilerplate) on the basis of shallow text features, structural features and densitometric features of text blocks in a web page. NReadability[1] is a tool, which removes clutter from HTML pages by applying several heuristics on DOM tree, based on textual and structural features of web page.

In comparison to the existing methods which are mostly heuristic based or supervised, we propose a simple, unsupervised and efficient approach which leverages static and dynamic nature of noise content for removing noise from news web pages.

3. OUR APPROACH: StaDyNoT

As discussed in the Introduction section, if we can remove both S*tatic Noise Tags* and *Dynamic Noise Tags* from news web page then only article text content will remain in the web page. Our approach works in 2 stages: 1) Marking of *Static Noise Tags* 2) Marking of *Dynamic Noise Tags.* Note that, in each stage we only mark the noise tags, but we do not remove noise tags. This is because removal of certain nodes may distort the DOM tree structure and we may not be able to process the tree in subsequent stage.

3.1 Mark Static Noise Tags

This stage utilizes a set of neighbor article web pages to mark a static data that are present in a target web page. A target web page is a page from which we want to extract the articles. By the term "*neighbor article web pages*", we mean the set of news web pages belongs to the same news website and same category as of the target article news web page. For example if the target article news web page belongs to sports category of CNN, then other news web pages from the sports category of CNN will be neighbor article web pages. "neighbor article web pages" are naturally available in news aggregators systems, as they crawls multiple web pages of same websites, Thus we leverage availability of multiple news web pages belongs to same news website. In Figure 2, we explain the adopted steps in detail. The proposed algorithm works in two phases. In the first phase, we analyze the neighbor article web pages, learn static data and then mark the tags having static data as noise tags.

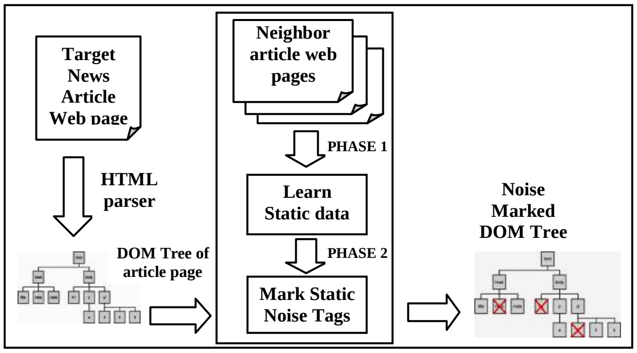


Figure 2: The process of marking *Static Noise Tags*

In particular, we traverse the DOM tree of every article web pages in neighbor article web pages. While traversal, we extract tag name, tag attribute and tag data of every node in a DOM tree and put them into a global hash table using key "tag-name : tag-attribute: tag_data" with support value 1. If similar key is already present in the global hash table, then we increment the value by 1. At the end, we prepared a hash table having tag information along with their support count. Finally, we iterate each key-value pair in the hash table and output a tag having value equals to the number of neighbor pages as a *Static Noise Tag*. Table 1 lists examples of some of the content of *Static Noise Tags* identified for the sport category of CNN using 10 neighbor articles.

Table 1: Example of *Static Noise Tags*

Subset of "*Static Noise Tags" content* of CNN Sports Category
Terms of service | Privacy guidelines | Ad choices | Advertise with us | License our content | About us | Contact us | Work for us | Help:
EDITION: INTERNATIONAL U.S. MÉXICO ARABIC TV: CNNi CNN en Español Set edition preference Sign up Log in: Most Popular:
Tools & Widgets | RSS | Podcasts | Blogs | CNN Mobile | My Profile | E-mail Alerts | CNN Shop | Site map | CNN Partner Hotels:
Comments » SHARE THIS Print Email More sharing Reddit StumbleUpon
© 2013 Cable News Network. Turner Broadcasting System, Inc. All Rights Reserved.:

Once, *Static Noise Tags* are identified, we traverse DOM tree of the target web page and mark the node as a noise tag if the node matches with any of the identified Static Noise Tag. For marking a tag as noise tag we add a special attribute "NOISE" to the tag.

---

[1] https://github.com/marek-stoj/NReadability

3.2 Mark Dynamic Noise Tags

Now, we have a DOM tree of the target web page in which *Static Noise Tags* are marked. However, we noticed that there are many noise nodes in DOM tree that have dynamic contents. For example, the node related to "Recent News" section contains hyperlinks that occur together under a <ul> tag. Such tags have high percentage of hyperlink content. We call such type of tags as *Dynamic Noise Tags*. In Figure 3, we present an example of such tag. In this example, "ul" node (in dotted ellipse) is identified as *Dynamic Noise Tag*. Some hyperlinks may also occur under article text nodes, but percentage of hyperlink content is quite low in such article text nodes. Our aim is to identify and mark *Dynamic Noise Tags* in the DOM tree of the target web page.

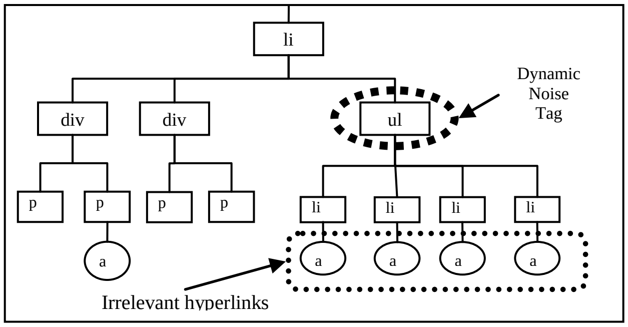


Figure 3: Illustrating irrelevant hyperlinks under *Dynamic Noise Tag* in part of DOM Tree

We apply Least Common Ancestor (LCA) on the output DOM tree of Stage 1 to find the *Dynamic Noise Tags*. Given a DOM tree, we represent each node with tag "a" (i.e., hyper-link tag) by a path-string. A path-string of a node *n* is a path from root to the node *n* in DOM tree along with the position information of each node on the path. In our case, position information of a node is obtained with respect to its parent. For example, consider a path-string for node with tag "p":

0(html)-1(body)-12(div)-0(div)-3(p)

This path string is for representing a "p" tag in the DOM tree since it appears last in the string. Value "3" just before "p" in the string indicates that node "p" is the 4th child of "div" tag which is second last in the string. This "div" tag is 1st child of its parent "div" tag which is 13th child of "body" tag and body tag is 2nd child of "html" tag. Here "html" tag is root tag. The path-strings of tag <a> in Figure 3 is Illustrated in Figure 4. Note that, the path-strings shown in Figure 4 are complete however only a partial DOM tree is shown in Figure 3. As we can see from the Figure 3 and 4, the *Dynamic Noise Tags* are least common ancestor of hyperlink nodes. (html)-1(body)-12(div)-0(div)-3(footer)-0(div)-0(div)-4(ul)-0(li)-2(ul): is the Candidate *Dynamic Noise Tag*.

Thus, after obtaining the set of path-strings for each anchor (<a>)node, we identify least common ancestors of discovered path-strings. These least common ancestors are also nodes in a DOM tree and marked as a C*andidate Dynamic noise Tag*. However, there may be hyperlink nodes inside article text nodes. So some article text node may be identified as *Dynamic Noise Tags* if we follow only LCA method. To avoid marking nodes related to informative content as a *Dynamic Noise Tag*, we further apply two simple filtering strategies for each node that are identified as a *Candidate Dynamic Noise Tag* by Least Common Ancestor method.

1) *Candidate Dynamic Noise Tag* with <p> tag cannot be a "*Dynamic Noise Tag*".

2) A *Candidate Dynamic Noise* is "*Dynamic noise Tag"*

- If the non-link text count (i.e. number of characters including spaces) appears in the node is less than the threshold α. **OR**
- If the link fraction of the node *n*, denoted as LF_n, is greater than β. Here,LF_n is defined as follow:

$$Link\ Fraction\ (LF_n) = \frac{Link\ text\ count\ of\ node\ n}{Total\ text\ count\ of\ node\ n}$$

Here α and β are threshold values for non-link text (no of characters) and Link Fraction respectively. We empirically determine the values α = 50 and β = 20.

This Stage may re-mark some of the nodes of Stage1, but this does not create any problems.

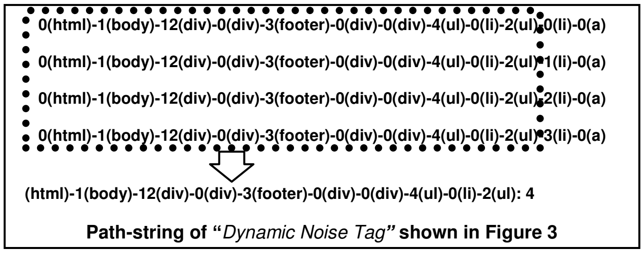

0(html)-1(body)-12(div)-0(div)-3(footer)-0(div)-0(div)-4(ul)-0(li)-2(ul)-0(li)-0(a)
0(html)-1(body)-12(div)-0(div)-3(footer)-0(div)-0(div)-4(ul)-0(li)-2(ul)-1(li)-0(a)
0(html)-1(body)-12(div)-0(div)-3(footer)-0(div)-0(div)-4(ul)-0(li)-2(ul)-2(li)-0(a)
0(html)-1(body)-12(div)-0(div)-3(footer)-0(div)-0(div)-4(ul)-0(li)-2(ul)-3(li)-0(a)
(html)-1(body)-12(div)-0(div)-3(footer)-0(div)-0(div)-4(ul)-0(li)-2(ul): 4
Path-string of "*Dynamic Noise Tag*" shown in Figure 3

Figure 4: Illustration of identifying Dynamic noise nodes by applying LCA on path-strings

3.3 Post processing

After marking of *Static Noise Tags* and *Dynamic Noise Tags*, we are now having a noise marked DOM tree. In post processing we perform two operations: (1) **Remove Noise Marked Tags**: While marking of S*tatic and Dynamic Noise Tags*, DOM tree was marked by adding a new attribute "NOISE" to the tags. So these tags can be found by finding the tags having attribute "NOISE" and removed easily. (2) **Remove non-informative Formatting Tags**: There are some tags which never carry any informative content in them, because they are used for showing images, creating form elements, and formatting, styling and other purposes in html. So these tags are called as "Non-informative Formatting Tags". The list of such tags is shown below:

<img>,<link>,<script>,<style>,<noscript>,<embed>,<object>, <param>,<form>,<input>,<caption>,<meta>,<textarea>.

All the non-informative formatting tags should be removed along with Noise marked tags. Thus after post processing we will have the DOM tree with article text content only. The html web page of this DOM tree consists only article content.

4. EXPERIMENTS

We compare our technique "StaDyNot" with three techniques "Boilerplate[2]", JusText[3], and NReadability. For Boilerplate, we have used the "ArticleExtractor Instance" (as suggested by Boilerpipe API) for extracting the news articles. Our experiments are performed on 440 news article web pages having average corpus size 554 words, collected from 11 different news websites . To cover the variation of formatting and style of a news website, we have gathered about 10 pages from four categories, such as business, entertainment, politics and sports of each web site. Thus, in total we have total 11x4x10 web pages. We have manually prepared the ground truth for each web page in our dataset. While preparation of our ground truth dataset, along with the article text, we have also identified the title, author, date and time of

---

[2] https://code.google.com/p/boilerpipe/

[3] https://code.google.com/p/justext

Table 2: Comparative study of Article Extraction Techniques: StaDyNoT, NReadability, Boilerplate and JusText

News Source	**StaDyNoT**			**NReadability**			**Boilerplate**			**JusText**		
	P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**
Anchorage daily	91.03%	98.80%	**94.68%**	90.82%	93.11%	91.69%	74.93%	95.70%	82.99%	75.05%	90.22%	80.97%
BostonHerald	90.09%	97.78%	**93.70%**	92.60%	93.09%	92.71%	89.37%	94.80%	91.81%	90.79%	87.45%	88.08%
BusinessWeek	92.70%	95.85%	**94.21%**	93.88%	93.81%	93.84%	90.82%	90.49%	90.64%	92.32%	88.29%	90.18%
DNA India	96.64%	99.63%	**98.09%**	87.42%	87.04%	87.16%	97.11%	98.87%	97.94%	92.60%	76.35%	77.85%
Chicago Tribune	90.82%	96.45%	**93.22%**	91.27%	92.57%	91.86%	86.67%	88.50%	87.29%	85.11%	82.34%	83.55%
CNN	72.93%	96.40%	81.82%	80.02%	91.65%	84.53%	73.62%	85.47%	78.67%	64.45%	80.70%	69.44%
DailyMirror (UK)	86.24%	99.19%	**91.26%**	77.89%	79.60%	78.09%	82.77%	84.90%	83.05%	86.48%	47.36%	49.57%
NDTV	88.34%	91.03%	89.56%	89.35%	87.42%	88.36%	89.67%	91.18%	90.36%	86.41%	73.59%	77.01%
BostonGlobe	90.70%	94.05%	**90.92%**	88.25%	91.12%	88.88%	83.89%	79.99%	81.30%	89.12%	87.85%	87.63%
Deccan Chronicle	76.99%	98.57%	83.71%	93.28%	96.34%	94.74%	84.38%	86.61%	85.20%	83.06%	91.23%	81.62%
DenverPost	90.60%	95.97%	**93.08%**	88.40%	90.04%	88.93%	89.50%	92.25%	90.72%	66.70%	82.07%	72.93%
Overall Result	**87.92%**	**96.70%**	**91.29%**	**88.47%**	**90.53%**	**89.16%**	**85.70%**	**89.89%**	**86.92%**	**82.92%**	**80.68%**	**78.59%**

publish as article content. Recall, our algorithm for marking *Static Noise Tags* uses neighbor article web pages for learning. While experimenting in this case, we use the news web pages of same category of a news website as "neighbor article web pages". For example, To mark S*tatic Noise Tags* in sports news article webpage of Boston herald, we will use 9 other pages from the sports category of Boston herald as "neighbor article web pages". Next, we extract articles from each news page and compare it with ground truth. We have utilized the standard evaluation measures, specifically, precision, recall and F1-scores were calculated by comparing the results of each method against the ground truth dataset. Note that, we have represented ground truth and extracted article as a bag of words that take into account the frequency of words without removing any stop-words. Table 2 shows the experiment results of our approach and other techniques.

For each news website, we have compared all the techniques on all the three measures precision (P), recall(R) and F1-score (F1) . Table 2 and Figure 5 shows summary of results. The results show average of precision, recall and F1-score values of all categories in each news website. Starting from poorest performing technique, *jusText* scored low score in all three measures compared to other methods. It indicates that simple heuristic based rules do not perform well on variety of data. Boilerplate technique is not extracting news title, author name and date-time of publish information from most of news web pages of DailyMirror(UK), AnchorageDaily, DeccanChronicle, BostonGlobe, and many other news web pages. *NReadability* is closest competitor of our approach and scored precision value equal to our StaDyNoT approach, but its recall value is 90%, which is low compared to our StaDyNoT approach. *NReadability* is not extracting the date-time and author information from AnchorageDaily, BostonGlobe, BusinessWeek and many other news websites, also many extracted articles missing some article text contents which affects its recall value. We can see in Figure 5 that our approach *StaDyNoT* performed well and scored highest F1-score, and recall value. Our approach has scored a significant 6% higher recall value compared to best performing *NReadability* technique. A good recall value helps web-search-engines for better indexing of extracted news articles.

5. CONCLUSION

In this paper we have proposed an effective approach for cleaning of news web pages by identifying the noise content on the basis of its "static" or "dynamic" nature among its neighbor web pages. We have observed that static noise content repeats itself in neighbor pages while most of the dynamic noise content accumulates under "*Dynamic Noise Tags*". We present a two stage algorithm for effective identification and removal of noise content from news web pages, while majority of other techniques emphasize on identification and extraction of informative content. Due to different strategy, our approach is able to achieve a recall value of 96% in experiments. It reflects that output of our approach contains all the article content in most of web pages with the overall precision of 88% which is quite fair.

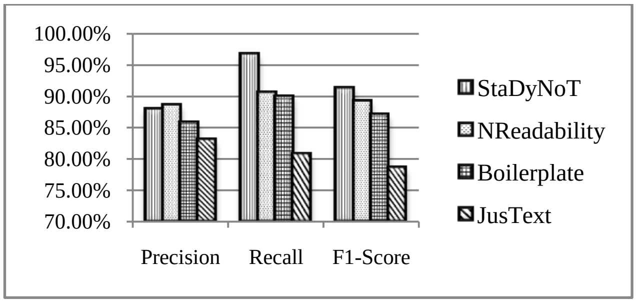


Figure 5: Result comparison among all the techniques

6. REFERENCES

[1] Lan Yi, Bing Liu, and Xiaoli Li. 2003. Eliminating noisy information in Web pages for data mining. *SIGKDD, ACM.*

[2] Jian Fan, Ping Luo, Suk Hwan Lim, Sam Liu, Parag Joshi, and Jerry Liu. 2011. Article clipper: a system for web article extraction. *SIGKDD, ACM.*

[3] J. Pomikalek. 2011. Removing Boilerplate and Duplicate Content from Web Corpora. PhD thesis, Faculty of Informatics, Masaryk University, Brno, Czech Republic,

[4] Fei Sun, Dandan Song, and Lejian Liao. 2011. DOM based content extraction via text density. *SIGIR, ACM* .

[5] Borut Sluban and Miha Grčar. 2013. URL tree: efficient unsupervised content extraction from streams of web documents. *CIKM, ACM.*

[6] Matthew E. Peters and Dan Lecocq. 2013. Content extraction using diverse feature sets. *WWW, ACM* .

[7] Guo, Y., Tang, H., Song, L., Wang, Y. and Ding, G. 201*0.* 'ECON: An approach to extract content from Web News Page'. *APWEB, IEEE.*

[8] Christian Kohlschütter, Peter Fankhauser, and Wolfgang Nejdl. 2010. Boilerplate detection using shallow text features. *WSDM, ACM.*

[9] Dandan Song, Fei Sun, Lejian Liao, 2013. A hybrid approach for content extraction with text density and visual importance of DOM nodes. *KAIS Journal, SPRINGER.*

[10] Zhan Bhu, Chenguchi Zhang, Zengyou Xia, Jiangdong Wang. 2013. An FAR-SW based approach for webpage information extraction. *Inf. Syst. Front. Journal, SPRINGER.*

Transaction support for HBase

Krishnaprasad Shastry
Hewlett Packard
India Software operation
Bangalore-48
+91-8033866316
krishnaprasad.shastry@hp.com

Sandesh Madhyastha
Hewlett Packard
India Software operation
Bangalore-48
+91-8033867304
Sandesh-v.madhyastha@hp.com

Saket Kumar
Hewlett Packard
India Software operation
Bangalore-48
+91-8033868318
Saket.kumar3@hp.com

Kirk M Bresniker
Hewlett Packard
1501 Page Mill Road, Palo Alto
California- 94304-1100, U.S
+1-650 583533
kirk.bresniker@hp.com

Greg Battas
Hewlett Packard
11060 Desert Glen Drive, Fishers
Indiana- 46037,U.S
+1-3178423618
greg.battas@hp.com

ABSTRACT

NoSQL technologies such as HBase, Cassandra, MongoDB are becoming popular due to their ability to scale and handle large volumes of data as opposed to a traditional Relational Database Management System (RDBMS). However they lack two major functionalities provided by traditional RDBMS namely "transactional support" and "SQL interface". Transactions are designed to maintain database integrity in a known, consistent state, by ensuring that interdependent operations on the system complete successfully or all the operations are canceled. This paper describes a non-intrusive approach to provide transaction support for HBase based on optimistic concurrency model.

1. INTRODUCTION

NoSQL technologies such as HBase[4], Cassandra[5], MongoDB[6] are becoming popular due to their ability to scale and handle large volumes of data at breakthrough levels of cost and query performance. However transaction support is lacking in these NoSQL products. Without multi-row, multi-object transaction support in NoSQL products, the application has to implement transactions as part of its business logic. This makes development and maintenance of applications complex.

The workloads such as online transaction processing (OLTP), event processing, real-time analytics, etc. are characterized as operational workload. These workloads typically have stringent requirements in terms transactional data integrity, sub-second response time, concurrency and availability. With the growing "Internet Of Things (IOT)" there is a significant increase in number of data generation sources, volume of data and the type of the data that needs to be captured as part of transactions. These next generation operational applications need transactional support on multi-structured data types. For example, there are several Web2.0 applications, like "online shopping", "online gaming", "online index updates" etc., that require transaction support.

Several of these next generation operational applications will benefit from the flexibility in schema, data types and the scalability of NoSQL products like HBase, Cassandra and MongoDB. But the lack of transaction support is currently preventing from moving to these NoSQL technologies.

There are many attempts in academia as well as the open source community to provide transaction support for NoSQL products. In this paper we describe a non-intrusive approach to provide transaction support for HBase.

2. OUR SOLUTION

The design goal for our solution is to develop a non-intrusive transaction system for HBase. Another aim is to make this solution portable across different NoSQL technologies, thus no functional dependency of HBase.

Our solution provides transaction support to HBase by leveraging the versioning capability in HBase to implement snapshot isolation. The transaction functionality is implemented in a highly available centralized transaction server.

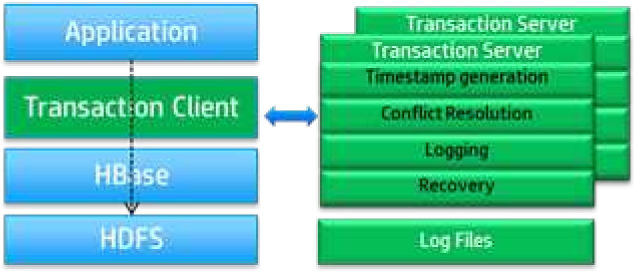


Figure 1: Architecture

Figure 1, illustrates the architecture of the solution. The transaction client provides transaction management APIs like beginTransaction, endTransaction etc. It also extends the generic HBase APIs, like get, put etc. to provide transactional support. The transaction client intercepts the HBase APIs from an application to provide transaction capabilities using the transaction server. It leverages the multi-versioning feature of HBase to write in-process transaction data into HBase tables.

The transaction server generates the transaction IDs, maintains begin and commit timestamp and manages the state of transactions. It implements the logic to resolve the conflicts during the transactions. The transaction server also implements the logging and recovery logic.

The transaction server maintains a table to track the status of transactions. The transaction state can be in (a) begin (b) commit-ready (c) committed (d) aborted. At the beginning of transaction, the transaction server generates globally unique transaction IDs that will have the value less than the epoch value, January 1, 1970. The transaction ID is used as version number for writes (put operation) from in-process transactions. The actual timestamps are used as version number to write the committed records. The committed records will always have the version numbers greater than epoch. We effectively use this data to control the visibility of in-process writes, thus provide snapshot isolation.

The transaction server maintains the timestamp value of the latest committed transaction, which is called as Last Commit Timestamp (LCT). The LCT value is assigned as the start time at the beginning of the transaction and is used to define the snapshot for the transaction.

The transaction server maintains all the modified row keys for a given transaction in an in-memory table. It uses this information to detect the conflict among concurrent transactions. The transaction server uses the transaction start time (TS1) and the modified row key set (RS1) to identify whether any transactions that are committed after the time TS1 has modified any of the rows in RS1. If yes it means the transactions are conflicting. In this case the transaction server marks the transaction for abort.

On receipt of beginTransaction call, the transaction client contacts the transaction server to get transaction ID and LCT. The transaction server generates new transaction ID and adds it into its status table. The transaction client uses the transaction ID as the timestamp value for intermediate "put" operations. These "put" values are also cached in the transaction client.

While reading data, in "get" and "scan" operations, the transaction client first looks for the rows in the cache thus it will read its own changes. If the rows are not in the cache, the "get" and "scan" operations read the data from HBase table using LCT as timestamp value, which indicates the snapshot of the database at the beginning of the transaction.

At the time of commit the transaction client sends the modified records to the transaction server to determine conflicts by other concurrent transactions. The transaction server uses the modified records and the transaction's start timestamp to determine whether any other transactions have modified and committed the same rows. If there are no conflicts it generates a transaction commit timestamp and sends it back to transaction client. The transaction server updates the status of the transaction to commit-ready.

The transaction client performs the final "put" operations using this commit timestamp. After completing the final "put" operations, the transaction client acknowledges transaction server and the server updates the transaction status as committed.

The transaction server updates the LCT value with the commit timestamp of this transaction. At this point the records are committed in HBase table and they are visible to other transactions. If it finds any conflicts then it marks the transaction for abort and updates the status as aborted. The transaction server sends abort information to transaction client. The transaction client aborts the transaction and returns to application. The client will not clean up the intermediate records. The transaction server takes care of this as explained below.

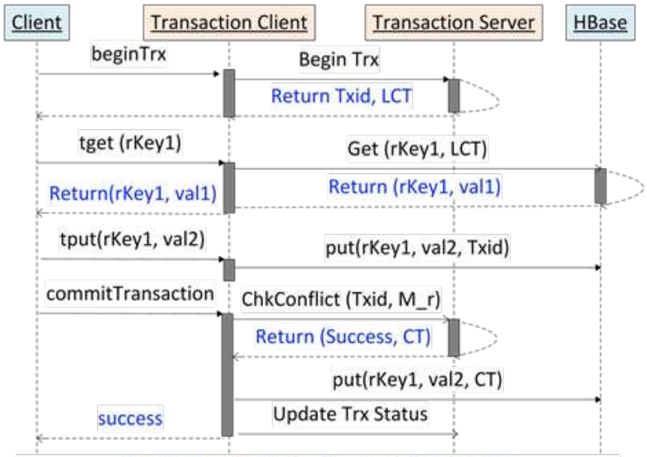


Figure 2: Sequence Diagram for a successful transaction

Figure 2, illustrates the sequence diagram for a successful transaction. During the begin transaction call from application, the transaction client calls transaction server to get the new transaction ID (Txid) and last commit timestamp (LCT). The transaction client intercepts the "tget" call from the application client (also called as client), checks for the corresponding record in its cache, if not found uses the LCT as timestamp to make "get" call to HBase. HBase returns the version of the record (rKey1, val1) that is visible at timestamp LCT. The client then executes the business logic and inserts the updated value (val2). The transaction client inserts this record using the Txid to ensure this intermediate record is not visible to other transactions. At the time of commit the transaction client sends the modified records set (M_r, in this case only rKey1) and the Txid to transaction server for detecting conflicts. In this example, there is no conflict and server returns success with the new commit timestamp (CT). The transaction client uses CT for final "put" operation and on completion acknowledges transaction server. The transaction server updates the transaction status and LCT. The CT becomes the new LCT.

The transaction client will not delete the intermediate records inserted with transaction ID as version. The transaction server implements a "purger thread" to clean up these residual records in the HBase table. The intermediate records modified by the transaction has to be removed for both committed and aborted transactions. The clean-up logic is same for both cases. The purger thread runs in background at specified frequency and deletes the residual records of the committed and aborted transactions.

The transaction servers implement a heartbeat mechanism to determine the client failures. If the client fails during the execution of transaction the server aborts the transaction. If the client fails after the commit-ready state and before commit acknowledgement the server makes the final "put" operation for the records modified by this transaction.

The transaction server will be implemented as process pair to provide high availability. If the primary server fails the backup will take over.

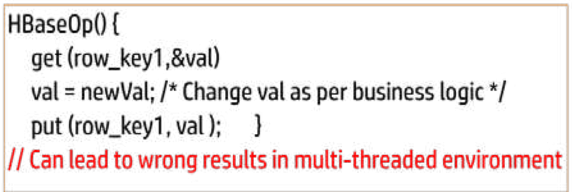
The transaction server logs the transaction state, including order in which they are okayed for commit into persistent space for recovery. The key of the records modified by the transaction are also logged for the commit-ready transactions. During the recovery of transaction server it reads the log to determine the transaction status. For the transactions that are in commit-ready state the server builds the list of records modified by this transaction from the log and inserts them into the HBase table with the transactions commit time. For the in-process transactions it cleans up the intermediate records by using the transaction id.

Our solution is non-intrusive and modular. It is not tightly coupled to HBase implementation. The transactions are supported both for the new tables as well as existing tables. The existing client applications must be modified to use the new APIs provided by transaction client to support transactions. The solution can be easily extended to other NoSQL products with minimal changes to transaction client.

3. EVIDENCE THE SOLUTION WORKS

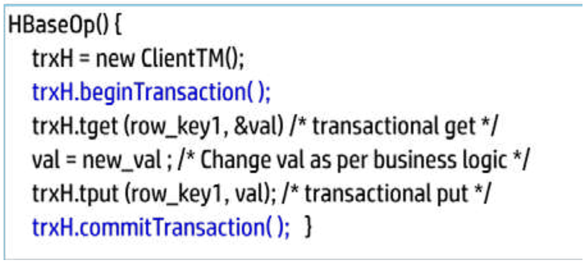
We have implemented the transaction server and transaction client. Tested it with a generic transactional application to validate the functionality.

Listing 1 and 2 shows the code snippet for a sample HBase application and the same application with transaction support.



```
HBaseOp() {
   get (row_key1,&val)
   val = newVal; /* Change val as per business logic */
   put (row_key1, val );     }
// Can lead to wrong results in multi-threaded environment
```

Listing 1: Code sample without transaction support



```
HBaseOp() {
   trxH = new ClientTM();
   trxH.beginTransaction( );
   trxH.tget (row_key1, &val) /* transactional get */
   val = new_val ; /* Change val as per business logic */
   trxH.tput (row_key1, val); /* transactional put */
   trxH.commitTransaction( );  }
```

Listing 2: Code sample with transaction support

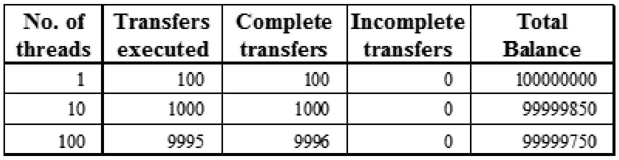
The transactional application creates an instance of transaction client (TClient) and uses the methods exposed by it to create and commit the transaction.

Also it uses the "get" and "put" methods extended by transaction client. Without transaction support the sample application produces wrong results in multi-threaded environment.

To validate this we have developed a sample debit/credit application. The application operates on a single accounts table that contains the account identifier and the balance amount. The application transaction (a) deducts a fixed amount from one randomly selected account and (b) deposits the same amount in another randomly selected account. The application does some basic checks to ensure the debit and credit accounts are different, the balance never goes below a minimal value etc. The correctness of the transactions is measured by calculating the total sum of the value in all the accounts.

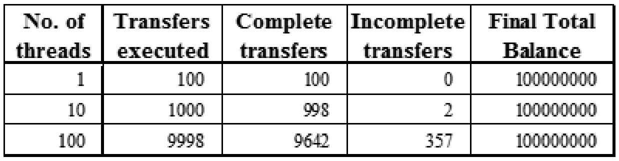
The application is tested with two different modes, (i) Transaction mode – wherein the application is linked with newly developed Transaction Client and uses the transaction API's offered by it, (ii) Non Transaction mode – wherein the application directly uses the API's exposed by HBase. The application is run with different configurations by varying (1) Total number of transactions (2) The number of simultaneous transactions (or threads).

We have run the application with multiple different values for (i) number of threads and (ii) number of transactions in a thread. The tables 1 and 2 show the results for multiple threads with each thread executing 100 transactions.



No. of threads	Transfers executed	Complete transfers	Incomplete transfers	Total Balance
1	100	100	0	100000000
10	1000	1000	0	99999850
100	9995	9996	0	99999750

Table 1: Without transactions



No. of threads	Transfers executed	Complete transfers	Incomplete transfers	Final Total Balance
1	100	100	0	100000000
10	1000	998	2	100000000
100	9998	9642	357	100000000

Table 2: With transactions

The first column represents the number of parallel threads; the second column indicates the total number of transaction that is tried. As mentioned earlier each thread runs 100 transactions. If the random number generator generates the same number for both debit and credit accounts then that transaction is not tried. Hence we see the value in this column to be less than the expected value for some cases – example row 3 (with transactions) has the value of 9998 instead of 10000. Third column indicates the number of completed transfers in case of default HBase and the number of committed transactions in the case of HBase-trx. The fourth column provides the number of failed transfers in the default HBase and number of aborted transactions in the case of HBase. The fifth column provides the total sum of balance in all the accounts.

As we can see form the table 1, there are inconsistencies in default HBase when run with parallel threads. The final balance is not same as the original and indicates the data loss/corruption due to failures. In case of "with transactions" (table 2) the transaction support guarantees the consistency and we always get the correct balance. We also see some failed transactions, indicated by "incomplete transfers" column, which are due to the conflicts.

4. COMPETITIVE APPROACHES

There are few attempts to provide transaction support for HBase. They fall into two main categories. One approach is to implement the transaction support on client side. HAcid [1] and HBaseSI [2] are two examples of this. They rely on additional metadata tables being created in HBase.

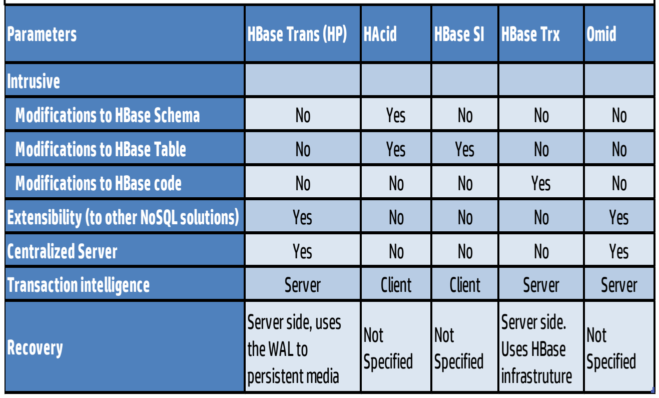
HAcid [1] is implemented as client library. It modifies the user tables in HBase to store additional metadata related to transaction management. Concurrency issues are handled at the client library by using the metadata information stored in user tables.

HBaseSI [2] is a client library that maintains special tables in HBase for supporting transactions. The transaction management logic is implemented in the client side based on the metadata in HBase tables.

The other approach is to implement a centralized transaction server. Omid [3][7] is an example of this, which is quite similar to our approach. Omid uses “transaction status Oracle” to manage the transactions. Omid caches the transaction metadata on client side to improve the performance. This results in multiple copies of metadata and increases the data traffic between client and server. Also maintaining the metadata adds additional overhead. The Omid clients cache the intermediate modification and hence need larger memory for long running transaction. This helps them to reduce the number of “put” operations.

HBase-Trx [8] was another open source attempt from Apache group to support transactions for HBase, which was later discontinued. HBase-Trx is tightly coupled to HBase and hence leverages the HBase code for transaction management and recovery. The concurrency is handled by HBase-Trx server library, which is implemented as an extension to HBase Region. The table 3 summarizes the characteristics of each of the solution.

Table 3: Comparison Table



Parameters	HBase Trans (HP)	HAcid	HBase SI	HBase Trx	Omid
Intrusive					
Modifications to HBase Schema	No	Yes	No	No	No
Modifications to HBase Table	No	Yes	Yes	No	No
Modifications to HBase code	No	No	No	Yes	No
Extensibility (to other NoSQL solutions)	Yes	No	No	No	Yes
Centralized Server	Yes	No	No	No	Yes
Transaction intelligence	Server	Client	Client	Server	Server
Recovery	Server side, uses the WAL to persistent media	Not Specified	Not Specified	Server side. Uses HBase infrastruture	Not Specified

5. Conclusion

We have presented a reliable and efficient implementation of transaction support library for HBase which is non-intrusive in nature. The approach does not need any changes to HBase schema or tables. It is implemented as a light weight centralized transaction server which provides the transaction management, conflict detection, logging and recovery services. A light weight transaction client library exposes the transaction support to users through transactional APIs. We have evaluated the approach for correctness.

As next steps we will measure the performance implications of the newly introduced transaction server and optimize it for both the latency and through-put.

6. References

[1] Mederos, A.:HAcid: A lightweight transaction system for HBase. Master’s Thesis, Espoo, September 24, 2012, Aalto University, School of Science, Degree program of Computer Science and Engineering

[2] Zhang, C., Sterck, H.D.:HBaseSI: Multi-row distributed transactions with strong snapshot isolation on clouds, Scalable Computing: Practice and Experience, Scientific International Journal for parallel and Distributed computing, Vol 12, No 2, 2011.

[3] Ferro, D.G.:Omid:Efficient Transaction Management and Incremental Processing for HBase, Yahoo Inc..

[4] HBase: http://hbase.apache.org/

[5] Cassandra: http://cassandra.apache.org/

[6] MongoDB: http://www.mongodb.org/

[7] Junqueira, F., Reed, B., Yabandeh, M.:Lock-free Transactional Support for Large-scale Storage Systems:IEEE/IFIP 41st International Conference, Dependable systems and Networks Workshop, Pages 176-181, June 2011..

[8] HBase-trx, https://github.com/hbase-trx, Git-Hub.

Hades: A Hadoop-based Framework for Detection of Peer-to-Peer Botnets

Pratik Narang, Abhishek Thakur, Chittaranjan Hota
Dept. of Computer Science and Information Systems,
Birla Institute of Technology and Science-Pilani, Hyderabad Campus,
Hyderabad, India
{p2011414, abhishek, hota}@hyderabad.bits-pilani.ac.in

ABSTRACT

This paper presents `Hades`, a Hadoop-based framework for detection of P2P botnets in an enterprise-level network, which is distributed and scalable by design. The contributions of this work are two-fold: Firstly, our work uses the Hadoop-ecosystem to adopt a 'host-aggregation based' approach which aggregates behavioral metrics for each Peer-to-Peer (P2P) host seen in network communications, and uses them to distinguish between benign P2P hosts and hosts infected by P2P botnets. Secondly, we propose a distributed data-collection architecture which can monitor *inside-to-inside* LAN traffic, as opposed to relying solely on the NetFlow information available at a backbone router which cannot see the LAN communications happening in the network.

1. INTRODUCTION

In the past few years, botnets have emerged as the largest threat to modern networks. With enterprise-level networks regularly generating billions of events and gathering Terabytes of data each day, tracking malicious activity inside a network is nothing less than the proverbial *needle in the haystack* problem. The evolution of Peer-to-Peer (P2P) based botnets, which have a distributed and decentralized architecture, has created further challenges in their detection.

Traditional methods of botnet detection used signature-based or port-based approaches [11]. Such techniques were rendered useless by botnets which did not operate over fixed ports or used encryption. Although a lot of research has gone into detection and take-down of botnets (P2P or otherwise), not much work has been seen which identifies botnets based on their network behavior.

Detecting P2P botnets is a challenging task because P2P botnet traffic can very easily blend with benign P2P traffic in a network, like that of Skype, BitTorrent, eMule etc. Although some past work evaluated the detection of P2P botnets in Internet traffic [4], the detection of P2P botnets in the presence of benign P2P traffic is a challenging scenario. Furthermore, scalable detection approaches have received very little attention (such as in [10]).

Most of the previous work utilizing network behavior of botnets uses the traditional 5-tuple flow-based analysis of network traces. The 'flow-based' approach classifies packets on the basis of the 5-tuple: `source IP, destination IP, source port, destination port, protocol`. Some of the recent work has used 5-tuple based flow analysis along with supervised [7] and unsupervised [10] machine learning techniques or other statistical measures [9] in order to separate P2P botnet traffic from benign traffic.

The 'flow' information is typically obtained in the form of Cisco's NetFlow (or by using tools like Argus[1]) from a backbone router of an enterprise. Large-scale networks may involve multiple border routers. The NetFlow data collected at one router will not give a complete picture of the communications which happened to and from the network. A distributed data collection approach – where data collectors sit closer to the nodes in the network – can give a much better view of the communications. Such a distributed approach is especially beneficial and essential for the detection of smart P2P bots *inside* the perimeter of a network, which talk to each other and send upgrades to themselves on LAN in a P2P fashion, and limit communication to the outside world via one or two peers only. The activity of such bots, which communicate to each other on LAN in a P2P fashion, cannot be detected by traditional 'flow-based' approaches which only monitor the data crossing the backbone router(s).

Flow-based approaches are known to suffer from another drawback [6] that many modern P2P applications and P2P botnets randomize their port numbers and switch their communication between TCP and UDP. The classical 'flow' definition relies of port numbers and protocol, and cannot give the true picture of the communications that the P2P hosts inside the network are engaged in.

In this work, we present `Hades`, which is an acronym for '**H**ost-**a**ggregation based **de**tection **s**ystem' for P2P botnets. `Hades` utilizes the distributed computing power of the Hadoop ecosystem to parse large network traces and extract 'behavioral' features for every P2P host seen in network communications. The extracted feature-set is then used to train supervised machine learning models which can differentiate P2P botnets from P2P applications. `Hades` does not require signature-based detection approaches, Deep Packet Inspection (DPI) or a 'seed' information of bots obtained

The 20th International Conference on Management of Data (COMAD),
17th-19th Dec 2014 at Hyderabad, India.

---

[1] http://qosient.com/argus/

from a blacklist of IPs. `Hades` just relies on the header information in the network and transport layer, and extracts statistical features which quantify the 'P2P' behavior of different kinds of P2P applications.

`Hades` addresses certain limitations of past works and makes the following contributions:

1. `Hades` is built on top of the Hadoop ecosystem, which is a *de facto* standard for big data analytics. Since it utilizes the power of distributed computing through Hadoop [2], `Hades` is scalable by design.

2. We propose a distributed data collection architecture wherein data collectors are distributed at multiple locations inside an enterprise network and sit close to the peers, say at an Access switch or a Wi-fi access point. This approach allows *inside-to-inside* communication view, which can be vital for detecting P2P botnets inside a network which communicate to each other over LAN.

3. `Hades` does not rely on the traditional 'flow' definition. We adopt a Host-aggregation based approach which obtains statistical features *per host* for all P2P hosts involved in network communications.

In order to facilitate reproducible research, we also discuss the implementation aspects of `Hades` in detail.

2. SYSTEM DESIGN AND IMPLEMENTATION DETAILS

The system design of `Hades` employs the `libpcap` library for collecting and parsing network traces. It utilizes the Hadoop ecosystem for aggregation of host-based data and for building scalable models for the detection of P2P botnets. `Hades` has been implemented on top of the Hadoop ecosystem with the open-source projects of Apache Hive [8] and Apache Mahout [1]. The system architecture of `Hades` is given in Figure 1.

2.1 Distributed Data Collection

Instead of relying upon NetFlow data obtained at a backbone router, `Hades` proposes a distributed data-collection technique wherein data collectors sit close to the peers inside the network perimeter. As mentioned above, this allows `Hades` to have a view of the conversations which happen inside-to-outside (or vice-versa) as well as inside-to-inside.

The implementation of `Hades` has multiple data collectors distributed inside the network perimeter. The initial deployment of the system has data collectors deployed at Wi-fi access points within the University campus of the authors. The multiple data collectors consist of commodity-grade hardware machines with 2 GB RAM, 2 CPU cores and 200 GB of disk space. The Wi-fi access points used are NetGear N150 and Belkin N150. Each data collector uses a `libpcap` library based module to capture traffic in the form of network traces `.pcap` files. Each data collector runs an automated parser module (built with `libpcap` library and Python) which parses the network traces and extracts packet-level features of interest to us. Features are extracted from the IP header and TCP/UDP header, and no DPI is required. For this work, the features extracted from each packet are:

1. Time-stamp of the packet

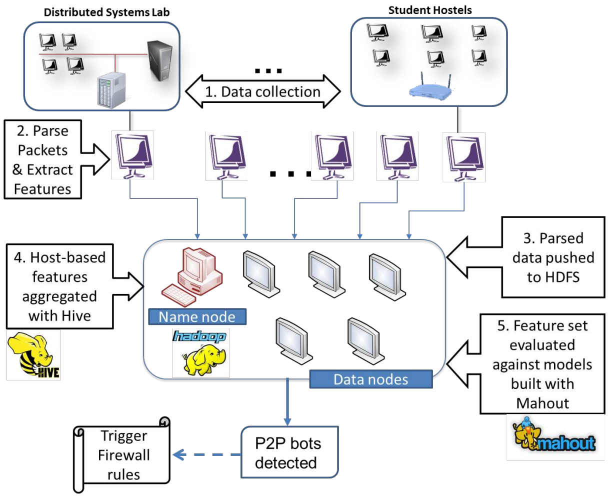

Distributed Systems Lab
Student Hostels
1. Data collection
2. Parse Packets & Extract Features
3. Parsed data pushed to HDFS
4. Host-based features aggregated with Hive
HIVE
Name node
hadoop
Data nodes
5. Feature set evaluated against models built with Mahout
mahout
Trigger Firewall rules
P2P bots detected

Figure 1: Hades: System Architecture

2. Source IP

3. Destination IP

4. Time-to-live (TTL) value

5. Transport layer protocol (TCP/UDP)

6. TCP or UDP payload length (as applicable)

The extracted features are stored in a `.csv` file at each data collector. Instead of transferring large `.pcap` files, these `.csv` files are periodically transferred from all data collectors to the Hadoop Distributed File System (HDFS) [3]. For the purpose of data sanitization, all packets which are found to *not* contain a valid IPv4 header are removed (E.g- corrupted packets). Presently `Hades` does not support IPv6. The present approach of `Hades` also disregards all packets corresponding layers below the IP layer, such as ARP broadcast messages. The implications of this choice will be further discussed in section 4.

2.2 Host data aggregation

The Hadoop cluster deployed for `Hades` consists of a 'name node' Virtual Machine with 8 GB RAM, 8 CPU cores and 200 GB disk space, and ten 'data node' Virtual Machines, each having 2 GB RAM, 2 CPU cores and 200 GB of disk space. Each Virtual Machine runs Ubuntu 12.04 Operating System.

Packet-level data obtained from multiple data collectors is aggregated *per host* for every host seen in network communication. The packet-level data is stored in Hive in the form of external tables. Hive commands are written in HQL (Hive query language) which is very similar to SQL [8]. The Hive command used to create the table for storing packet-level data is given here:

```
CREATE EXTERNAL TABLE packet_data (
    timestamp DECIMAL, ip_source STRING,
    ip_destination STRING, ttl INT,
    proto INT, payload_length INT )
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
LOCATION '/user/hdfs/PacketDump';
```

For the task of detecting P2P botnets, we aggregate the following statistical features over a time-period T (say, one hour) for every P2P host inside the network:

1. **Number of distinct destination hosts contacted:** P2P hosts are involved in sharing content and downloading download different chunks of a file from different peers across the globe. A benign P2P host might be involved in downloading a certain file (or its chunk) from Adelaide, uploading a file to another peer at Birmingham, and download music content from a peer at California. As a normal user of the Internet engaged in P2P file sharing, a benign P2P host is expected to contact a number of peers in the different parts of the world, with no specific pattern involved in the destinations contacted. Moreover, due to the sheer size of these networks, most interactions in P2P file sharing networks are one-time transactions, where peers who share content with each other may never interact again. But, the behavior of hosts infected by P2P botnets gives a contrast. Bot-peers do not engage in file sharing or downloads. Rather they regularly and repeatedly contact their set of bot peers to receive or propagate commands and updates. Thus the number of unique destination hosts contacted by bot-peers are expected to be less as compared to benign P2P hosts.
2. **The total volume of data sent from the source host:** As stated above, benign P2P hosts are engaged in file transfers, downloads, uploads etc. whereas bot hosts are not expected to be engaged in these activities. The volume of data sent from a benign host is, quite clearly, expected to be more than the exchange of data seen at bots which are primarily involved in exchange of Command & Control information.
3. **The average of the TTL value of the packets sent from the source host:** A user of P2P file sharing systems who is involved in downloading some music content will not bother whether the seeding peers happen to be from his home country or some other part of the world. Rather, while the user might himself be situated in India, he may download one part of the file from a peer in China, another chunk from a peer in Holland, and another from a peer in Australia. Since the file requests of benign P2P users travel all over the world, these requests typically have high TTL values associated with them. In contrast, bot hosts tend to repeatedly contact their set of bot-peers. For the sake of efficient design and avoiding latency/overheads, botmasters would not want their bots to talk to peers in different parts of the world. Bots are expected to engage in communication with other bot peers near to them. This leads to the requests sent by bots having lower TTL values when compared to requests seen from benign P2P hosts.

The host-aggregated features described above are stored in another table:

```
CREATE TABLE host_data (
    host STRING, destinations DECIMAL,
    avg_ttl DECIMAL, volume BIGINT )
ROW FORMAT DELIMITED FIELDS TERMINATED BY ‘,’
LINES TERMINATED BY ‘\n’ STORED AS TEXTFILE;
```

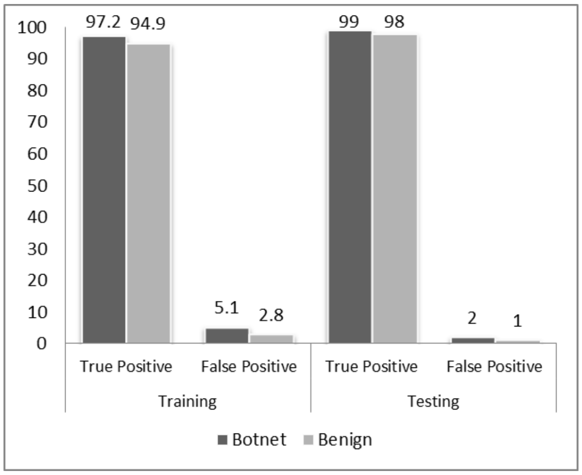


Figure 2: True Positive rate and False Positive rate with training and testing data for Random forests of ten trees

Since the packet-level data arrives from different data collectors periodically, a Hive script is run periodically to convert it into host-aggregated form and store it in the table `host_data` created above. The Hive script given below uses a ‘GROUP BY’ operation to obtain data in host-aggregated format:

```
INSERT INTO TABLE host_data
SELECT ip_source, COUNT (DISTINCT ip_destination),
       AVG(ttl), SUM(payload_length)
FROM packet_data
GROUP BY ip_source;
```

2.3 Detecting P2P bots from hosts

The host-based features extracted above are used to train and test supervised machine learning models. Apache Mahout is used for this purpose. Mahout is a fairly new tool, and at present does not offer many machine learning algorithms. Further, many of Mahout's algorithms (for classification and clustering) do not run as MapReduce jobs. Parallelized implementations are important for scalability of `Hades` over large datasets. Thus, for this work, we stick to the Random Forest implementation of Mahout which is a parallelized implementation (in contrast to other implementations like Linear regression, AdaBoost etc., which are not). More details on the data used are given in the next section.

Results generated from `Hades` can be used to alert a network administrator for suspicious activity in the network, trigger rules to a Firewall, and/or log or drop botnet traffic. This way `Hades` can be used by network administrators as an assisting tool which is ‘P2P-aware’.

3. EVALUATION AND RESULTS

For evaluation of `Hades`, we use P2P data obtained from a recent work at the University of Georgia [7]. Our dataset consists of network traces of two P2P applications, namely Vuze and Frostwire, and two P2P botnets, namely Storm and Waledac. The data of P2P applications was generated by Rahbarinia et al. by running these applications in their lab environment for a number of days. The data of P2P botnets was obtained by them from third-parties, and corresponds to real-world traces of these botnets.

The size of this dataset was around 20 GB (14 GB for Vuze and FrostWire, and 6 GB for Storm and Waledac) in `.pcap` format, and around 10.5 GB (6.6 GB for Vuze and FrostWire, and 3.9 GB for Storm and Waledac) when parsed to `.csv` format.

After extracting host-based features from each application, we created a 'labeled' dataset. Instances belonging to P2P hosts (Vuze or Frostwire) are labeled 'benign', while the instances belonging to P2P bots (Storm or Waledac) are labeled 'malicious'. This dataset was split into training and testing data in 2:1 ratio. With the training data, Random Forest models were built for different number of trees in each run. The models were then evaluated for their accuracy with the test data. Due to limitations of space, we only present the results for number of trees in the forest equal to 10. Figure 2 shows the accuracy obtained for the training and testing data with a Random Forest of 10 trees. Our system could detect bot-infected hosts with a True Positive rate of 97% and 99%, and a low False Positive rate of 5% and 2% over training and testing datasets respectively.

4. DISCUSSION & POSSIBLE EVASIONS

It was explained in Section 2 that `Hades` ignores messages below the IP layer, such as ARP broadcast messages. Its implications will be discussed here. In the Introduction, we argued on the case of 'smart' P2P bots which may exchange C & C with peers on a LAN and limit communication with the outside world to one or two peers. A bot-master may also configure such smart bots to utilize protocols lower than the IP layer – such as ARP messages – to facilitate communication between the bots on the same LAN. `Hades` will not be able to detect the communication of such bots since it does not deal with those messages in its present approach. Although no past work has touched upon this issue and no such botnets are known to exist at present, we argue that with the evolution of botnet detection mechanisms, bot-masters will also improvise their botnets in these ways to make them more efficient and harder to detect.

Further, the present approach of `Hades` is limited to detection of bots when bots and apps are *not* running on the *same host*. If a host which is running P2P applications is also infected with a bot, `Hades` will be unable to correctly classify it as an infected host.

5. CONCLUSIONS AND FUTURE WORK

This work presented `Hades`, an approach to collected P2P data inside a network in a distributed manner, and extract host-aggregated features to distinguish between P2P applications and botnets using supervised machine learning approaches. To the best of our knowledge, `Hades` is the first attempt at distributed data collection for the detection of P2P botnet traffic. The distributed data collection architecture proposed by us gives *inside-to-inside* visibility of traffic. With such an approach, `Hades` attempts to target the detection of 'smart' P2P bots. However, such botnets are not known to exist at present[2], and thus no network traces corresponding to such behavior could be evaluated. We plan to evaluate `Hades` on such data by generating synthetic botnet data.

Acknowledgments

The authors would like to thank their teammates G. Sharath Chandra and Kamaldeep Singh for their efforts on Hadoop set-up and deployment.
This work was supported by Grant number 12(13)/2012-ESD for scientific research under Cyber Security area from the Department of Information Technology, Govt. of India, New Delhi, India.

6. REFERENCES

[1] Mahout: Scalable machine-learning and data-mining library. `http://mahout.apache.org`, Accessed on 30 November 2013.

[2] A. Bialecki, M. Cafarella, D. Cutting, and O. O MALLEY. Hadoop: a framework for running applications on large clusters built of commodity hardware. *Wiki at http://lucene.apache.org/hadoop*, 11, 2005.

[3] D. Borthakur. The hadoop distributed file system: Architecture and design, 2007. *Apache Software Foundation*, 2011.

[4] J. François, S. Wang, R. State, and T. Engel. Bottrack: Tracking botnets using netflow and pagerank. In *Proceedings of the 10th International IFIP TC 6 Conference on Networking - Volume Part I*, NETWORKING'11, pages 1–14. Springer-Verlag, Berlin, Heidelberg, 2011.

[5] L. O. Murchu. Stuxnet p2p component. `http://www.symantec.com/connect/blogs/stuxnet-p2p-component`. Accessed on 12th February 2014.

[6] P. Narang, C. Hota, and V. Venkatakrishnan. Peershark: flow-clustering and conversation-generation for malicious peer-to-peer traffic identification. *EURASIP Journal on Information Security*, 2014(1):1–12, 2014.

[7] B. Rahbarinia, R. Perdisci, A. Lanzi, and K. Li. Peerrush: Mining for unwanted p2p traffic. In *Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 62–82. Springer-Verlag, Berlin, Heidelberg, 2013.

[8] A. Thusoo, J. S. Sarma, N. Jain, Z. Shao, P. Chakka, S. Anthony, H. Liu, P. Wyckoff, and R. Murthy. Hive: a warehousing solution over a map-reduce framework. *Proceedings of the VLDB Endowment*, 2(2):1626–1629, 2009.

[9] T.-F. Yen and M. K. Reiter. Are your hosts trading or plotting? telling p2p file-sharing and bots apart. In *Proceedings of the 2010 30th International Conference on Distributed Computing Systems*, ICDCS '10, pages 241–252. IEEE, 2010.

[10] J. Zhang, R. Perdisci, W. Lee, X. Luo, and U. Sarfraz. Building a scalable system for stealthy p2p-botnet detection. *Information Forensics and Security, IEEE Transactions on*, 9(1):27–38, 2014.

[11] Z. Zhu, G. Lu, Y. Chen, Z. Fu, P. Roberts, and K. Han. Botnet research survey. In *Computer Software and Applications, 2008. COMPSAC'08. 32nd Annual IEEE International*, pages 967–972. IEEE, 2008.

---

[2] With the exception of Stuxnet [5], which we ignore here since it targets SCADA systems, and evaluating its detection with Internet traffic would not be possible

DEMOS

RootSet: A Distributed Trust-based Knowledge Representation Framework For Collaborative Data Exchange
(Demo Paper)

Chinmay Jog
International Institute of Information Technology Bangalore,
26/C, Electronics City,
Bangalore India
jog.chinmay@iiitb.org

Sweety Agrawal
International Institute of Information Technology Bangalore,
26/C, Electronics City,
Bangalore India
sweety.v.agrawal@iiitb.org

Srinath Srinivasa
International Institute of Information Technology Bangalore,
26/C, Electronics City,
Bangalore India
sri@iiitb.ac.in

ABSTRACT

In a collaborative setting, several organizations need to exchange data with each other. Trust management is a major hindrance for inter-organizational collaborations. Organizations show reluctance towards replication of data outside their own datacenters. This work demonstrates a distributed trust-based platform for data exchange, called RootSet, among several collaborating organizations. RootSet enables organizations to share data without replication. It provides *credential-based access control* to manage data access by users throughout the platform using access rules. This work showcases several features of this tool.

Keywords: Knowledge Representation, Collaborative Data Exchange, Trust Management, Distributed Knowledge Management

1. INTRODUCTION

Collaborations between organizations frequently occur in domains like healthcare, agriculture and governance to carry out variety of tasks. Most of these collaborations involve trust-based data exchange. While some collaborations may use open datasets, others may use sensitive, private information, especially in case of healthcare projects. TRUMP[1] is one such multi-organization healthcare project, which aims to build a trusted mobile platform for self-management of chronic illness in rural areas. This initiative requires to minimize the risks in sharing private information like health records[3]. In such cases, collaborators may wish to expose only specific information that is related to the collaboration. As in case of TRUMP, collaborators may also wish to restrict the access to only some other collaborators. Also, issues like legal compliance and data sensitivity may put restrictions on physical location of the data, hindering collaboration efforts. For example, government organizations are typically reluctant to replicate data outside the jurisdiction of their nation. This problem of selective data sharing without replicating data, is a very big impediment in forming many collaborations.

In this paper, we present a distributed trust-based knowledge management framework called RootSet, that enables users to share data selectively, in a collaborative setting. RootSet is based on a knowledge model called Many Worlds on a Frame(MWF)[6], which allows users to store data inside a semantic boundary called the world. Data access is controlled by defining rules on the world. In collaborations, all the collaborators can have their own MWF servers, which are interconnected to form a distributed MWF grid. Using it, the collaborators can share their data with other collaborators, without replicating it out of their data centers. Many cloud based integrity management solutions like TrustStore[8] provide a trust-layer over the existing cloud storage. These cloud storage solutions may not be able to address the restrictions based on jurisdictions. Therefore, these may still be unusable for organizations that have a binding on the physical locations of the datacenters. RootSet addresses this problem by distributing the MWF servers at the datacenters of collaborators.

The rest of this paper is organized as follows – section 2 describes the RootSet framework, followed by its features in section 3. Section 4 discusses the implementation in brief, and section 5 discusses the conclusions and future work.

2. ROOTSET

RootSet has three components – The core MWF model, the grid manager, and the access manager. MWF is the context-aware knowledge modelling framework. The grid manager is responsible for managing the MWF grid. The access manager is responsible for managing the credentials based access control. In this section, we briefly discuss about each of these components.

---

[1] `http://www.trump-india-uk.org/`

The 20th International Conference on Management of Data (COMAD),
17th-19th Dec 2014 at Hyderabad, India.

2.1 Many Worlds on a Frame(MWF)

Modelling context-sensitive knowledge has been a challenge for knowledge representation frameworks[6]. The MWF model addresses this problem by providing semantic boundary called *world*, that contains knowledge elements belonging to that semantic context. Each world has a type and a location. The worlds are semantically interconnected to each other using these type and location hierarchies. These semantic interconnections form the Frame in MWF. The type and the location relationships are transitive, reflexive and anti-symmetric. The type hierarchy captures the generalization-specialization semantics. Through the type hierarchy, worlds can inherit properties. The location hierarchy captures the containment semantics. Thus, through the location hierarchy, container worlds impose privileges and control access over all the contained worlds.

Roles and Associations. In MWF, worlds host data in form of roles, and associations between roles. The roles are played by role-players, which are other worlds. Associations represent the relationships between the roles. In figure 1, Person, Subject and College are worlds. Person plays a role of Student and Faculty Member in College. Student and Faculty member roles are associated with each other by the *Teaches* association. Roles and associations have attributes of their own, which form the *schema* or the structure of a world. The role players and the associations between them form the *data*.

2.2 Distributed MWF

The distributed MWF is a conglomeration of many standalone MWF servers, that are fused together to form a single frame that can span over the network. The distributed MWF is useful especially in collaborative settings, where all the collaborators can host their data on their own MWF servers. The MWF servers can then come together to form a grid, where each collaborator can decide on their own access rules to allow users access to their data.

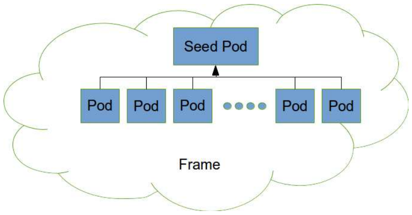


Figure 2: An MWF Grid

Pods and the MWF Grid. An MWF server is called a Pod. Each pod ships with its own database, and can connect to a grid. A grid is a set of MWF pods which are fused together in one Frame. In the grid, one pod acts as the seed pod. All other pods connect to the grid using the URI of the seed pod. This is depicted using figure 2. Other than joining the grid, no other communication needs to go through the seed pod. Therefore, the seed pod does not function as a bottleneck or a single point of failure for the grid.

Home location and Proxy of a World. In a collaboration, worlds may be extended from or placed in other worlds which are defined on other pods in the grid. In such scenarios, to avoid frequent accesses to the remotely located worlds, we may need to cache some information related to the world locally. This locally stored information represents the *proxy* of the world. It is a read-only copy of the world, in which the structure and the privilege information are cached. It does not cache any data that is stored in the world. The *home location* of the world is where the world can be edited. Any changes to the structure can only be performed at the home location. Any changes in the type or location hierarchy can also be done at the home location. While we can extend a world type from its proxy, we cannot create a new proxy for an existing proxy. A world has a unique home location, but may have any number of proxies. However, each pod may have at most one proxy of a given world.

Consistency in Distributed MWF. RootSet guarantees causal consistency on the data. Causal consistency is defined over all the world structures as well as percolation of privileges across the grid of MWF servers. By causal consistency, we mean that if an event e has occurred before an event $e\prime$ on a world w, then all the proxies of the world w will also reflect the same consequence in the same order. For example, if two users u_1 and u_2 simultaneously modify a world w, and u_1 revokes privilege of u_2 to change the world before u_2 commits the change, then only changes made by u_1 are committed. The changes made by user u_2 are not committed. All proxies of world w are notified of both the changes, and the changes reflect in the same order at all the proxies.

2.3 Credentials based access control

Traditionally, trust in terms of data exchange has been modelled on the basis of access control models. Various access control mechanisms including identity-based access[1, 5, 7], role-based access[4], credentials-based access[6] have been discussed in literature. RootSet provides an access control mechanism based on the roles played by a user in different worlds[2]. A detailed comparison between credentials based access control with other access control mechanisms is given in [2]. The credentials-based access control is based on a set of privilege rules[2]. The credentials of a user are the set of roles played by the user in different worlds. An access rule is defined using a set of credentials which when satisfied grants a privilege package. The privilege rules are of the form $PrivilegePackage_i \leftarrow r_1(t_1, l_1), r_2(t_2, l_2), \ldots, r_n(t_n, l_n)$, where r_i is a role defined in a world of type t_i and located in world l_i. A user satisfies a privilege rule when his participation set includes all the $r_i(t_i, l_i)$ tuples for the given privilege rule. On satisfying a privilege rule, a user gets a privilege package. A privilege package is a 5-bit binary string. The bits represent, in order, the five operations that a user can perform – Visibility, Privilege, Frame, Structure and Data. A visibility privilege grants read-access to a user on a world. Privilege-level privilege grants a user to add, modify or delete the privilege rules for a world. Frame level privilege grants access to modify the type and containment lineage, as well as add new worlds in the containment lineage. Structure level privilege grants access to add, modify or delete roles and associations in a world. Data level privilege grants access to add or delete role instances and association instances. Apart from these, each

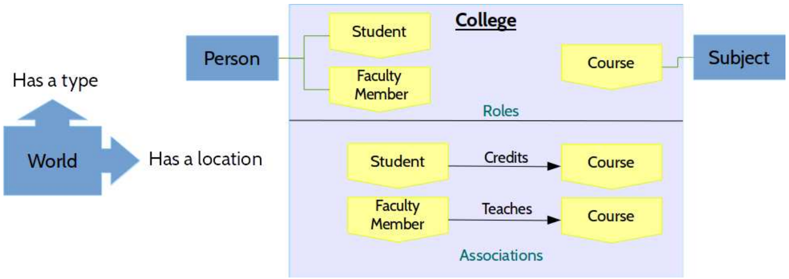
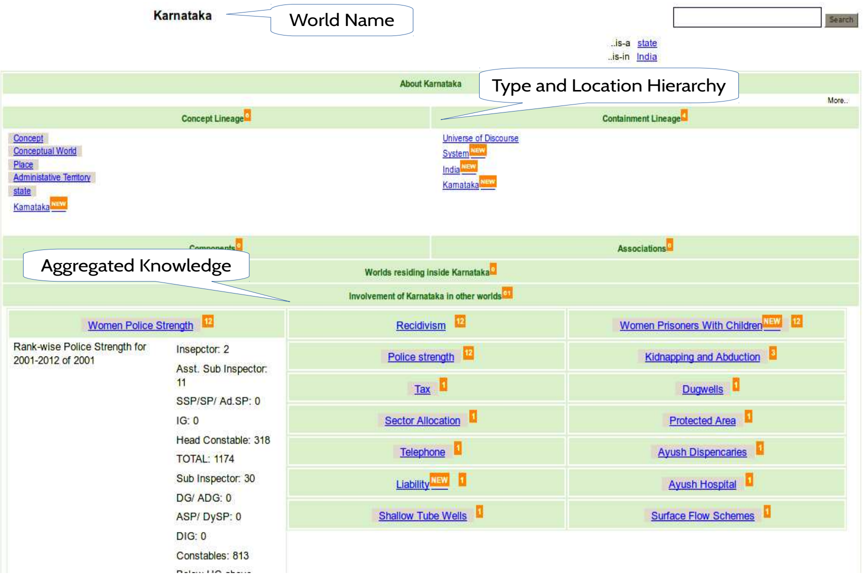


Figure 1: Many worlds on a frame



Rank-wise Police Strength for 2001-2012 of 2001

Insepctor: 2
Asst. Sub Inspector: 11
SSP/SP/ Ad.SP: 0
IG: 0
Head Constable: 318
TOTAL: 1174
Sub Inspector: 30
DG/ ADG: 0
ASP/ DySP: 0
DIG: 0
Constables: 813

Figure 3: Knowledge aggregation in Worlds

world also has a set of administrators. Administrators have the all privileges except visibility of data. A world also has a default rule. Default rule is a rule which is applied when a user does not satisfy any of the privilege rules defined for the world.

3. FEATURES

In this section, we discuss the key features of RootSet that are critical for a collaborative cross domain inter-organizational data exchange. We outline the planned and implemented features of RootSet.

Aggregation of Knowledge. A main feature of RootSet is data aggregation. If a world participates in various other worlds, it aggregates all the data, and makes it available for users at one place. This can be seen in figure 3.

Exporting worlds. RootSet provides a mechanism to export a world in XML and HTML formats. Using this feature, users can extract context specific data in an aggregated form. The export utility exports the lineage, structure and the data contained in the world.

Provenance. RootSet provides a way to trace back all the changes committed for a world, along with the privilege packages used to perform the changes. All the changes are logged in the database. Thus, a user can look back at the exact sequence of events to trace back a particular change.

Bulk load. RootSet provides a way to bulk load data and ontologies. Given mappings for each of the entity types in a structured file, it has the ability to search for worlds, create required worlds, and upload data into them. This feature helps in migrating from an existing ontology based tool onto RootSet.

4. IMPLEMENTATION

RootSet uses the Model-View-Controller architecture in its implementation. It's prototype has been built using the Ruby on Rails framework. It uses Rails 4 framework and Ruby 2.1. HTML 5 and the bootstrap Javascript framework powers its user interface. It also uses *d3.js* to display charts. It uses *sqlite3* as the database engine. It supports google authentication. It uses the Apache Solr indexing engine to locally index the worlds. The Solr server powers a full-text search capability in the implementation.

5. CONCLUSIONS AND FUTURE WORK

RootSet has been proposed to enhance multi-organizational collaborations. Some key features have been discussed. It mainly addresses the problem of sharing data in a collaborative setting. It nullifies a key challenge of giving away data outside an organization's datacenters for enabling collaborations. Implementing constraint based data-validation, exporting worlds into standard RDF representations, and a stronger risk-aware credentials based authorization are planned for future work.

6. REFERENCES

[1] S. Agarwal, B. Sprick, and S. Wortmann. Credential based access control for semantic web services. In *AAAI Spring Symposium-Semantic Web Services*, volume 1, 2004.

[2] S. Agrawal, C. Jog, and S. Srinivasa. Integrity management in a trusted utilitarian data exchange platform. In *To appear- 13th International Conference on Ontologies, DataBases, and Applications of Semantics (ODBASE 2014)*. Springer, 2014.

[3] C. Burnett, P. Edwards, T. Norman, L. Chen, Y. Rahulamathavan, M. Jaffray, and E. Pignotti. Trump: A trusted mobile platform for self-management of chronic illness in rural areas. In M. Huth, N. Asokan, S. apkun, I. Flechais, and L. Coles-Kemp, editors, *Trust and Trustworthy Computing*, volume 7904 of *Lecture Notes in Computer Science*, pages 142–150. Springer Berlin Heidelberg, 2013.

[4] D. F. Ferraiolo and D. R. Kuhn. Role-based access controls. *arXiv preprint arXiv:0903.2171*, 2009.

[5] A. J. Lee. Credential-based access control. In *Encyclopedia of Cryptography and Security*, pages 271–272. Springer, 2011.

[6] S. Srinivasa, S. Agrawal, C. Jog, and J. Deshmukh. Characterizing utilitarian aggregation of open knowledge. In *Proceedings of First ACM IKDD Conference on Data Sciences, Delhi, March 2014*, pages 789–796. ACM Digital Library, 2014.

[7] S. D. C. D. Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, G. Psaila, and P. Samarati. Integrating trust management and access control in data-intensive web applications. *ACM Transactions on the Web (TWEB)*, 6(2):6, 2012.

[8] J. Yao, S. Chen, S. Nepal, D. Levy, and J. Zic. Truststore: Making amazon s3 trustworthy with services composition. In *Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, pages 600–605. IEEE Computer Society, 2010.

Akshaya: A Framework for Mining General Knowledge Semantics From Unstructured Text
(Demo Paper)

Sumant Kulkarni*
International Institute of Information Technology Bangalore,
26/C, Electronics City,
Bangalore India
sumant.k@iiitb.org

Srinath Srinivasa
International Institute of Information Technology Bangalore,
26/C, Electronics City,
Bangalore India
sri@iiitb.org

Priyanaka Shukla
International Institute of Information Technology Bangalore,
26/C, Electronics City,
Bangalore India
priyanka.shukla@iiitb.org

ABSTRACT

We report a tool called *Akshaya*, which implements a framework to mine four types of "general knowledge semantics" (analytical semantics) from unstructured text. The semantics being mined are - *semantic siblings, topical anchors, topic expansion* and *topical markers*. The framework provides options to embed more such general knowledge semantic mining algorithms into it. We use a term co-occurrence graph representation of unstructured text corpora to mine these semantics relations between terms. The semantic mining algorithms use different graph algorithms like random walk, graph clustering and so on to mine semantic relations. The tool can currently read plain text documents and generate a term co-occurrence graph and perform semantic association mining on it.

Keywords: Analytical Semantics, General Knowledge Semantics, Text Semantics, Text Mining, Semantic Siblings, Topical Anchors, Topic Expansion, Topical Markers

1. INTRODUCTION

Rachakonda et.al [6] have proposed methods to mine four forms of general knowledge semantic associations from co-occurrence graphs. Akshaya is a tool developed to build a comprehensive framework to mine these analytical semantics from a given unstructured text corpus. An example of an analytical semantics is a semantic relation like "is of type" which connects a given concept to its *type* concept. For example, given a concept like "car", and asked for its *type*, the semantic mining algorithm is expected to generate "vehicle" as output.

Mining general knowledge semantics has widespread applications in health care, patent management, judiciary and various other domains that deal with textual data. For example, it can be used in print media industry to link different documents using labeled semantic relations. We can link two documents discussing similar topics by a relation called "discusses similar topics". Similarly, we can cluster semantically similar documents using these approaches. We can use semantic mining to semi-automate the workflows in all such domains where extensive text data is available.

There have been many attempts to mine such analytical semantic relationships earlier. The major ones are [2, 3, 4, 5, 6, 7]. We have derived our inspiration to develop Akshaya from [2, 5, 6]. There are four semantic associations listed in [6], which form the basis for Akshaya. They are listed below.

Semantic Siblings is an algorithm which takes a set of sibling concepts as input and identifies other similar sibling concepts.

Topical Anchors is an algorithm, which takes a set of concepts as input and produces the concept which represents the topic of the input.

Topic Expansion generates a hypothetical context, which captures the predominant sense of the usage of the given input term in the corpus. It also separates out different contexts where the term assumes different meanings.

Topical Markers algorithm takes a concept as input and generates another term which uniquely and predominately identifies the input concept.

Table 1 gives hypothetical examples for the input and outputs for each semantic relationship mining algorithm. The detailed explanation of the algorithms can be found in [6].

Akshaya intends to mine these semantic relationships from unstructured text corpora. It converts the given unstructured text corpus into a term co-occurrence graph and runs these semantic mining algorithms for the given input terms. The semantic relationships between concepts are specific to the corpus being used. We currently use Wikipedia corpora of years 2006 and 2011 to mine these semantics. However, we can also use any other unstructured human generated corpus for the same.

---

*We thank Paras Mittal and Dipesh Joshi for their help in implementation.

The 20th International Conference on Management of Data (COMAD),
17th-19th Dec 2014 at Hyderabad, India.

Algorithm	Input	Output
Semantic Sibling	*Apple, Orange, Banana*	*Papaya, Mango, Pear, Sweet Lime*
Semantic Sibling	*Pele, Maradona, Messi, Ronaldo*	*Beckham, Zidane, Rooney, Kaka*
Topical Anchor	*Pele, Penalty Shootout, Offside*	*Football*
Topic Expansion	*Tennis*	*Nadal, Court, Wimbledon, Grand Slam, Open, Service, Ace, Volley*
Topical Marker	*Cricket*	*Googly*

Table 1: Example Input and Results of the Four Algorithms on a Hypothetical Text Corpus.

2. THE ALGORITHMS

In this section, we briefly describe each semantics association mining algorithm. The intention here is not to get into an in-depth understanding of the algorithms. Instead, we focus on understanding just the idea behind each algorithm. The algorithms have been discussed in detail in [6].

Semantic Siblings. The semantic siblings algorithm intends to identify the concepts which play the same role as the given concepts in given contexts. But the identified concepts are not synonyms of the given concepts. Table 1 shows two examples of semantic siblings. The core idea behind semantic sibling identification is "replaceability". A semantic sibling of given terms can replace them in predominant contexts common to the input, without distorting the meaning of those contexts. For example, in the statement - "Pele scored a goal", we can replace Pele with any of Beckham, Zidane, Rooney, Kaka. Hence, they are all semantic siblings of Pele. The semantic sibling algorithm attempts to identify the replaceability for the concepts in corpus. There are two approaches used to identify the replaceability. The details of the two algorithms are given in [6].

Topical Anchor. Topical anchor of a set of concepts represents the topic of the coherent context where the concepts appear predominantly. Table 1 gives a hypothetical example of a topical anchor. The topic is the concept whose probability of generation is the most, in the coherent context of the given concepts. In other words, topic is the most central concept to the conversation involving input terms. For example, if there is a conversation which mentions "glucose, blood sugar, insulin, hypoglycemia", then the term "diabetes" becomes the most expected (central) term in the conversation. Hence, it is the topic of the conversation. Authors use cash leaking random walk on the term co-occurrence graph to identify the topic of the given input concepts. There are three different varieties of the topical anchors algorithm. The details of the algorithms are given in [6].

Topic Expansion. Topic expansion is a process of expanding a given concept into different hypothetical contexts which represent the predominant usage of the concept in corpus. In text corpus, where polysemy exists, topic expansion generates many clusters for different senses of input term. For example, the term "Java" can have two hypothetical topic expansions as given below.

1. Java, Object Oriented Programming, Class, Object, Interface, Byte code, JVM, Compiler, ...
2. Java, Jakarta, Javanese, West Java, Central Java, East Java, Banten, Indonesia, Island, Yawadvipa, Population Density, ...

Topic expansion algorithm uses clustering based techniques on the term co-occurrence graph to generate such clusters of terms. There are two types of topic expansion algorithms. The details of the algorithms are given in [6, 2].

Topical Markers. Topical markers of an input concept are the concepts which are unique to the topic of input and are very unlikely to be related to other topics. For example, *double fault* is a topical marker of *tennis*. A term t_m which is well known and is very specifically used with another term t in the corpus becomes the topical marker of t. The authors use HITS algorithm [1] on the term co-occurrence graph to identify the topical markers. The co-occurrence graph is duplicated to create a bi-partite graph on which HITS algorithm is run. The details of the algorithms are given in [6].

All the above mentioned semantic association mining algorithms require a term co-occurrence graph representation of unstructured text corpus. They take concepts as input and generate other concepts which are connected by the mentioned association to the input terms.

3. THE TOOL

Akshaya is a library which allows us to perform following tasks.

1. Load data to create term co-occurrence graph.
2. Query the primitives of co-occurrence graph mentioned in [6].
3. Query for the documents containing a given term.
4. Query for the tf-idf scores of a given term for all documents where it is present.
5. Perform semantic association mining given a set of terms.

To perform the above mentioned tasks, Akshaya maintains two data structures – a term co-occurrence graph and an inverted index. We extract noun phrases from the text using statistical noun phrase extraction techniques. The extracted noun phrases are used to build a co-occurrence graph as explained in [6]. The graph is used for querying co-occurrence primitives and also to perform semantic association mining. Akshaya maintains an inverted index for all the documents used to create co-occurrence graph. The inverted index is used to answer queries related to document retrieval and tf-idf scores. Akshaya currently supports four semantic association mining algorithms discussed earlier. It also allows addition of similar semantic association mining algorithms, which take a set of terms as input, work on the term co-occurrence graph and generate terms as output.

Akshaya is a ruby gem shipped with a command line interface. The Agama graph store [1] is used to store the co-occurrence graph. It uses SQLite [2] to store the inverted index. The Akshaya command line tool lets us run all semantics mining algorithms. Any algorithm bundled in Akshaya can be run using a generic command of the following form.

```
$ <algorithm-name> <options> <terms>
```

An example command to generate topic expansion results for the term “corpus” is given below.

```
$topicexpansion -s fast -d similar -c
wiki2006 corpus
```

The flag `-s fast` tells that it should be a fast execution and `-d similar` tells that the clusters generated can be similar. The flag `-c wiki2006` tells the algorithm to run topic expansion on Wikipedia 2006 corpus. The term `corpus` is the query term. Results of topic expansion on 2006 Wikipedia corpus for the term “corpus” are given below.

1. corpus, habeas corpus, eighth amendment, healthy americans act, theodore marley brooks, andrew blodgett mayfair, patricia savage, william, harper littlejohn, section 1983, thomas j. roberts

2. corpus, brown corpus, part-of-speech tagging, george kingsley zipf, empirical law, derose, ambiguous, native speaker, word sense disambiguation, parts of speech

3. corpus, hippocratic corpus, hippocratic cap-shaped bandage, medicine in ancient greece, project hippocrates, risus sardonicus, pyopneumothorax, the hippocrates project, 370 bc, clinical medicine

4. CONCLUSION AND FUTURE WORK

In this work, we have explained a generic semantic association mining framework called Akshaya. Akshaya is a ruby gem, which lets us mine different semantic associations like semantic siblings, topical anchors, topic expansion and topical markers. It uses term co-occurrence graph representation of the given corpus to mine semantics. It also gives the flexibility to add more such algorithms to the library. The future work includes adding other semantic mining algorithms like [3]. We also intend to develop this into a complete web application to perform end-to-end tasks – from corpus upload till semantic association mining.

5. REFERENCES

[1] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)*, 46(5):604–632, 1999.

[2] S. Kulkarni, S. Srinivasa, and R. Arora. Cognitive modeling for topic expansion. In *On the Move to Meaningful Internet Systems: OTM 2013 Conferences*, pages 703–710. Springer, 2013.

[3] S. Kulkarni, S. Srinivasa, J. N. Khasnabish, K. Nagal, and S. G. Kurdagi. Sortinghat: A framework for deep matching between classes of entities. In *Data Engineering Workshops (ICDEW), 2014 IEEE 30th International Conference on*, pages 90–93. IEEE, 2014.

[4] R. Navigli and M. Lapata. An experimental study of graph connectivity for unsupervised word sense disambiguation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(4):678–692, 2010.

[5] A. R. Rachakonda and S. Srinivasa. Finding the topical anchors of a context using lexical cooccurrence data. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 1741–1744. ACM, 2009.

[6] A. R. Rachakonda, S. Srinivasa, S. Kulkarni, and M. Srinivasan. A generic framework and methodology for extracting semantics from co-occurrences. *Data & Knowledge Engineering*, 2014.

[7] B. Wei, J. Liu, J. Ma, Q. Zheng, W. Zhang, and B. Feng. Motif-re: motif-based hypernym/hyponym relation extraction from wikipedia links. In *Neural Information Processing*, pages 610–619. Springer, 2012.

---

[1] https://github.com/arrac/agama

[2] http://www.sqlite.org/

SortingHat: A Deep Matching Framework to Match Labeled Concepts
(Demo Paper)

Sumant Kulkarni*
International Institute of Information Technology Bangalore,
26/C, Electronics City, Bangalore India
sumant.k@iiitb.org

Srinath Srinivasa
International Institute of Information Technology Bangalore,
26/C, Electronics City, Bangalore India
sri@iiitb.org

ABSTRACT

We report a framework called *SortingHat* to perform semantic matching between labeled concepts in a partially labeled corpora such as workflow data. We create a labeled term co-occurrence graph as the representative data-structure of the given corpus. The semantic matching between concepts is performed using a variant of random walk algorithm on the term co-occurrence graph. The SortingHat system takes a set of concepts as input and generates a semantically matching set of concepts for them. In this experiment, we use data from bug tracking system of a large enterprise to demonstrate the results.

Keywords: Semantic Matching, Labeled Concept Matching, Deep Matching, Text Mining, Concept Matching

1. INTRODUCTION

Semantic matching can be defined as the identification of semantically related objects of a domain for a given set of input objects in the same domain based on the latent semantics. Some examples of such semantic matching problems and their domains are given in table 1.

Sl. No	Semantic Matching Problem	Domain
1	Identification of the resolvers for an issue in software industry	Software
2	Identifying the best customer care executive for the given type of complaints like online banking login failure	Banking
3	Identification of the best tourist destination for given calendar month	Tourism

Table 1: Examples of Semantic Matching Problems.

Large amounts of textual data like bug tracking system snapshot, banking customer care logs, and tourist review logs for the destinations get generated in the above mentioned domains. We can use these text corpora to solve the kind of problems listed in table 1. Solving these problems help to speed up the processes in many such domains. Hence, semantic matching is considered an important problem in text domain.

Semantic matching in text corpora aims to identify set of semantically related concepts for given set of concepts. Many times the textual data contains additional labeled information, like *types* associated with the data values, which can be used in semantic matching process. There also can exist some data values for which labels do not meaningfully capture the type. We call such a label as *pseudo-type*. Together, this kind of data is called "partially labeled data". For example consider a customer care system in banking domain. Table 2 shows hypothetical data collected over several transactions. This log contains data like complaint id, customer id, resolver, priority, which have *type* information. However, there can be additional data columns like summary of complaint which are *pseudo-types*.

SortingHat aims to perform semantic matching for such partially labeled data. An example of semantic matching between concepts on the data in table 2 could be – identifying the best resolvers for a complaint related to "credit card" account type which also has "high" priority. Kulkarni et.al [1] proposed a term co-occurrence graph based approach to perform the concept matching. The approach includes creating a co-occurrence graph of labeled and unlabeled terms and running a "cash leaking" random walk on it to generate the matching output concepts. There have been earlier attempts to use random walk to mine different kinds of semantics. In [2, 3], authors propose a "cash leaking" random walk to identify the topical anchor of a set of terms. Yazdani et.al [4] present a random walk based framework for semantic similarity calculation.

The SortingHat implementation is inspired from [1]. We have been using a snapshot of a bug tracking dataset from a big enterprise. The implementation aims at identifying the best resolvers who can take up newly reported bugs. The system also extends its functionality in identifying other

---

[*] We thank Paras Mittal and Dipesh Joshi for their help in implementation.

The 20th International Conference on Management of Data (COMAD),
17th-19th Dec 2014 at Hyderabad, India.

Comp ID	**Cust ID**	**Resolver**	**Priority**	**Account Type**	..	**Conversation Summary**	..
aaa	xxxx	Anand	High	Corporate	..	Faced issues with online access of account information. System was giving error 307. Requested customer to login after 3 hours.	..
bbb	yyyy	Sahana	Medium	Credit Card	..	Communication address needs to be changed. Requested employee to send mail to customer care email id with new address proof.	..

Table 2: Data from a hypothetical banking customer care system.

important aspects of a new bug like component, probable priority and so on.

2. THE METHOD

As discussed earlier, semantic matching in text domain is the process of identifying a set of concepts R for a given set of concepts Q in a given corpus based on the latent semantics. Identifying the probable defaulters from a banking transaction dataset, identifying defective components using the server logs, identifying an application software similar to a given application software based on the description and reviews are example of semantic matching.

This work uses partially labeled data as the corpus. Let C be the set of concepts in the corpus. There are some concepts which are identifiable with labels (or have attached type information). We label the unlabeled concepts with special label "concepts". The set of all unique labels in the corpus is represented as L.

We build a co-occurrence graph using C. In this graph, every labeled concept co-occurs with every other labeled concept in the same context. This co-occurrence graph is formally represented as,

$$G = (C, E, w, L, \gamma) \tag{1}$$

where, $E \subseteq [C]^2$ is pairwise co-occurrences of labeled concepts. $\gamma : C \to L$ represents a functions which assigns label types to concepts. The function $w : E \to Z^+$ assigns edge weight $w(t, u)$ which is same as the co-occurrence count, where $t, u \in C$. This edges are undirected. We convert the graph into generatability graph, using the procedure explained in [3].

The SortingHat algorithm takes a set of labeled concepts $Q = \{q_1, q_2, \ldots, q_m\}$ as input and generates semantically matching labeled concepts $R = \{r_1, r_2, \ldots, r_p\}$ as result from the co-occurrence graph G. For each q_i, the algorithm generates a set of neighbors $N(q_i)$. A new set $N(Q^*)$, called as *neighbourhood closure*, is generated by taking union of all these set of neighbours. We generate the *semantic context* $S(Q^*)$ by considering all the edges between all the concepts in $N(Q^*)$. The details of the calculation of generatability, neighborhood closure and semantic context are available in [1].

To identify the semantically matching concepts R for Q, we run "cash leaking" random walk algorithm [2]. We start the random walk by distributing equal cash to each query concept and zero cash to every other concept in $S(Q^*)$. After reaching stationary distribution, we identify concepts with higher cash accumulation and output them as R. These are the concepts which are semantically related to the set of labeled query concepts Q.

3. THE TOOL

The SortingHat tool is developed in Java environment. The noun phrases are extracted using the Apache openNLP [1]. We generate the co-occurrence graph and store it in MySQL [2]. We are also parallely attempting to use Neo4j [3] to store co-occurrence graph. We have developed this tool as a web application. Figure 1 shows the webpage to input the query. The input concepts are entered as $\langle term \rangle$: $\langle type \rangle$. For example, *performance:concept* represents a concept named *performance* of type *concept*.

The tool runs the "cash leaking" random walk using the input and generates the semantically matching output for it. The output is displayed in the form of a table, where each column represents a type, as shown in figure 2. We have intentionally blanked out some columns from the output screen shot to hide sensitive information. The concepts in the columns are ranked as per their semantic relatedness to the query.

Entity	**Description**
Project	The name of the project.
Issue	The Issue Id of the issue.
Resolver	The person who resolved the issue.
Resolution	State of the issue after resolution (Not a Bug, Duplicate, Fixed etc.,).
Component	Name of the project component.
Time Taken	Time taken to fix issue.
Issue Priority	Importance of the issue.
Classification	Type of the fix (API, Code etc.,).
Root Cause	Reason for the issue to arise.

Table 3: The different labeled concepts in bug tracking dataset.

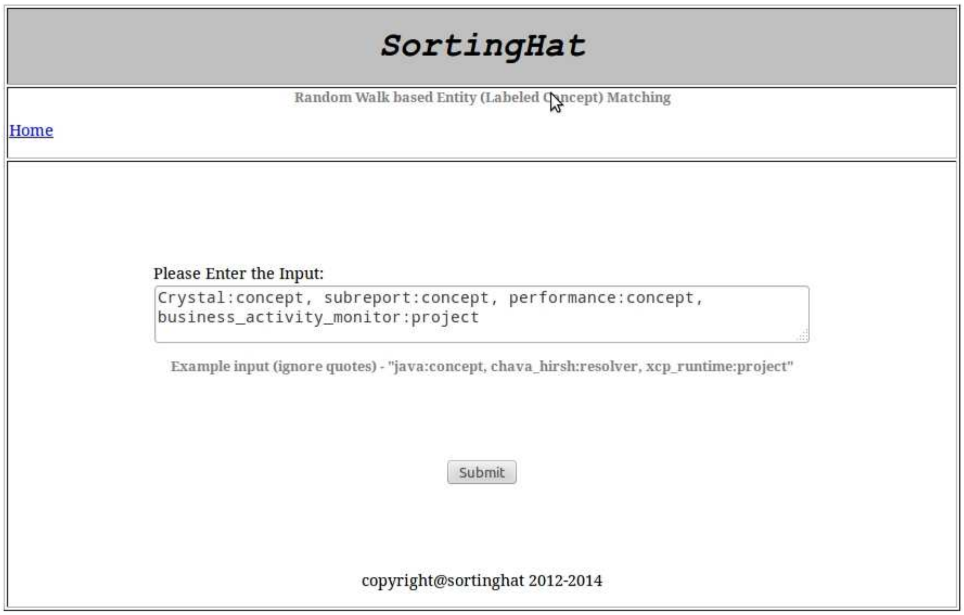
As mentioned earlier, we are currently using a bug tracking system snapshot as the partially labeled dataset. We have manually identified 9 types(lables) [1] in the dataset as given in table 3. We similarly identified 3 fields in the

---

[1] https://opennlp.apache.org/

[2] http://www.mysql.com/

[3] http://www.neo4j.org/



SortingHat

Random Walk based Entity (Labeled Concept) Matching

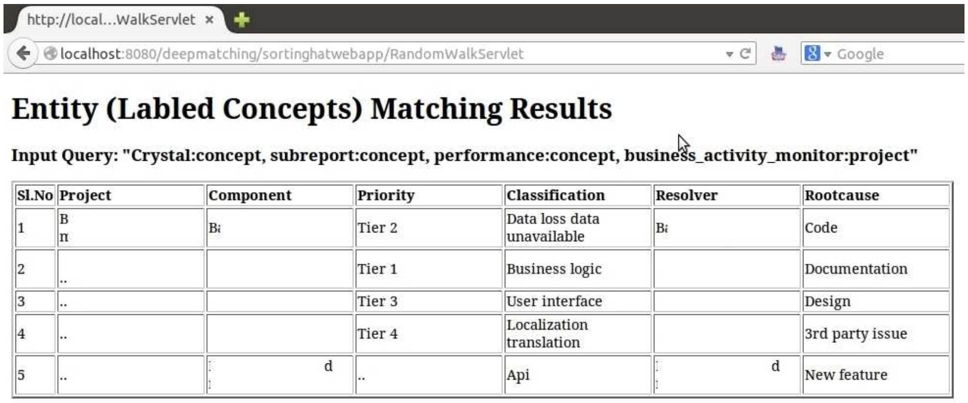
Please Enter the Input:

```
Crystal:concept, subreport:concept, performance:concept,
business_activity_monitor:project
```

Example input (ignore quotes) - "java:concept, chava_hirsh:resolver, xcp_runtime:project"

copyright@sortinghat 2012-2014

Figure 1: Input Screen of SortingHat Web Application


http://local...WalkServlet
localhost:8080/deepmatching/sortinghatwebapp/RandomWalkServlet
Google

Entity (Labled Concepts) Matching Results

Input Query: "Crystal:concept, subreport:concept, performance:concept, business_activity_monitor:project"

Sl.No	Project	Component	Priority	Classification	Resolver	Rootcause
1	B n	B	Tier 2	Data loss data unavailable	B	Code
2	..		Tier 1	Business logic		Documentation
3	..		Tier 3	User interface		Design
4	..		Tier 4	Localization translation		3rd party issue
5	..	d	..	Api	d	New feature

Figure 2: Output Screen of SortingHat Web Application

dataset from where "concepts" can be extracted. They are given in table 4.

The primary semantic matching problem in this data is to identify the best suited *resolver* for a given new issue, for which the information of *Project, Priority, Summary* and *Description* is available. This would help the managers to identify the resolver more efficiently. We can also extend this to identify the *component* of the issue. For both the use cases, the inputs are the values of *Project, Priority*, and *concepts* from unstructured fields - *Summary* and *Description*. The input contains comma separated concepts, where each concept has its type information separated by ":". An example query is given in figure 1. Further, we can also use this framework to identify many other semantic similarities like projects similar to other projects, resolvers similar to other resolvers and so on.

The SortingHat framework can handle many other types of generic semantic matching queries on given textual dataset. It can take any coherent set of concepts and generate semantically matching set of concepts for it. There are domains like insurance claim management, bank customer grievance resolution, and many other where such partially labeled textual data is available. There is a need to identify different labeled concepts like *claim resolver* for the given inputs like *insurance claim*. In such cases, SortingHat can come in handy as a supporting tool.

4. CONCLUSION AND FUTURE WORK

SortingHat is a tool developed to identify semantically related concepts for a given set of concepts. The tool addresses semantic matching in text domain. Many domains generate textual data, and hence this tool can play an important role in semi–automating large number of important tasks in those domains. The future work includes the de-

Column Name	**Description**
Summary	Short description of the issue.
Description	Detailed description of the issue.
Customer Facing Description for Release Notes	The message passed to the customer to convey the existence of the issue. It describes how the customer can observe the issues.

Table 4: The different fields contributing to concept space in JIRA dataset.

velopment of a functionality to identify similar concepts of same *type* for given input concept. This would help to suggest more number of semantically similar concepts for the input query.

5. REFERENCES

[1] S. Kulkarni, S. Srinivasa, J. N. Khasnabish, K. Nagal, and S. G. Kurdagi. Sortinghat: A framework for deep matching between classes of entities. In *Data Engineering Workshops (ICDEW), 2014 IEEE 30th International Conference on*, pages 90–93. IEEE, 2014.

[2] A. R. Rachakonda and S. Srinivasa. Finding the topical anchors of a context using lexical cooccurrence data. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 1741–1744. ACM, 2009.

[3] A. R. Rachakonda, S. Srinivasa, S. Kulkarni, and M. Srinivasan. A generic framework and methodology for extracting semantics from co-occurrences. *Data & Knowledge Engineering*, 2014.

[4] M. Yazdani and A. Popescu-Belis. A random walk framework to compute textual semantic similarity: a unified model for three benchmark tasks. In *Semantic Computing (ICSC), 2010 IEEE Fourth International Conference on*, pages 424–429. IEEE, 2010.