2018 Imperial College Computing Student Workshop

ICCSW 2018, September 20–21, 2018
London, United Kingdom

Edited by

Edoardo Pirovano

Eva Graversen

Editors

Edoardo Pirovano
Department of Computing
Imperial College London
e.pirovano17@imperial.ac.uk

Eva Graversen
Department of Computing
Imperial College London
e.graversen16@imperial.ac.uk

ACM Classification 2012
General and reference → General conference proceedings

ISBN 978-3-95977-097-2

Published online and open access by
Schloss Dagstuhl – Leibniz-Zentrum für Informatik GmbH, Dagstuhl Publishing, Saarbrücken/Wadern, Germany. Online available at http://www.dagstuhl.de/dagpub/978-3-95977-097-2.

Publication date
January, 2019

Bibliographic information published by the Deutsche Nationalbibliothek
The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available in the Internet at http://dnb.d-nb.de.

Digital Object Identifier: 10.4230/OASIcs.ICCSW.2018.0

OASIcs – OpenAccess Series in Informatics

OASIcs aims at a suitable publication venue to publish peer-reviewed collections of papers emerging from a scientific event. OASIcs volumes are published according to the principle of Open Access, i.e., they are available online and free of charge.

ISSN 2190-6807

http://www.dagstuhl.de/oasics

Contents

Main Track

Poster Track

Invited Talk

2018 Imperial College Computing Student Workshop (ICCSW 2018).
Editors: Edoardo Pirovano and Eva Graversen
OpenAccess Series in Informatics
OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Preface

Welcome to the 2018 Imperial College Computing Student Workshop (ICCSW18), the seventh workshop in its series. ICCSW was initiated with a "by students, for students" spirit: it is a workshop organised by Imperial students and aims to give students of all universities a chance to present their research work. The organising students gain valuable experience in all the steps of organising a conference – from writing a call for papers to chairing a session. On the other hand, the participating students benefit from interacting with other researchers who are at an early stage in their career. It is also an opportunity for students to develop skills presenting their work to a general audience of computer scientists working in a variety of different areas.

This volume contains the seven papers presented at the 2018 Imperial College Computing Student Workshop. It also includes abstracts of three of the posters presented during the workshop's poster session and an abstract of a fantastic invited talk given by Matthew Johnson from Microsoft Research.

In addition to the talks and poster sessions, ICCSW18 included a social event which took the students to visit Sky Garden on the Walkie-Talkie. Then, after a scenic walk taking us past London sights including the Tower of London and Tower Bridge, we invited the students to a meal at ASK Italian.

ICCSW18 has been a great success. We would like to thank everyone who participated in making this the case – the ICCSW committee, the student authors, our sponsors and the the department here at Imperial.

We wish the best of luck to the new committee. Until next year!

Edoardo Pirovano and Eva Graversen,
ICCSW18 Editors,
ACM Student Chapter 2017 – 2018

2018 Imperial College Computing Student Workshop (ICCSW 2018).
Editors: Edoardo Pirovano and Eva Graversen
OpenAccess Series in Informatics
OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Organisers

Organising Committee

In alphabetical order:

- Ahmad Alnafessah
- Pamela Bezerra
- Donato Clun
- Oana Cocarascu
- Alastair Donaldson
- Aydan Gasimova
- Eva Graversen
- Riccardo Moriconi
- Simon Olofsson
- Edoardo Pirovano
- Linh Tran
- Shale Xiong

Sponsors

2018 Imperial College Computing Student Workshop (ICCSW 2018).
Editors: Edoardo Pirovano and Eva Graversen
OpenAccess Series in Informatics
OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Speeding Up BigClam Implementation on SNAP

C. H. Bryan Liu[1]
Department of Computing, Imperial College London, United Kingdom
liu.ch.bryan@gmail.com

Benjamin Paul Chamberlain
Department of Computing, Imperial College London, United Kingdom
b.chamberlain14@imperial.ac.uk

Abstract

We perform a detailed analysis of the C++ implementation of the Cluster Affiliation Model for Big Networks (BigClam) on the Stanford Network Analysis Project (SNAP). BigClam is a popular graph mining algorithm that is capable of finding overlapping communities in networks containing millions of nodes. Our analysis shows a key stage of the algorithm – determining if a node belongs to a community – dominates the runtime of the implementation, yet the computation is not parallelized. We show that by parallelizing computations across multiple threads using OpenMP we can speed up the algorithm by 5.3 times when solving large networks for communities, while preserving the integrity of the program and the result.

2012 ACM Subject Classification Computing methodologies → Concurrent algorithms

Keywords and phrases BigClam, Community Detection, Parallelization, Networks

Digital Object Identifier 10.4230/OASIcs.ICCSW.2018.1

Category Main Track

Acknowledgements The authors thank Marc P. Deisenroth for useful discussions and the anonymous reviewers for providing many improvements to the original manuscript.

1 Introduction

Networks can represent many systems including social interactions, transport systems, financial transactions, communications infrastructure and biological functions. In all cases they describe interactions (edges) between dependent entities (nodes). One of the most important and best studied fields of network science is community detection [8, 13, 14]. A community can be thought as a group of nodes having a higher density of internal than external connections [4]. Early community detection algorithms partitioned small networks into disjoint regions, assigning each node to a single community [1, 17]. Later algorithmic advances both relax the disjointness requirement (allowing overlapping communities) and scale to much larger networks. Overlapping community detection algorithms are more general than partitioning methods, which they include as special cases [3, 18, 22]. Methods that focus on scaling community detection have allowed communities to be detected in networks with millions or even billions of nodes [2, 16, 24].

The Cluster Affiliation Model for Big Networks (BigClam), proposed by Yang and Leskovec [26] is both scalable and discovers overlapping communities. Under BigClam, nodes can be in multiple communities, and affiliation weight between a node and a community is modeled as a positive continuous number. The right half of Figure 1 shows the affiliation

[1] Now at ASOS.com, London, UK

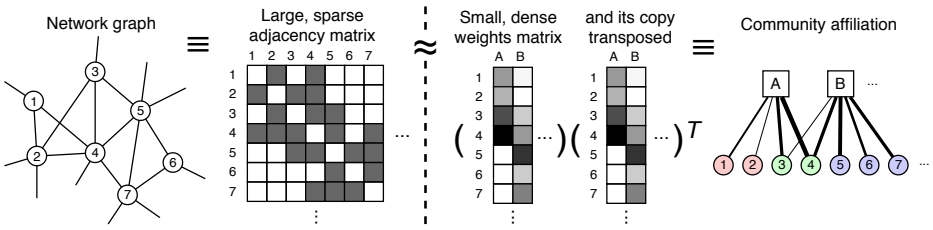


Figure 1 Illustration of community detection in a graph in terms of non-negative matrix factorization. (Left half) It is common to represent a network graph by a large, sparse adjacency matrix. (Right half) Yang and Leskovec proposed modeling community affiliations with a bipartite graph between communities and nodes, with affiliation weights represented by a small, dense, non-negative matrix [26]. (Middle two panels) By finding the most likely non-negative matrix, which when multiplied with its transpose best resembles the given adjacency matrix, we can obtain the most likely community affiliation w.r.t. the given network graph.

weights for seven nodes to two communities. This can be represented as a *bipartite graph*, or an *affiliation weights matrix*. The graph of a network is usually represented by a sparse binary adjacency matrix (left half). BigClam infers the affiliation weights matrix by applying non-negative matrix factorization [6] to the adjacency matrix. The algorithm learns the affiliation weights matrix that is best able to reconstruct the underlying adjacency matrix subject to the constraints of positivity and local optimality.

BigClam is a popular and highly cited method that features in a number of lectures and tutorials [10, 19]. The related software project [11] has attracted hundreds of GitHub stars. Due to the popularity of this model amongst both researchers and practitioners, we perform a rigorous analysis of the C++ implementation provided on the Stanford Network Analysis Project (SNAP) [11]. Our analysis of the BigClam source code reveals that the algorithm has three stages. In particular, the final Community Association (CA) stage, which makes assignments of nodes to communities, generally dominates the runtime, yet its computation is not parallelized across CPU threads. The runtime domination of CA is especially true for networks with large numbers of communities, which is common in real networks (see [9]). Not parallelizing computation where available results in lengthened runtime and wastes available hardware resources as they are put on idle.

This motivates our work in parallelizing computation in the CA stage to speed up the BigClam implementation on SNAP. Our major consideration is the parallelization must not introduce race condition on shared objects that compromise the integrity of the results. We parallelize the CA stage with OpenMP, a specification for high-level parallelism in C++ programs, and we show that the parallelization achieves as much as 5.3 times speed up and saves as much as 12.8 hours when solving networks by Leskovec and Krevl [9] using an eight-thread machine (Intel i7-4790 @ 3.60 GHz CPU).

To summarize, our contributions are as follow: (1) We profile the runtime of the BigClam implementation on SNAP in terms of its three stages. (2) We show that the CA stage dominates the runtime in current BigClam implementation on SNAP when solving networks with large numbers of communities, which is common in real networks. (3) We provide a detailed description, and the code implementation of how we parallelize computation on the CA stage, with a comprehensive discussion on avoiding race conditions. We also provide experimental results showing that the speed up is statistically significant, and preserves the result's integrity. [2]

[2] All code and experiment data are available on `https://github.com/liuchbryan/snap/tree/master/contrib/ICL-bigclam_speedup`.

Table 1 The average-case runtime complexity for the three stages of the BigClam community detection algorithm. $|V|$, $|E|$, $|C|$, r, k, t^* represents the number of nodes, edges, communities, community affiliations per node, epochs, and the speed-up multiple achieved by parallelizing computation across threads respectively. Derivations of the complexity are detailed in Appendix B.

Stage	Conductance Test	Gradient Ascent	Community Association
Complexity	$O\left(\lvert V\rvert \left(\frac{\lvert E\rvert}{\lvert V\rvert}\right)^2\right)$	$O\left(\lvert V\rvert \frac{kr}{t^*} \left(\frac{\lvert E\rvert}{\lvert V\rvert}\right)^2\right)$	$O(\lvert V\rvert\lvert C\rvert)$

Table 2 Number of nodes ($|V|$), communities ($|C|$), edges ($|E|$), and the average number of affiliations (r) recorded in the networks by Leskovec and Krevl [9].

	$\lvert V\rvert$	$\lvert C\rvert$	$\lvert E\rvert$	r
Amazon product co-purchase network	334,863	75,149	925,872	6.78
DBLP collaboration network	317,080	13,477	1,049,866	2.27
LiveJournal online social network	3,997,962	287,512	34,681,189	1.79
Youtube online social network	1,134,890	8,385	2,987,624	0.113

2 SNAP Implementation: The Bottleneck

We first examine the BigClam community detection algorithm and identify the bottleneck(s) in its implementation on SNAP. The core idea of BigClam is to find the affiliation weights matrix F that maximizes the log-likelihood function.[3] The mathematical formulation is detailed in Appendix A.

By examining its implementation on SNAP, we observe that the community detection algorithm has three stages: Conductance Test (CT), which initializes the affiliation strength matrix; Gradient Ascent (GA), which finds the optimal affiliation weights matrix; and Community Association (CA), which determines if an affiliation exists between a community and a node based on the value of affiliation weight recorded under the said matrix in relation to a pre-specified threshold.

We show the average-case runtime complexity of the three stages in Table 1. The full derivation is available in Appendix B. It can be seen that the CA stage will dominate the runtime if the number of communities is large, which we formalize as: $|C| \gg \frac{kr}{t^*}\left(\frac{|E|}{|V|}\right)^2$, where $|V|$, $|E|$, $|C|$, r, k, t^* represents the number of nodes, edges, communities, community affiliations per node, epochs, and the speed-up multiple achieved by parallelizing computation across threads respectively (see Appendix A.1).

Networks satisfying the inequality above are common. For example, all networks with ground-truth communities featured in Leskovec and Krevl [9] (shown in Table 2) satisfy the inequality when $k = 100$ and $t^* = 4$.[4] We confirm this by running the BigClam implementation on the networks shown in Table 2 using an eight-thread machine (Intel i7-4790 @ 3.60 GHz CPU), and measure the proportion of runtime spent in each of the three stages. Figure 2 shows the results of these experiments. While the time spent on the CT

[3] The $(u, c)^{\text{th}}$ entry of F represents the strength of the community affiliation between user u and community c in a network (see Figure 1 for an illustration).

[4] A conservative estimate of the speed up achieved by parallelizing the GA stage across eight threads.

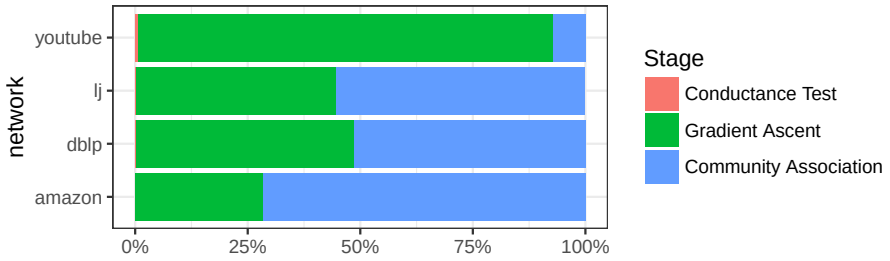


Figure 2 Average proportion of time spent on the three stages of the BigClam community detection algorithm (From left to right: Conductance Test (red), Gradient Ascent (green), and Community Association (blue)) in Leskovec and Sosič's implementation [11] for the four networks shown in Table 2. The implementation is tested on an eight-thread machine (Intel i7-4790 @ 3.60 GHz CPU), with the number of communities to detect for each network set to that recorded in Table 2.

stage is negligible, the time spent on the CA stage generally accounts for more than half of the entire runtime.[5]

Thus, the CA stage is usually the bottleneck in the algorithm. We notice that unlike the GA stage, in which computation is parallelized, the CA stage is not parallelized. Therefore, the majority of the CPU resources and man-time is wasted by idling. Parallelizing computation in the CA stage will better utilize available resources and hence improve scalability.

3 Speeding Up BigClam Via Parallel Computing

In this section we describe how to speed up BigClam with the use of OpenMP, a specification for parallel programming [15] that is supported in C++ and currently used in SNAP. The goal is to speed up the CA stage while ensuring that the input to output mapping is identical to the non-parallelized version.

3.1 Requirements in Result Correctness

As with any parallel computing application, it is important to prevent race conditions between threads from undermining the correctness of the result. In the context of speeding up the CA stage of BigClam, this means that the parallelized version must produce the same set of community affiliations as the unparallelized version given the same input F.

The theoretical representation of the communities returned by the algorithm is a set of sets: $M = \{M_1, M_2, ..., M_c\}$ where each community is itself a set $M_c = \{u_1, u_2, ...u_k\}$. Two runs of BigClam produce the same output if $M^{(1)} = M^{(2)}$. However, the BigClam SNAP implementation uses vectors of vectors (implemented as C++ STL-like objects) instead of sets of sets and enforces additional ordering that is not present theoretically. We denote the vector of vectors representation as $\mathcal{M}$. Using this representation it is possible to have $M^{(1)} = M^{(2)}$ and $\mathcal{M}^{(1)} \neq \mathcal{M}^{(2)}$ (see Figure 3).

[5] The Youtube network is an exception: we believe the average number of affiliations per node estimated by the BigClam algorithm is far greater than that recorded in the ground-truth (over 90% of the nodes do not have any community affiliations). This results in a far greater value of r than that reported in Table 2, which violates the inequality.

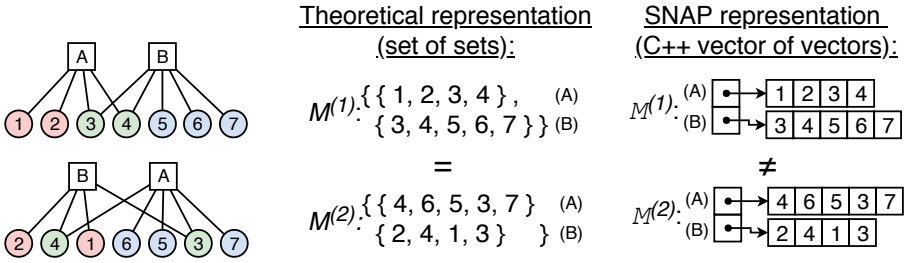


Figure 3 The two representations of the community affiliations. (Left) the underlying bipartite graph. (Middle) the theoretical representation as a set of sets. (Right) the SNAP C++ STL-like vector of vectors representation. In the theoretical representation the ordering is exchangeable, in the SNAP representation it is not.

As the ordering of values in the inner vectors and the single outer vector does not matter, we can allow race conditions between threads when performing append operations into these vectors, and thus increase the degree of parallelism in our implementation.

3.2 Methodology

Algorithm 1 outlines the existing CA stage implementation. The algorithm is simplified to include only operations related to scanning the matrix F and extracting community affiliations.[6] The algorithm is rewritten in pseudocode to enhance readability.

Algorithm 1 The existing implementation of the CA stage, simplified and rewritten in pseudocode. F is the affiliation weights matrix, and δ is the minimum affiliation strength threshold for a node to be considered a member of a community.

```
1: Initialize 𝓜 as an empty vector
2: for all c ∈ C do
3:     Initialize 𝓜_c as an empty vector
4:     for all u ∈ V do
5:         if F_uc ≥ δ then
6:             Append u to 𝓜_c
7:         end if
8:     end for
9:     Append 𝓜_c to 𝓜
10: end for
11: return 𝓜
```

As discussed in Section 3.1, we can spread the task of scanning a particular column of F over multiple threads while maintaining correctness – all node IDs will be added to the correct community vector, and with the proper synchronization mechanism (see discussion below) all community vectors will be present in the final result. In the terminology of OpenMP, we can parallelize the outer **for** loop over all communities, covering operations in lines 3–8 of Algorithm 1.

To prevent unintended race conditions while maintaining the highest level of parallelism, we declare a *critical operation* as any operation that involves objects that are shared between threads. Each critical operations is controlled by a mutex that prevents multiple threads from simultaneously writing to an object. It is safe to parallelize operations that involve only

[6] We exclude operations such as 1) sorting the vector $\left(\sum_u F_{uc}\right)_{c \in C}$ and scan columns which have a higher total affiliation strength first, and 2) excluding communities if it does not have enough members from being included, as they are less computationally expensive than scanning the matrix.

Table 3 Average time taken, in seconds, to run **a)** the community association (CA) stage **b)** the entire BigClam community detection algorithm, without and with parallelization of the community association stage. The implementations are tested on eight-thread machines with the same CPU specifications (Intel i7-4790 @ 3.60 GHz CPU).

Networks	CA stage		Overall	
	Unparallelized	Parallelized	Unparallelized	Parallelized
Amazon	1077.58	203.74	1505.38	610.23
DBLP	160.39	30.00	312.47	180.69
LiveJournal	55363.22	9233.02	100146.81	55259.48
Youtube	213.62	43.07	2965.12	2717.85

objects used by a single thread (a.k.a. private objects/variables) and read-only objects that are shared between threads. In our case, the only object that is shared between threads and involves write operations is the set of community affiliations $\mathcal{M}$. All other objects are either shared and read-only, or private to a thread.

- The affiliation weights matrix F is read-only by all threads
- The lower affiliation weight threshold δ is defined as a C++ constant (which is unmodifiable once defined), and hence is read-only
- The vector / list used to keep track of current community's members ($\mathcal{M}_c$) is local in the scope of the outer **for** loop, and hence is private to a thread according to the OpenMP specification [15].

Therefore, the only operation that needs to be declared as critical is the append to $\mathcal{M}$ in line 9 of Algorithm 1. Only one thread can append $\mathcal{M}_c$ to $\mathcal{M}$ at a time while all other operations can be parallelized.

4 Experiments

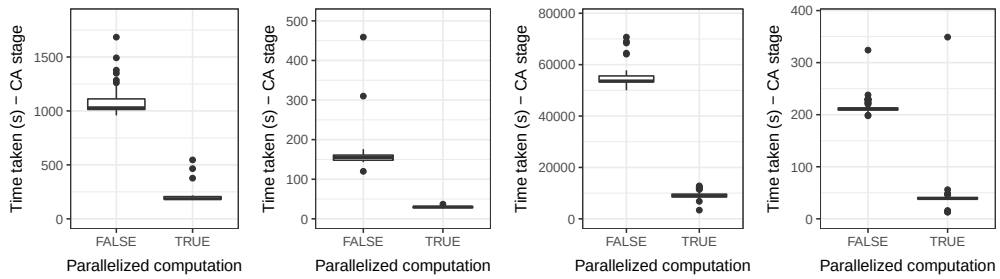
We run a number of experiments to validate the methodology described in Section 3. We show that parallelizing computation of the CA stage over multiple threads 1) reduces the runtime in the CA stage (and hence the overall BigClam implementation), and 2) retains the result correctness.

We use the datasets featured in Leskovec and Krevl [9] (see Table 2 for details of the datasets), which are widely used to benchmark the runtime of overlapping community detection algorithms [20, 22, 25, 27], including by BigClam itself [26].

4.1 Runtime Reduction

To demonstrate that parallelization reduces the algorithmic runtime, we run the unparallelized and parallelized variants of BigClam on multiple machines with Intel i7-4790 @ 3.60 GHz CPU (eight threads) for 100 epochs. Each machine runs only one of the variants at any time to ensure all CPU threads are dedicated to one variant. For each run, the program detects communities in the networks specified in Table 2, with the number of communities to detect set to that specified by the table. We measure the runtime of each stage of the BigClam for both the parallelized and unparallelized implementations across multiple runs.

The average runtime is reported in Table 3 and we perform a Welch's t-test to determine if the parallelized implementation achieves a significantly lower runtime for a) the CA stage, and b) the entire BigClam program. We visualize the results of this experiment in Figure 4.



(a) Amazon **(b)** DBLP **(c)** LiveJournal **(d)** YouTube

Figure 4 Box plot of the time taken, in seconds, to run the community association stage on different networks with ground-truth communities, without and with parallelization on the stage. The time taken with parallelization on the stage is significantly lower.

It is clear from Table 3 and Figure 4 that our implementation produces a significant runtime reduction in the CA stage for all networks shown in Table 2. With an eight-thread machine, we achieve a 5.3 times speed up in the CA stage, and subsequently a 2.5 times speed up in the overall BigClam algorithm for the Amazon product co-purchase network. For the LiveJournal network the runtime of the CA stage is reduced by 46,130 seconds (or 12.8 hours) on average.[7]

On the other hand, parallelizing the CA stage does not bring massive improvements in runtime on networks with low numbers of communities (those that do not satisfy the inequality in Section 2). We only achieve a 1.1 times speed up on the overall runtime solving the Youtube network, despite achieving a 4.96 times speed up on the CA stage. The speedup is not apparent in networks where the algorithm is dominated by the GA stage, where parallelizing the computation in the CA stage brings only marginal improvements.

4.2 Verification of Result Correctness

To confirm that parallelization of the CA stage produces the same set of community affiliation predictions as the unparallelized version we create a utility program. The program sorts the node IDs in a community and the communities in the program output in lexicographical order before comparing (strict) equality. This is necessary as common approaches to compare program outputs (e.g. `diff` or MD5 check sum) will fail even if two sets of communities are equal, as discussed in Section 3.1.

Our utility does not report any discrepancies between the program outputs produced by the parallelized and unparallelized variants, hence we conclude that our parallelization in the CA stage produces the same output, backed by a theoretical discussion in Section 3.2 and experimental verification.

5 Conclusion

In this work we profile the runtime of the BigClam implementation on SNAP, a popular overlapping community detection algorithm on an extensively used network analysis platform. We are able to split the runtime of the algorithm into three stages – the conductance test

[7] Yang and Leskovec state that, "with 20 threads, it takes about one day to fit BigClam to the LiveJournal network" [26] – we are able to fit this network with only eight threads in less than 16 hours.

(initialization) stage, the gradient ascent (optimization) stage and the community association (extraction) stage – and provide an average-case runtime complexity for each stage.

We show the community association stage is dominating the runtime in the current implementation when solving real networks, and parallelize its implementation to speed up BigClam. We show the speed up is both statistically significant and of practical utility, including a 5.3 times speed up on the community association stage (and 2.5 times overall) when solving the Amazon product co-purchase network, and saving 12.8 hours on the community association stage with an eight-thread machine. We release all relevant code and experimental data on our GitHub repository so that the research community can immediately benefit from our work and replicate our results.[8]

References

1 Reid Andersen, Fan Chung, and Kevin Lang. Local Graph Partitioning Using PageRank Vectors. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '06, pages 475–486, Washington, DC, USA, 2006. IEEE Computer Society. doi:10.1109/FOCS.2006.44.

2 Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008(10):P10008, 2008.

3 T. S. Evans and R. Lambiotte. Line graphs of weighted networks for overlapping communities. *The European Physical Journal B*, 77(2):265–272, September 2010. doi:10.1140/epjb/e2010-00261-8.

4 Santo Fortunato. Community Detection in Graphs. *Physics Reports*, 486(3):75–174, 2010. doi:10.1016/j.physrep.2009.11.002.

5 David F. Gleich and C. Seshadhri. Vertex Neighborhoods, Low Conductance Cuts, and Good Seeds for Local Community Methods. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '12, pages 597–605, New York, NY, USA, 2012. ACM. doi:10.1145/2339530.2339628.

6 Patrik O Hoyer. Non-negative matrix factorization with sparseness constraints. *Journal of Machine Learning Research*, 5(Nov):1457–1469, 2004.

7 Ravi Kannan, Santosh Vempala, and Adrian Vetta. On Clusterings: Good, Bad and Spectral. *J. ACM*, 51(3):497–515, May 2004. doi:10.1145/990308.990313.

8 Zhana Kuncheva and Giovanni Montana. Community Detection in Multiplex Networks Using Locally Adaptive Random Walks. In *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015*, ASONAM '15, pages 1308–1315, New York, NY, USA, 2015. ACM. doi:10.1145/2808797.2808852.

9 Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data, June 2014.

10 Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. *Mining of Massive Datasets*. Cambridge University Press, New York, NY, USA, 2nd edition, 2014.

11 Jure Leskovec and Rok Sosič. SNAP: A general-purpose network analysis and graph-mining library. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 8(1):1, 2016.

12 C. H. Bryan Liu. On overlapping community-based networks: generation, detection, and their applications. Master's thesis, Imperial College London, London, United Kingdom, June 2016.

[8] https://github.com/liuchbryan/snap/tree/master/contrib/ICL-bigclam_speedup

13 M. E. J. Newman. Detecting community structure in networks. *The European Physical Journal B*, 38(2):321–330, March 2004. doi:10.1140/epjb/e2004-00124-y.

14 M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Phys. Rev. E*, 69:026113, February 2004. doi:10.1103/PhysRevE.69.026113.

15 OpenMP Architecture Review Board. OpenMP application program interface version 4.5, 2015. URL: http://www.openmp.org/wp-content/uploads/openmp-4.5.pdf.

16 Pascal Pons and Matthieu Latapy. Computing Communities in Large Networks Using Random Walks. In *Proceedings of the 20th International Conference on Computer and Information Sciences*, ISCIS'05, pages 284–293, Berlin, Heidelberg, 2005. Springer-Verlag. doi:10.1007/11569596_31.

17 Alex Pothen. Graph Partitioning Algorithms with Applications to Scientific Computing. In David E. Keyes, Ahmed Sameh, and V. Venkatakrishnan, editors, *Parallel Numerical Algorithms*, pages 323–368, Dordrecht, 1997. Springer Netherlands. doi:10.1007/978-94-011-5412-3_12.

18 Ioannis Psorakis, Stephen Roberts, Mark Ebden, and Ben Sheldon. Overlapping community detection using Bayesian non-negative matrix factorization. *Phys. Rev. E*, 83:066114, June 2011. doi:10.1103/PhysRevE.83.066114.

19 Rik Sarkar. Community detection and cascades [Lecture]. http://www.inf.ed.ac.uk/teaching/courses/stn/files1516/slides/community-continued.pdf, 2015. Lecture slides from the course Social and Technological Networks (Autumn 2015), University of Edinburgh.

20 Meng Wang, Chaokun Wang, Jeffrey Xu Yu, and Jun Zhang. Community Detection in Social Networks: An In-depth Benchmarking Study with a Procedure-oriented Framework. *Proc. VLDB Endow.*, 8(10):998–1009, June 2015. doi:10.14778/2794367.2794370.

21 Duncan J Watts and Steven H Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442, 1998.

22 Joyce Jiyoung Whang, David F Gleich, and Inderjit S Dhillon. Overlapping community detection using seed set expansion. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, pages 2099–2108. ACM, 2013.

23 Yingjun Wu, Joy Arulraj, Jiexi Lin, Ran Xian, and Andrew Pavlo. An Empirical Evaluation of In-memory Multi-version Concurrency Control. *Proc. VLDB Endow.*, 10(7):781–792, March 2017. doi:10.14778/3067421.3067427.

24 Reynold S. Xin, Joseph E. Gonzalez, Michael J. Franklin, and Ion Stoica. GraphX: A resilient distributed graph system on spark. In *First International Workshop on Graph Data Management Experiences and Systems*, GRADES '13, pages 2:1–2:6, New York, NY, USA, 2013. ACM. doi:10.1145/2484425.2484427.

25 J. Yang and J. Leskovec. Community-Affiliation Graph Model for Overlapping Network Community Detection. In *2012 IEEE 12th International Conference on Data Mining*, pages 1170–1175, December 2012. doi:10.1109/ICDM.2012.139.

26 Jaewon Yang and Jure Leskovec. Overlapping Community Detection at Scale: A Non-negative Matrix Factorization Approach. In *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining*, WSDM '13, pages 587–596, New York, NY, USA, 2013. ACM. doi:10.1145/2433396.2433471.

27 Hongyi Zhang, Irwin King, and Michael R. Lyu. Incorporating Implicit Link Preference into Overlapping Community Detection. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, pages 396–402. AAAI Press, 2015. URL: http://dl.acm.org/citation.cfm?id=2887007.2887063.

A Key Formulation of BigClam Community Detection Algorithm

In this section we first introduce the nomenclature of networks, before moving on to the specifics of the BigClam community detection algorithm.

A.1 Network Preliminaries

A network is a data structure that contains a graph and a set of attributes. A graph $G(V, E)$ is composed of a set of nodes V and a set of edges $E = (v_i, v_j)$ where $v_i, v_j \in V$ that connect two nodes. The graph can be represented by an *adjacency matrix* $A \in \{0, 1\}^{|V|\times|V|}$ where A_{ij} is one if $(v_i, v_j) \in E$ and zero otherwise. Attributes can apply to either edges or nodes.

We denote the set of communities $C = \{c_1, c_2, ..., c_n\}$, where c_i indexes the i^{th} community. The set of community affiliations is defined as a set of sets $M = \{M_c : c \in C\}$, where $M_c = \{u_1, u_2, ..., u_k\}$ is a set containing the nodes affiliated to community $c \in C$.

We also denote $\mathcal{N}(u)$ as the set of neighbours of a node u in G, and the neighborhood containing u and its neighbors $N(u)$.

Part of our runtime analysis involves the average number of communities a node is affiliated with, which we formally define as:

▶ **Definition 1.** Let $D_u = \{c : u \in M_c\}$ be the set of communities that node $u \in V$ is affiliated with. Then the average number of community affiliations for all nodes r is

$$r = \frac{1}{|V|} \sum_{u\in V} |D_u| \,, \tag{1}$$

where $|D_u|$ is the number of communities that node u is affiliated with.

A.2 BigClam Community Detection Algorithm

The core idea of the BigClam community detection algorithm is to find the community affiliation weights matrix $F = (F_{uc})_{u\in V, c\in C}$, where the $(u, c)^{\text{th}}$ entry represents the strength of the community affiliation between user u and community c in a network (see Figure 1 for an illustration), that maximizes the log-likelihood function. Yang and Leskovec [26] use an iterative approach, where at each iteration they fix the affiliation weights for all but one node (say u), and perform a gradient ascent on the affiliation weights for node u. The log-likelihood for the corresponding row $\vec{F_u} = (F_{uc})_{c\in C}$ of F is specified as:

$$l(\vec{F_u}) = \sum_{v\in\mathcal{N}(u)} \log\left(1 - \exp(-\vec{F_u}\vec{F_v}^T)\right) - \sum_{v\notin\mathcal{N}(u)} \vec{F_u}\vec{F_v}^T \,. \tag{2}$$

We follow the original BigClam notation and so $\vec{F_u}\vec{F_v}^T$ is an *inner product*.

Differentiating Equation (2) w.r.t. $\vec{F_u}$ gives the gradient:

$$\nabla l(\vec{F_u}) = \sum_{v\in\mathcal{N}(u)} F_v \frac{\exp(-\vec{F_u}\vec{F_v}^T)}{1 - \exp(-\vec{F_u}\vec{F_v}^T)} - \sum_{v\notin\mathcal{N}(u)} \vec{F_v} \tag{3}$$

$$= \sum_{v\in\mathcal{N}(u)} F_v \frac{\exp(-\vec{F_u}\vec{F_v}^T)}{1 - \exp(-\vec{F_u}\vec{F_v}^T)} - \left(\sum_{v\in V} \vec{F_v} - \vec{F_u} - \sum_{v\in\mathcal{N}(u)} \vec{F_v}\right) \,. \tag{4}$$

In Equation (4) $\sum_{v\in V} \vec{F_v}$ can be precomputed, and $\sum_{v\in\mathcal{N}(u)} \vec{F_v}$ is computed on each gradient evaluation. This results in a more computationally efficient formulation as network graphs are usually sparse (i.e. $|\mathcal{N}(u)| \ll |V|$).

The BigClam community detection algorithm initializes F as:

$$F_{(u')(N(u))} = \begin{cases} 1 & \text{if } u' \in N(u) \text{ and } N(u) \text{ is a locally} \\ & \text{minimal neighborhood [5] of } u \\ 0 & \text{otherwise} \end{cases}, \tag{5}$$

where $N(u)$ represents u and its neighbours in G, and regards $u \in V$ as a member of $c \in C$ from the most likely affiliation weights matrix F if:

$$F_{uc} \geq \delta = \sqrt{-\log(1-\epsilon)}\,, \tag{6}$$

where $\epsilon = \frac{2|E|}{|V|(|V|-1)}$ is the background probability for a random edge to form in the graph.

B SNAP Implementation: A Runtime Complexity Analysis

We observe the BigClam community detection algorithm has three stages: Conductance Test, Gradient Ascent, and Community Association. Here we derive the runtime complexity for each of the three stages.

B.1 The Conductance Test Stage

The algorithm begins by testing each node to see if it belongs to a locally minimal neighborhood as defined by Gleich et al. [5]. The initial / seed communities are chosen to be the locally minimal neighborhoods.

For each node $u \in V$ we calculate the conductance of its neighborhood. The conductance of an neighbourhood $N(u)$ is the fraction of edges from nodes within $N(u)$ to nodes in the same neighborhood over that to nodes outside the neighborhood [7]. This involves traversing each neighbor $v \in \mathcal{N}(u)$ and finding out how many members of $\mathcal{N}(v)$ are not in $N(u)$. Hence there are $\sum_{u \in V} \sum_{v \in \mathcal{N}(u)} |\mathcal{N}(v)|$ operations involved.

We simplify the expression above by replacing $|\mathcal{N}(u)|\ \forall u \in V$ with the average number of neighbors, and using the fact that it is by definition the average degree of the network graph ($|E|/|V|$). This leads to an average-case complexity of $O\left(|V|\frac{|E|}{|V|}\frac{|E|}{|V|}\right)$.

B.2 Gradient Ascent Stage

After initialization, the algorithm optimizes the affiliation weights matrix F to maximize the log-likelihood function (see Equation (2)) using gradient ascent. To understand the runtime complexity of the GA stage, we first look at the two building blocks – calculating the dot product and summing $\vec{F_v}$ – and their runtime complexity in the implementation.

B.2.1 Dot Product Runtime Complexity

Calculating the dot product between two vectors $\vec{F_u}$ and $\vec{F_v}$ is required to calculate the gradient given in Equation (3). This is performed on each pairs of connected nodes in G for each epoch.

In a naïve implementation that sums the product of the corresponding (dense) vector elements, the number of operations required scales with the length of $\vec{F_u}$. Many such operations in this context are unnecessary – a node u in real networks is likely to be affiliated

with only a small number of communities,[9] leading to a large number of entries in $\vec{F_u}$ being set to zero (as u is unaffiliated to those communities).

The SNAP implementation stores $\vec{F_u}$ as a sparse vector, where only non-zero elements are recorded along with its position. Using sparse vectors, the number of operations for a dot product between F_u and F_v scales as:

$$d_{\mathrm{DP}}(u,v) \triangleq \min\left(|D_u|, |D_v|\right), \tag{7}$$

which is the minimum number of community affiliations possessed by the two nodes u and v.

B.2.2 Vector Sum Runtime Complexity

We then consider the number of elements to be traversed in each $\vec{F_v}$ when calculating the sum of affiliation weights for all neighbors of a node u, $\sum_{v\in\mathcal{N}(u)} \vec{F_v}$. The sum is featured in Equation (4) as part of the gradient calculation. Similar to the dot product calculation, implementing $\vec{F_v}$ as sparse vectors means the algorithm need not consider all $|C|$ affiliation weights in $\vec{F_v}$ but only the weights associated with communities that the neighbor nodes are a member of: $\bigcup_{v\in\mathcal{N}(u)} D_v \subseteq C$.

The cardinality of the set in the expression above is bounded above by

$$d_{\mathrm{VS}}(u) \triangleq |\mathcal{N}(u)| \max_{v\in\mathcal{N}(u)} (|D_v|), \tag{8}$$

assuming all neighbors of node u belong to disjoint sets of communities.[10]

B.2.3 Overall Runtime Complexity of the GA Stage

We can now estimate the runtime complexity of the GA stage. In this stage the algorithm iterates over the nodes multiple times, calculates the row gradient in Equation (3) and updates the affiliation weights. This is done until the convergence criteria is met, or for a pre-specified number of epochs.

Equation (3) shows that the gradient of the log-likelihood is the difference of two summations. The first summation involves calculating the vector sum and dot product over each neighbor $v \in \mathcal{N}(u)$, and the second summation involves calculating the vector sum over $v \notin \mathcal{N}(u)$. This is made more efficient by Equation (4) as real graphs are sparse and so the number of neighbors of a node is far less than the number of non-neighbors. The number of operations required in calculating the row gradient is then bounded above by:

$$\gamma \left[\sum_{v\in\mathcal{N}(u)} \left(d_{\mathrm{VS}}(u) + d_{\mathrm{DP}}(u,v)\right) + \sum_{v\in\mathcal{N}(u)} d_{\mathrm{VS}}(u) \right], \tag{9}$$

where γ is a constant multiplier.

We notice that $d_{\mathrm{VS}}(u)$ dominates $d_{\mathrm{DP}}(u,v)\ \forall v \in \mathcal{N}(u)$, and hence Expression (9) can be further simplified to $\gamma' \left[|\mathcal{N}(u)|\, d_{\mathrm{VS}}(u)\right]$, where γ' is another constant multiplier.

---

[9] Liu [12] has shown the the maximum number of affiliations for any node is 116 out of 75,149 possible communities in the Amazon product co-purchase network, and 682 out of 957,154 possible communities in the LiveJournal social network.

[10] In practice the cardinality will be much smaller due to the “small world" phenomenon: a node's neighbors are likely to be connected themselves [21], which according to BigClam is due to them being mutual members of one or more communities.

We replace $|\mathcal{N}(u)|$ by $|E|/|V|$, just as we did in Section B.1. Furthermore we approximate $d_{\text{VS}}(u)$ (see Equation (8)) in the average case by $|E|/|V| \times r$.

We have to calculate the row gradient for all $|V|$ nodes over k epochs (which we specify). Moreover, the computation of the GA stage is parallelized onto t threads using OpenMP [15], which will reduce the runtime by $t^* \leq t$ folds due to synchronization overhead [23]. We arrive at our average-case complexity of

$$O\left(\frac{k}{t^*}|V|\frac{|E|}{|V|}\frac{|E|}{|V|}r\right). \tag{10}$$

Note that r in Expression (10) is the BigClam estimate of the average number of affiliations per node, not the value realized in the ground-truth, and the two values can differ significantly.

B.3 Community Association Stage

The final stage of the algorithm takes the most likely community affiliation weights matrix F, and for each community $c \in C$ and each node $u \in V$ determines if u is affiliated to c by examining the entry F_{uc} (see Equation (6)).

The implementation treats rows of F as dense vectors, and requires scanning through all entries of F to determine all community affiliations. Hence, at least $|C||V|$ comparisons must be performed, leading to an average-case complexity of $O(|C||V|)$.

THRIFTY: Towards High Reduction In Flow Table memorY

Ali Malik
School of Computing, University of Portsmouth, United Kingdom
ali.al-bdairi@port.ac.uk

Benjamin Aziz
School of Computing, University of Portsmouth, United Kingdom
benjamin.aziz@port.ac.uk

Chih-Heng Ke
Department of Computer Science and Information Engineering, National Quemoy University, Taiwan
smallko@nqu.edu.tw

Abstract

The rapid evolution of information technology has compelled the ubiquitous systems and computing to adapt with this expeditious development. Because of its rigidity, computer networks failed to meet that evolvement for decades, however, the recently emerged paradigm of software-defined networks gives a glimpse of hope for a new networking architecture that provides more flexibility and adaptability. Fault tolerance is considered one of the key concerns with respect to the software-defined networks dependability. In this paper, we propose a new architecture, named THRIFTY, to ease the recovery process when failure occurs and save the storage space of forwarding elements, which is therefore aims to enhance the fault tolerance of software-defined networks. Unlike the prevailing concept of fault management, THRIFTY uses the Edge-Core technique to forward the incoming packets. THRIFTY is tailored to fit the only centrally controlled systems such as the new architecture of software-defined networks that interestingly maintain a global view of the entire network. The architecture of THRIFTY is illustrated and experimental study is reported showing the performance of the proposed method. Further directions are suggested in the context of scalability towards achieving further advances in this research area.

2012 ACM Subject Classification Networks → Network protocols

Keywords and phrases Source Routing, Resiliency, Fault Tolerance, SDN, TCAM

Digital Object Identifier 10.4230/OASIcs.ICCSW.2018.2

Category Main Track

1 Introduction

Computer networks play an essential role in changing the life style of modern society. Nowadays, most of the Internet services are located in data centers, which are consisting of thousands of computers that connected via large-scale data center networks. Typically, wide-area networks interconnecting the data centers that distributed across the globe. The Internet users are usually using their devices (i.e. computer, mobile, tablet, smart watch, etc.) to access the various available services of Internet through different ways such as WiFi, Ethernet and cellular networks. Traditionally, the distributed control systems in networking devices along with a set of defined protocols (e.g. OSPF [16] and RIP [9]) constitute a fundamental technology that have been adopted to send and receive data via networks

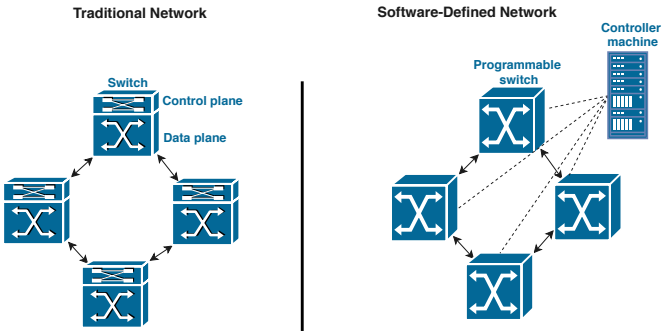


Figure 1 Traditional versus SDN architecture.

around the world in recent years. According to [2], these distributed protocols increase the inflexibility of network management through making the network operators to lose their visibility over their networks. Managing the networks efficiently to meet the requirements of the Quality of Service (QoS) and the Service Level Agreements (SLA) are the core challenging points of the computer networks, which need to be improved continuously in light of the increasing number of devices that are connected to the Internet, which are currently estimated to be in the range of 9 billion devices and expected to reach double that number by 2020. Therefore, the Internet ossification is highly expected as stated in [12]. One possible solution is to replace the complex/rigid networking system with an open and programmable network instead. Software-Defined Networking (SDN) is a promising paradigm that resulted from a long history of efforts aiming to simplify the computer networks management and control [5]. In SDN the *control plane* has been decoupled from the *data plane* and placed in a central location usually called the network *controller* or the network operating system. Figure 1 illustrates the difference between the SDNs and conventional networks architecture. Such a new networking architecture of SDN with much more flexibility comparing to the traditional networks meant that SDNs are nowadays adopted by many of the well known pioneering companies like Deutsche Telekom, Google, Microsoft, Verizon, and CISCO, which have recently combined in 2011 to launch the Open Network Foundation (ONF) [18] as a nonprofit consortium that aims to accelerate the adoption of SDN technologies.

Although SDNs have brought many advantages with dramatic network improvements, this innovation has been accompanied by several challenges, such as the management of network failures and updating of the network architecture [1].

2 Related Work

Since link and node failure is an issue almost as old as computer networks, so far, SDN follows the traditional fundamental strategies of failure recovery (i.e. protection and restoration) to recover from the data plane failures. However, the fault management in SDNs differs from the legacy networks in the way of computing and update the routing tables. Instead of the conventional way of reconfiguration in which each node makes the required changes to update the routing table locally, the controller in SDN is responsible to handle the network reconfiguration and instruct the relevant nodes on how to follow the new update, which is therefore made globally. Protection and restoration are currently the only two ways to reconfigure the network and mask failure incidents. However, each associated with some drawbacks in terms of time and memory space consumption. In protection, the alternative solutions (i.e. backups) are preplanned and installed in the relevant switches, however, in

restoration the possible solutions are not preplanned and will be calculated dynamically when failure occurs. A large number of studies have considered the issue of network failures and propose different contributions that are reviewed in [6]. Unfortunately, the current SDN switches in the market have a limited capacity of flow table due to the small space of the expensive Ternary Content-Addressable Memory (TCAM) [10]. Recently, this issue took place in the proposed schemes of failure recovery as the new schemes should consider the problem of TCAM limitation.

In this context, [19] propose SlickFlow, a source-based routing method to enhance the scalability and fault tolerance in OpenFlow networks. In SlickFlow, the controller computes the primary and the backup (disjoint) paths and then both are encoded in the packet header in addition to an *alternative* bit, which indicates the current using path. When the primary path is affected by link failure then, a switch will forward the packets through the backup path and change the value of the alternative, which is necessary for the neighbor switch to follow the backup as well. The packet header provides an additional limited segment of information that can be used for the purpose of encoding path details [17], where the alternative path should not exceed 16 hops.

The authors in [15] produce a protection scheme, as an extension to their previous work in [14], that minimises TCAM consumption. The authors developed two routing strategies: *Backward Local Rerouting* (BLR) and *Forward Local Rerouting* (FLR). In BLR, a node-disjoint path is computed as a backup for every primary path in the network and when a failure occurs, packets are sent back to the origin node to be rerouted over the pre-planned backup towards the destination. In FLR, a backup route for each link in the primary path is pre-computed. When a link failure occurs, the packets will be forwarded from the point of failure towards the downstream switch in the primary path by following the backup path, however, in case of there will be a multiple backups then, the one with least number of switches will be chosen. Instead of using fast failover group type, the authors have extended the OpenFlow protocol by adding an additional entry that called `BACKUP_OUTPUT` to the `ACTION SET` of the flow table entries, so that the new added entry is responsible to set the out put port when a link fails.

The authors in [23] propose a new flow tables compression algorithm as well as a compression-aware routing concept to enhance the ratio of the gained compression rate. The proposed algorithm reduces the consumed TCAM space by using the wildcard to match the tables who shared the same output and packet modification operations and hence the compression. The authors relied on their previous work [22] in which they proposed Plinko as a new forwarding model where the forwarding table entries apply the same action.

The authors in [24] discuss the problem of the protection schemes and its impact on the shortage of TCAM memory. The authors proposed *Flow Entry Sharing Protection* (FESP), which is a greedy algorithm that selects the node with larger available flow entry capacity and minimum backup flow entry. The study showed how the total number of flow entries can be minimised where the experimental results revealed that the reduction ratio of flow entries is up to 28.31% compared with the existing path and segment protection methods. With respect to all contributions, some issues still exist such as the following:

1) The disjointness as constraint for the calculated backups will require a totally new set of flow entries, which in turn will consume an additional TCAM space.

2) Compress flow tables using wildcard will affect the fine-grained per packet inspection and therefore might lead to policy/security violations.

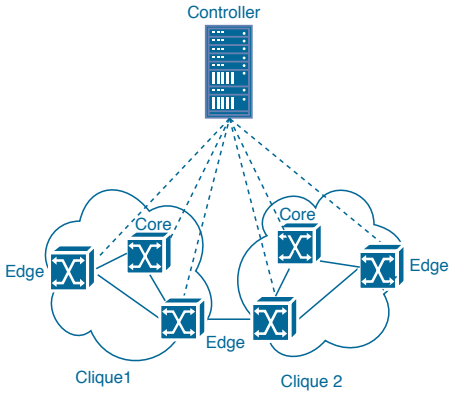


Figure 2 THRIFTY architecture.

3 Problem Statement

On one hand, protection solutions require an additional information, which have to be loaded into the data plane elements, to tell the nodes how to perform when failure occurs. However, the extra loaded information affects the storage memory of the network switches and therefore the designed fault tolerance mechanisms should consider the limited space of flow table and TCAM. On the other hand, it is very hard to meet the carrier-grade reliability requirements (i.e. recover within 50 μs) in restoration [20, 21] because the infrastructure layer equipment in SDN are dummy forwarding elements due to the split architecture, then, the central controller is responsible for calculating the alternative paths and then installing the flow entries (i.e. forwarding rules) in the relevant switches of each backup after detecting failures.

4 THRIFTY for SDNs

THRIFTY is a scalable fault tolerant system with aim to reduce the TCAM storage space of forwarding elements as much as possible. THRIFTY has the following properties:

- *Edge-Core based routing*: The idea of Edge-core design has been proposed in [4], in which the complex control functions have been removed to the ingress switches and keep the remain core switches as clean-slate. THRIFTY makes use the same idea of Edge-core design and to be applied on the partitioned network topology, as an extension to our previous work in [13] in which the network topology can be divided into N number of cliques.
- *Fast recovery*: Reacting to network link failures, THRIFTY is capable to recover from single/multi link failures in a carrier-grade time scale (i.e. less than 50 μs).
- *Scalable to large-scale networks*: As the size of network topology increases, the flow table entries of data plane will be still manageable due to the designed architecture.
- *Single network controller*: Network can be controlled by one controller and it is the entity that responsible for the network activities and adjust the global policy of network.

4.1 Architecture

Figure 2 depicts THRIFTY architecture, the controller comprises three modules, each responsible for a specific task as follows:

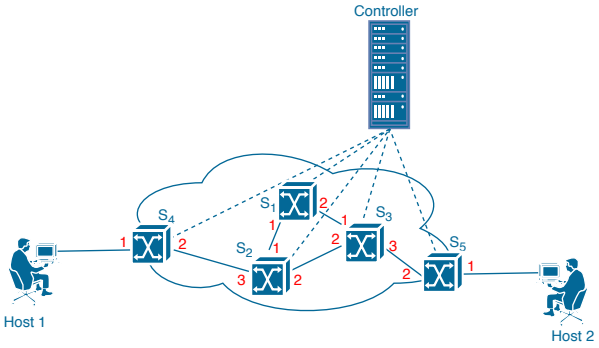


Figure 3 Example topology.

1) *Topology parser*: is responsible for fetching the underlying network topology characteristics and build a topological view in order to represent the gained network topology as a graph G, we utilised the NetworkX [8] tool, which is a pure python package with a powerful set of functions that can be used to manipulate and simplify network graphs.
2) *Cliques producer*: is responsible for partitioning the constructed network graph G into set of sub-graphs by incorporating the well known community detection algorithm *Girvan and Newman* [7] to produce a set of possible cliques (with any size). The densely connection between the resulted cliques' vertices is the main interesting feature of Girvan and Newman algorithm, in other words, the strong connection among the nodes in each clique could provide a multiple alternative paths that could be utilised when failures occur.
3) *Edge-Core finder*: Based on the resulted cliques, this module is responsible for dividing the set of nodes, in each single clique, into two sets *Edge* and *Core*. Therefore, we will have two kind of switches, namely Edge and Core. The key challenging point of this module is to find the optimal number of Edge switches.

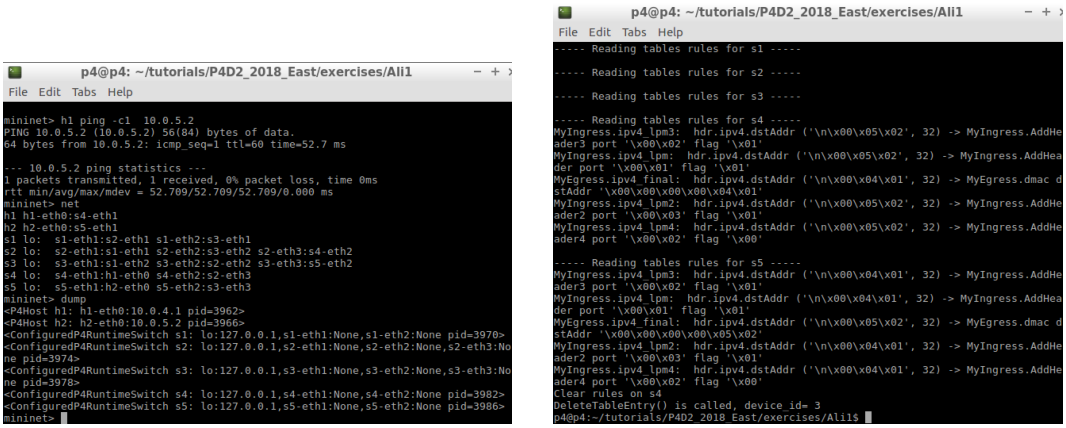
4.2 Prototype and Implementation

To demonstrate the feasibility of the proposed architecture, we provide a prototype implementation of THRIFTY. The current prototype is designed as a proof of concept as well as to show how the proposed solution can be applied.

The current implementation of THRIFTY is prototyped with the recently proposed P4 language [3] using a software switch as a platform. We use the open source P4[1] as a packet processing language to create a set of P4 switches in the specified topology of Figure 3 in which *Edge*={`S4, S5`} and *Core*={`S1, S2, S3`}. We evaluate our THRIFTY prototype using Mininet [11] as a virtual network emulator, which is suitable to generate customized virtual network topologies in a single Linux machine. The current implementation is divided into two schemes as follows:

1. Rules aggregation method ($Scheme_1$)[2]
   In this method, the necessary flow entries (from source to destination) of a particular path are stored in Edge switches of the network in addition to add one more flow entry

[1] `P4 switch model available at: https://github.com/p4lang`
[2] `The implementation can be found at: http://csie.nqu.edu.tw/smallko/sdn/mysource_routing.htm`



(a) Ping test. **(b)** Required rules.

Figure 4 Adding rules with Scheme$_1$.

in each Edge switch for the purpose of changing the destination mac address. Therefore, the number of required flow entries in this method can be calculated by:

`The number of traversed switches in a path + 1`

Figure 4 shows the preliminary results of this method. Although the scheme fails to reduce the number of required rules, it is still of interest since it collects the required rules in two locations (i.e. Edges) rather than distributed them over switches and therefore it might increase the flexibility of updating the network.

2. Rules compression method ($Scheme_2$)[3]
   In this method, the entire flow entries from source to destination of a particular path are reduced to one rule only, which also need to be stored in one of the path Edge switches. For instance, in the given example topology the shortest path between `Host1` and `Host2` is:

   `Host1-S4-S2-S3-S5-Host2`

   we set the next couple of rules to indicate the routing information at the ingress switch (`S4`):

   ```
   table_add ipv4_lpm set_path 10.0.5.2/32 => 4 1 3 2 2 0 0 0 0 0
     table_add ipv4_final dmac 10.0.4.1/32 => 00:00:00:00:04:01
   ```

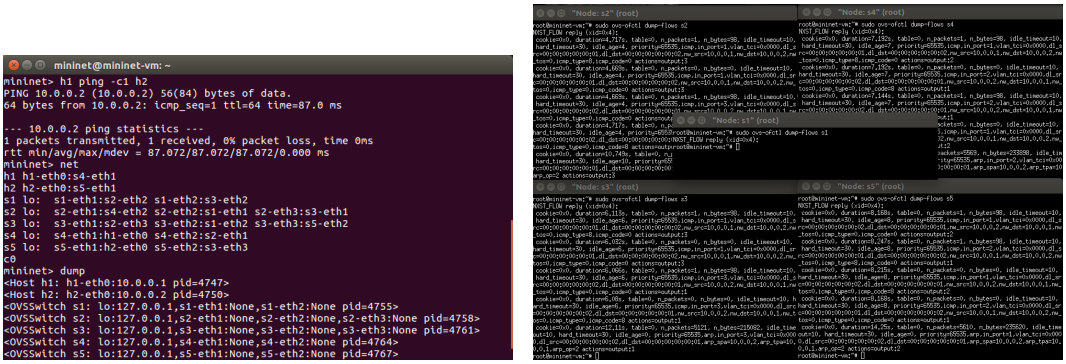
   While, in the egress switch (`S5`) we had the following;

   ```
   table_add ipv4_lpm set_path 10.0.4.1/32 => 4 1 3 2 2 0 0 0 0 0
     table_add ipv4_final dmac 10.0.5.2/32 => 00:00:00:00:05:02
   ```

   Where 4 indicates that the shortest path between `Host1` and `Host2` contains 4 nodes (i.e. 3 hops). While, the rest of digits denotes the set of output ports for the switches along the path as follow: 1 refers to the output port of `S5`, 3 refers to the output port of `S3`, 2 refers to the output port of `S2` and the last 2 refers to the output port of `S4`.

In order to compare our proposed methods to traditional/existing method, we replicated the same experimental procedure with POX controller where the *openflow.discovery*[4] and

[3] `The implementation can be found at:`
`http://csie.nqu.edu.tw/smallko/sdn/mysource_routing2.htm`
[4] `https://github.com/att/pox/blob/master/pox/openflow/discovery.py`



(a) Ping test. **(b)** Required rules.

Figure 5 Adding rules with traditional scheme.

l2_multi[5] modules have been utilised to discover and setup the shortest path from sender to receiver. Figure 5 shows the number of rules required to forward the incoming packets from `Host1` to `Host2`. It is worth mentioning here that we took into account the only IP packets, however, the arp packets have been discarded.

As a result, Figure 6 illustrates the total number of flow rules required by the three simulated schemes (i.e. $Scheme_1$, $Scheme_2$ and traditional).

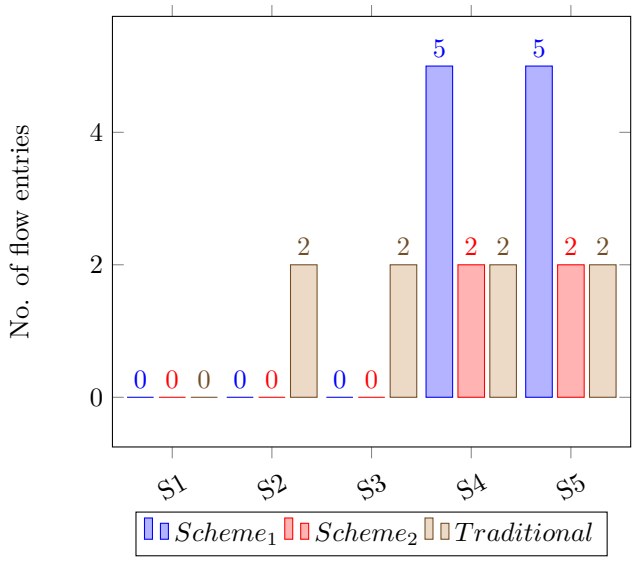


Figure 6 A comparison of three scenarios.

5 Conclusion and Future Work

In this paper, we presented the ongoing work in realising THRIFTY as a new solution to tackle the TCAM limitation problem as well as to accelerate the recovery from link failures. We showed how the proposed solution can be implemented using a couple of new schemes

[5] `https://github.com/att/pox/blob/master/pox/forwarding/l2_multi.py`

that aggregate and compress the forwarding rules. As a future plan, the authors will proceed to conduct failure scenarios in addition to extend the current piece of work by considering the following aspects:

Building a general framework: THRIFTY uses Edge-Core architecture on the basis of cliques concept with a view of dramatically simplifying the packet forwarding as well as reducing the number of flow table entries that will enhance the SDN scalability.

Dependability attributes: Currently, THRIFTY only supports the scenario of data plane link failures, however, our work envisions to include other attributes of dependability such as security.

References

1 Ian F Akyildiz, Ahyoung Lee, Pu Wang, Min Luo, and Wu Chou. A roadmap for traffic engineering in SDN-OpenFlow networks. *Computer Networks*, 71:1–30, 2014.

2 Theophilus Benson, Aditya Akella, and David A Maltz. Unraveling the Complexity of Network Management. In *NSDI*, pages 335–348, 2009.

3 Pat Bosshart, Dan Daly, Glen Gibb, Martin Izzard, Nick McKeown, Jennifer Rexford, Cole Schlesinger, Dan Talayco, Amin Vahdat, George Varghese, et al. P4: Programming protocol-independent packet processors. *ACM SIGCOMM Computer Communication Review*, 44(3):87–95, 2014.

4 Martin Casado, Teemu Koponen, Scott Shenker, and Amin Tootoonchian. Fabric: a retrospective on evolving SDN. In *Proceedings of the first workshop on Hot topics in software defined networks*, pages 85–90. ACM, 2012.

5 Nick Feamster, Jennifer Rexford, and Ellen Zegura. The road to SDN: an intellectual history of programmable networks. *ACM SIGCOMM Computer Communication Review*, 44(2):87–98, 2014.

6 Paulo Fonseca and Edjard Mota. A survey on fault management in software-defined networks. *IEEE Communications Surveys & Tutorials*, 2017.

7 Michelle Girvan and Mark EJ Newman. Community structure in social and biological networks. *Proceedings of the national academy of sciences*, 99(12):7821–7826, 2002.

8 Aric Hagberg, Pieter Swart, and Daniel S Chult. Exploring network structure, dynamics, and function using NetworkX. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008.

9 Charles L Hedrick. Routing information protocol. Technical report, Rutgers University, 1988. https://tools.ietf.org/html/rfc1058.

10 Diego Kreutz, Fernando MV Ramos, Paulo Esteves Verissimo, Christian Esteve Rothenberg, Siamak Azodolmolky, and Steve Uhlig. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76, 2015.

11 Bob Lantz, Brandon Heller, and Nick McKeown. A network in a laptop: rapid prototyping for software-defined networks. In *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, page 19. ACM, 2010.

12 Pingping Lin, Jun Bi, Hongyu Hu, Tao Feng, and Xiaoke Jiang. A quick survey on selected approaches for preparing programmable networks. In *Proceedings of the 7th Asian Internet Engineering Conference*, pages 160–163. ACM, 2011.

13 Ali Malik, Benjamin Aziz, Chih-Heng Ke, Han Liu, and Mo Adda. Virtual topology partitioning towards an efficient failure recovery of software defined networks. In *The 16th International Conference on Machine Learning and Cybernetics (ICMLC)*. IEEE, 2017.

14 Purnima Murali Mohan, Tram Truong-Huu, and Mohan Gurusamy. TCAM-aware local rerouting for fast and efficient failure recovery in software defined networks. In *Global Communications Conference (GLOBECOM), 2015 IEEE*, pages 1–6. IEEE, 2015.

15 Purnima Murali Mohan, Tram Truong-Huu, and Mohan Gurusamy. Fault tolerance in TCAM-limited software defined networks. *Computer Networks*, 116:47–62, 2017.

16 John Moy. OSPF version 2. Technical report, Ascend Communications, Inc., 1998. https://tools.ietf.org/html/rfc2328.

17 Giang TK Nguyen, Rachit Agarwal, Junda Liu, Matthew Caesar, P Godfrey, and Scott Shenker. Slick packets. *ACM SIGMETRICS Performance Evaluation Review*, 39(1):205–216, 2011.

18 ONF. Open Networking Foundation, 2018. https://www.opennetworking.org/.

19 Ramon Marques Ramos, Magnos Martinello, and Christian Esteve Rothenberg. Slickflow: Resilient source routing in data center networks unlocked by openflow. In *Local Computer Networks (LCN), 2013 IEEE 38th Conference on*, pages 606–613. IEEE, 2013.

20 Sachin Sharma, Dimitri Staessens, Didier Colle, Mario Pickavet, and Piet Demeester. Enabling fast failure recovery in OpenFlow networks. In *Design of Reliable Communication Networks (DRCN), 2011 8th International Workshop on the*, pages 164–171. IEEE, 2011.

21 Dimitri Staessens, Sachin Sharma, Didier Colle, Mario Pickavet, and Piet Demeester. Software defined networking: Meeting carrier grade requirements. In *Local & Metropolitan Area Networks (LANMAN), 2011 18th IEEE Workshop on*, pages 1–6. IEEE, 2011.

22 Brent Stephens, Alan L Cox, and Scott Rixner. Plinko: Building provably resilient forwarding tables. In *Proceedings of the Twelfth ACM Workshop on Hot Topics in Networks*, page 26. ACM, 2013.

23 Brent Stephens, Alan L Cox, and Scott Rixner. Scalable multi-failure fast failover via forwarding table compression. In *Proceedings of the Symposium on SDN Research*, page 9. ACM, 2016.

24 Xiaoning Zhang, Shui Yu, Zhichao Xu, Yichao Li, Zijing Cheng, and Wanlei Zhou. Flow Entry Sharing in Protection Design for Software Defined Networks. In *GLOBECOM 2017-2017 IEEE Global Communications Conference*, pages 1–7. IEEE, 2017.

Data-Driven Chinese Walls

Gulsum Akkuzu
School of Computing, University of Portsmouth, United Kingdom
gulsum.akkuzu@port.ac.uk

Benjamin Aziz
School of Computing, University of Portsmouth, United Kingdom
benjamin.aziz@port.ac.uk

Abstract

Security policy and access control models are often based on qualitative attributes, e.g. security labels, cryptographic credentials. In this paper, we enrich one such model, namely the Chinese Walls model, with quantitative attributes derived from data. Therefore, we advocate a data-driven approach that considers a quantitative definition of access we term, working relations.

2012 ACM Subject Classification Security and privacy → Access control

Keywords and phrases Access Control, Big Data, Security Policies, Chinese Walls Model

Digital Object Identifier 10.4230/OASIcs.ICCSW.2018.3

Category Main Track

1 Introduction

Organisations require controlling and monitoring the access of their information, shared files and resources in order to ensure the security of information and assets. Access control provides access rights to authorized users. Unauthorized users are refused to access to information and assets. There are various access control models in the literature, including mandatory access control, discretionary access control, role-based access control, lattice-based information flow control and Chinese Wall Access Control (CWAC) [4, 10].

CWAC was introduced by Brewer and Nash [2] with the aim preventing information flows that cause Conflicts of Interest (CoI) for consultants. It is considered in the commercial domain in which consultants and analysts of organisations access sets of data resources from different groups in companies that provide different types of services. CWAC prevents data leaks from one company to another within the same CoI class.

CWAC, like many other models, lacks quantitative attributes and analysis that could render the model more usable within data-driven domains. We therefore argue that such a model should be enhanced with definitions that take into consideration statistical information derived from available datasets. We introduce in this paper one such initial enhancement by taking into account working relations of users towards computers they access. A working relation represents persistent accesses that exceed some minimum number of times. We define a new version of the simple secrecy property based on working relations. To the best of our knowledge, this research is the first for incorporating quantitative aspects such as probabilistic or stochastic variations of CWAC model.

2 Related Work

Our work is closely related to the basic concept of CWAC. We provide an overview of the most relevant works to the this paper in the followings. CWAC is a well known access control for the secured sharing of commercial consultancy services. A concept analysis is introduced which shows effects of lattice structure to classify the access permission of consultants based on the CWAC. And also helps to understand the CWAC access permission of every consultant depends on their level in the lattice structure [11].

A new way of thinking on Chinese Wall Model Security has been brought by Cheng and et all [3], they address risk access control decisions on the CWAC and other security policies where access decisions are based on the history. The new concept has been illustrated as a Fuzzy Logic control system because of the uncertainty in security labels, risk modulating factors and loss variance based access decision have effective roles on the access decisions.

CWAC is proposed to address the CoI problems in the decentralized work-flow environment [13] . The modified CWAC for decentralized control that solves the various problems such as, placing the task execution agents in same COI class's, undesirable results that are similar and tend to be in the wrong side of the Chinese Wall.

Formal Concept Analysis CWAC is modelled [12] , results show that the proposed method is able to satisfy the limitation of the Chinese Wall Security Policy and its main properties that are simple security (ss) and *-property. Similarities between CWAC security properties and Multi-Level Security (MLS) policies' security attributes are described by McLean [8]. Bella-LaPadula is one of the MLS policies, it has a simple security rules and *-property that are partly similar to CWAC access rules .According to Gollmann [5], ss-property allows a subject s to access an object o either only if object o has already been accessed by the user or request is entirely different conflict of class. He describes *-property for restricting write access; an object o can be accessed by a subject s only if s/he has no read access to any conflict class object o'. All of the previous works mentioned so far, has not considered a quantitative approach on Chinese Wall Security Model. We believe that our work is the only work that redefines security formula of CWAC and approaches quantitative aspects of it.

3 An Overview of Chinese Wall Security Model

CWAC is one of the most common policy that is usually used in commercial organizations [2]. The main idea behind this is that one is not allowed to access to two conflicting classes or competitor organizations' files at the same time. It can be described quite simply, object that is the wrong side of the wall should not been allowed to be accessed by a subject.

CWAC has two access restriction properties; simple security rule and *-property. Simple security rule gives an access a subject to an object if the object either is in the same company datasets and object already accessed by subject or belongs to a completely different conflict of interest class. Chinese Wall *-property comprises permissions on write access, it is permitted if access is permitted by the simple security rule and object can not be read which is in different organisation dataset than the one which write access is requested for [5].

Our model has similarities with the exist CWAC on the checking of the historical accesses but also we have a new approach that is working relationship. In our case, if a user has accessed to a computer more than n times then we think user has a working relationship with that computer, hence if the user attempts to access a computer that belongs to the CoI, then our property does not allow user. We explain our new model definitions in the following section and we implement our new model on the real world dataset in Section 5.2.

4 A New Data-driven Chinese Walls Model

The existing Chinese Walls model [2] is based on the concept that a subject may have accessed some object in the past that belonged to a conflicting organisation. However, the model is coarsely defined as it relies on the single notion of access. There are no attempts in literature to provide more refined definitions that would incorporate quantitative aspects such as probabilistic or stochastic variations of this model. Therefore, we start with the definition of a *working relation* between a subject and an object to capture such quantification in access control. We start first by reviewing the basic concepts underlying the Chinese Walls model.

Subjects are defined as the set $s, s' \ldots \in \mathcal{S}$. These could be users or computer machines. Objects are defined as the set $o, o' \ldots \in \mathcal{O}$. These could be individual files. Companies (represented by their datasets) are defined as the set $c, c' \ldots \in \mathcal{C}$. There are two labeling operations on objects. The first is $y : \mathcal{O} \to \mathcal{C}$, which defines for an object the company (dataset) to which it belongs. The second labeling operation is $x : \mathcal{O} \to \wp(\mathcal{C})$, which defines for an object, the set of companies (datasets) that are in conflict with its owner. We call this set, the *conflict class* for o.

Based on the above concepts, the history of subjects' access to objects is defined as a matrix $N : \mathcal{S} \times \mathcal{O} \to \mathbb{B}$, where $N_{s,o} = \textbf{True}$ means that s has accessed o in the past, and $N_{s,o} = \textbf{False}$ means it has not. However, in our model, we redefine the history of accesses as a numeric concept rather than a Boolean one.

▶ **Definition 1** (Numeric History of Accesses)**.** We define the numeric history of accesses of objects by subjects as a matrix, $N : \mathcal{S} \times \mathcal{O} \to \mathbb{N}$, which returns for each subject a natural number, n, representing the number of times a subject s has accessed an object o in the past:

$$N_{s,o} = n$$

Using this new definition of the history of accesses, we can define a new relation to capture working relationships between users and computers.

▶ **Definition 2** (Working Relations)**.** We say that a subject s (e.g. a user) has a *working relation* with an object o (e.g. a computer) written as a predicate, $wr(s, o)$, if and only if, based on some predefined minimum number of accesses, n, then:

$$N_{s,o} \geq n$$

In other words, s, in its history, has accessed o at least n number of times. The choice of n depends on the organisation or on the context in which the policy is deployed. By contrast, the standard case where s is deemed to have only *accessed* o, is represented by the next definition.

▶ **Definition 3** (Standard Access)**.** We say that a subject s (e.g. a user) has *accessed* an object o (e.g. a computer) if and only if, based on some predefined minimum number of accesses, n, then:

$$n > N_{s,o} \geq 1$$

In other words, s, in its history, has at least accessed o once, but fewer times than n. Therefore, s has accessed o but not have had a working relation with o. Finally, the case of no access is defined simply as as follows.

```
1, U1, C1
1, U1, C2
2, U2, C3
3, U3, C4
6, U4, C5
7, U4, C5
12, U8, C9
```

Figure 1 Example data lines.

▶ **Definition 4** (No Access)**.** We say that a subject s (e.g. a user) has *not accessed* an object o (e.g. a computer) in its history, if and only if, the following holds true:

$$N_{s,o} = 0$$

We now introduce the new variation of the Chinese Wall accessibility property, based on the definition of a working relation above.

▶ **Property 1** (Working Relations-based Simple Security)**.** *A subject, s, can access an object, o, if and only if, for all objects o', it must be the case that:*

$$(wr(s, o') = \textbf{\textit{True}}) \;\Rightarrow\; (y(o) = y(o') \;\vee\; y(o) \notin x(o'))$$

This property weakens the original Simple Security property defined by Brewer and Nash [2] in that it takes into account only that part of the history of the subject where accesses to objects reached to some particular level of significance (i.e. n as defined in Definition 1). Informally, this means that we are not worried with subjects who have accessed objects fewer times that n in their history.

5 Case Example: LANL Dataset

5.1 Dataset Description

We use here the "User-Computer Authentication Associations in Time" (UCAAT) dataset [6, 9] collected by the Los Alamos National Laboratory (LANL) [1], as our case example to validate the ideas presented in the previous section. We used first five thousands data from the dataset due to memory of the computer that is used for coding. The data ranges over 9 months and represents 708,304,516 successful authentication events from users to computers. An example of some lines in the dataset is shown in Figure 1.

Each line contains three metadata elements; the first represents the time at which the authentication event occurred, the second represents the user who logged in into the computer, and the third the computer on which the login happened. The time epoch starts at 1 with a resolution of 1 second. There are in total 11,362 users, represented by the pseudo values `Ui` and 22,284 computers, represented by the pseudo values `Cj`, where `i, j` represent the number of the user and the computer, respectively. To enhance anonymity, the time frame of the actual data collection is not provided and some centralized computers (e.g. active directory servers) and their associated authentication events have also been removed. The dataset is available either as a single compressed file (size 2.3GB) or as a set of 9 individual files (sizes ranging from 177MB to 273MB).

5.2 Model Implementation Based on the Dataset

We now instantiate our new data-driven Chinese Walls model using data from the UCAAT dataset. Due to the size of the dataset, we selected as a proof-of-concept only the first 5000 entries. Initially, we find the number of times each user logged in to a specific machine. Table 1 shows an example of such analysis.

Table 1 An example table showing the number of times users log in to computers.

User Name	Computer Name	Number of Login Times
U1	C1	2
	C2	4
	C978	2
U12	C54	62
	C94	8
	C801	3
U66	C1	18
	C117	3
	C133	1
U105	C113	2
	C130	3
	C160	37
U106	C136	8
U116	C155	32
U127	C155	41
U13	C14	21
	C172	20
	C282	20
	C32	30

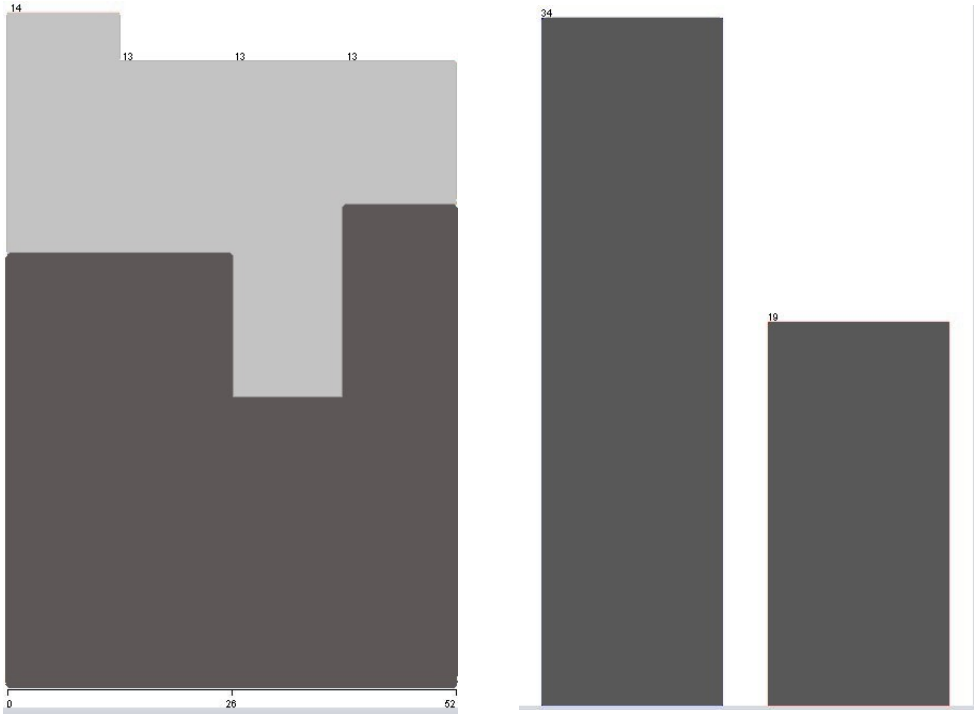
We applied clustering techniques to find the value of n (for definition of n see Section 4. Clustering is a technique that is used to group data together [7]. It is used to classify each data point into a specific group. We give the number of login time, and classify them into two groups. One of the common clustering algorithms is K-means which is a method to partition a data set into k groups [14]. The clustering results are given in Figure 2. The numbers are clustered into two groups and the boundary number of the numbers was 20. Therefore, in our case (for first five thousands data) n value is 20. We use u instead of s, similarly c instead of o. In this way, if

$$N_{u,c} \geq 20$$

then we say user has a working relations with the computer. For example, U12 has working relations with C54 because U12 has logged into C54 68 times. Table 2 represents the full list of users with working relations to computers. Standard access value becomes as follows;

$$20 > N_{u,c} \geq 1$$

For example, U1 has only standard access in relation to all the computers that have been logged in to by U1, because $N_{u,c}$ is between 1 and 20. On the other hand, U13 has working relation with all computers (see Table 1), n is greater than 20. We use the same approach with the Chinese Walls model *simple security*; In our case, if a user wants to login to the computer and s/he has a working relation with a CoI group of computers, then our model



(a) Clustering 1.

(b) Clustering 2.

Figure 2 K means Clustering.

Table 2 The group of users who have working relations with computers.

User Name	Computer Name	User Name	Computer Name
U12	C54,C13	U105	C160
U116	C155	U120	C164
U124	C168, C192	U127	C155
U128	C167, C176	U13	C14, C172, C282, C32, C42
U130	C179	U153	C207
U156	C161	U159	C160
U16	C17, C148	U179	C175
U184	C154	U188	C256
U197	C269	U202	C275
U204	C101	U21	C22
U211	C286	U216	C291
U29	C30	U39	C41
U53	C58	U6	C7
U60	C108, C173	U63	C234, C68
U66	C1	U67	C49
U68	C71	U77	C234, C78
U93	C107,C53	U92	C154
U95	C154	U97	C161
U105	C160	U116	C155
U120	C164	U9	C10
U96	C115	U97	C116, C161

does not allow her/him to access the computer. A user, s, can log into a computer, o, if and only if, for all computers o', it is the case that:

$$(wr(u, c') = \textbf{True}) \Rightarrow (y(c) = y(c') \vee y(c) \notin x(c'))$$

The property allows a user if and only if, s/he has either standard access or, has no access which means that s/he has not logged-in to the computers. However, if a user has working relation, then s/he can not be permitted access to computers that belong to a conflict class. Table 2 represents the full list of users with working relations to computers.

6 Conclusion and Future Work

In this paper, we first defined a robust, quantitative model for Chinese Walls Access Control. We showed that our new model is suitable to apply on the real world dataset, in our case, we used the UCAAT dataset for implementing our quantitative Chinese Walls Model. We then explained the way to find out the value of the n that shows boundary of access times by applying clustering algorithms. Our model allows to put access restrictions on the sensitive and personal information so that users cannot manipulate other users' sensitive files for their own advantages. There is an opportunity for the further work, we plan to extend this work with quantitative description of *property. We also believe that our model can be applied other Multi Level Security (MLS) models.

References

1 Los Alamos National Laboratory: Cyber Security Science. `https://csr.lanl.gov/data/`. Accessed: 14-06-2018.

2 D.F.C. Brewer and M.J. Nash. The Chinese Wall Security Policy. In *Proceedings of the* 1989 *IEEE Symposium on Security and Privacy*, pages 206–214, Oakland, California, USA, 1989. IEEE Computer Society Press.

3 Pau–Chen Cheng, Pankaj Rohatgi, Claudia Keser, Paul A. Karger, Grant M. Wagner, and Angela Schuett Reninger. Fuzzy Multi-Level Security:An experiment on quantified risk-adaptive access control. In *2007 IEEE Symposium on Security and Privacy (S&P 2007), 20-23 May 2007, Oakland, California, USA*, pages 222–230. IEEE Computer Society, 2007. `doi:10.1109/SP.2007.21`.

4 Dhillon G. and G. Torkzadeh. Value-focused assessment of information system security in organizations. *Information Systems Journal*, 16(3), 293-314, 2015.

5 Dieter Gollmann. *Computer Security*. John Wiley & Son Ltd, 1999.

6 Aric Hagberg, Alex Kent, Nathan Lemons, and Joshua Neil. Credential hopping in authentication graphs. In *2014 International Conference on Signal-Image Technology Internet-Based Systems (SITIS)*. IEEE Computer Society, 2014.

7 Anil K. Jain. Data clustering: 50 years beyond K-means. *Pattern Recognition Letters*, 31(8):651–666, 2010.

8 McLean John. The algebra of security. In *In Security and Privacy*, volume 1290, pages 2–7. IEEE Symposium, 1988.

9 Alexander D. Kent. User-Computer Authentication Associations in Time. Los Alamos National Laboratory, 2014. `doi:10.11578/1160076`.

10 Butler Lampson. A note on the confinement problem. *Communications of the ACM*, 16(10):613–615, 1973.

11 S. C. Mouliswaran, C. A. Kumar, and C Chandrasekar. Modeling Chinese wall access control using formal concept analysis. In *2014 International Conference on Contemporary Computing and Informatics (IC3I)*. IEEE, 2014. `doi:10.1109/IC3I.2014.7019619`.

12 S. C. Mouliswaran, C. A. Kumar, and C Chandrasekar. Modeling Chinese wall access control using formal concept analysis. In *2014 International Conference on Contemporary Computing and Informatics (IC3I)*. IEEE, 2014. doi:10.1109/IC3I.2014.7019619.

13 Atluri Vijayalakshmi, Chun Soon, and Mazzoleni Pietro. A Chinese wall security model for decentralized workflow systems. In *CCS 2001, Proceedings of the 8th ACM Conference on Computer and Communications Security, Philadelphia, Pennsylvania, USA, November 6-8, 2001.*, pages 48–57. ACM, 2001. doi:10.1145/501983.501991.

14 Kiri Wagstaff, Claire Cardie, Seth Rogers, and Stefan Schrödl. Constrained k-means clustering with background knowledge. In Carla E. Brodley and Andrea Pohoreckyj Danyluk, editors, *Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001), Williams College, Williamstown, MA, USA, June 28 - July 1, 2001*, pages 577–584. Morgan Kaufmann, 2001.

Comparison of Platforms for Recommender Algorithm on Large Datasets

Christina Diedhiou
School of Computing, University of Portsmouth, United Kingdom
christina.diedhiou@port.ac.uk

Bryan Carpenter
School of Computing, University of Portsmouth, United Kingdom
bryan.carpenter@port.ac.uk

Ramazan Esmeli
School of Computing, University of Portsmouth, United Kingdom
Ramazan.Esmeli@myport.ac.uk

Abstract

One of the challenges our society faces is the ever increasing amount of data. Among existing platforms that address the system requirements, Hadoop is a framework widely used to store and analyze "big data". On the human side, one of the aids to finding the things people really want is recommendation systems. This paper evaluates highly scalable parallel algorithms for recommendation systems with application to very large data sets. A particular goal is to evaluate an open source Java message passing library for parallel computing called MPJ Express, which has been integrated with Hadoop. As a demonstration we use MPJ Express to implement collaborative filtering on various data sets using the algorithm ALSWR (Alternating-Least-Squares with Weighted-λ-Regularization). We benchmark the performance and demonstrate parallel speedup on Movielens and Yahoo Music data sets, comparing our results with two other frameworks: Mahout and Spark. Our results indicate that MPJ Express implementation of ALSWR has very competitive performance and scalability in comparison with the two other frameworks.

2012 ACM Subject Classification Information systems → Database management system engines

Keywords and phrases HPC, MPJ Express, Hadoop, Spark, Mahout

Digital Object Identifier 10.4230/OASIcs.ICCSW.2018.4

Category Main Track

1 Introduction

Over the last decade Apache Hadoop has established itself as a pillar in the ecosystem of software frameworks for "big data" processing. As an open source, mostly Java-based Apache project with many industrial contributors, it retains a commanding position in its field.

When first released Hadoop was a platform primarily supporting the MapReduce programming model, and other projects built on top of MapReduce. Around 2014 with the release of Hadoop 2.0 the platform was re-factored into a separate YARN (Yet Another Resource Negotiator) resource allocation manager, with MapReduce now just one of multiple possible distributed computation frameworks that could be supported on top of YARN. Several other major big data projects rapidly migrated to allow execution on the Hadoop YARN platform (for example Apache Spark [19], Apache Giraph [2], Apache Tez [13], and Microsoft Dryad [9]). Around the same time the present authors envisaged adding our existing MPJ Express

framework for MPI-like computation in Java to that distinguished group, and developed a version of our software that could also run under Hadoop YARN [17].

MPJ Express is a relatively conservative port of the standard MPI 1.2 parallel programming interface to Java, and is provided with both "pure Java" implementations (based on Java sockets and threads) and "native" implementations exploiting specific interconnect interfaces, or implementations on top of standard MPI. The vision was thus to support MPJ as one computational framework among many largely Java-based or JVM-based frameworks that could be mixed and matched for different stages of complex big data processing, with Hadoop and HDFS (the Hadoop Distributed File System) as the "glue" between stages. More details on MPJ Express and Hadoop can be found in appendix A.

The main goal of the present paper is to provide evidence that such a scenario can be realized and that it may be advantageous. We concentrate on one particular computationally intensive "big data" problem - generating product recommendations through the collaborative filtering algorithm ALSWR (Alternating Least Squares with Lambda Regularization). A version of this algorithm is developed and evaluated using MPJ running under Hadoop. We then go on to compare our implementation with two existing implementations of ALSWR that can run under Hadoop – one taken from the Apache Mahout project using MapReduce, and one using Apache Spark. Results suggest the MPJ approach can provide useful performance gains over these other established Big Data frameworks on suitable compute-intensive kernels.

The rest of the paper is organized as follows. Section 2 gives an overview of the collaborative filtering technique. Section 3 describes how we implement the collaborative filtering with ALSWR in MPJ. The Section 4 evaluates and compare our results with Mahout and Spark. Section 5 concludes the paper and discusses future work.

2 Collaborative Filtering Techniques

Recommender systems are software tools and techniques that provide suggestions to users to help them find and evaluate items likely to match their requirements. Collaborative filtering systems are based on users' purchases or decisions histories. Assuming two individuals share the same opinion on an item, they are also more likely to have similar taste on another item. In our experiments we have opted for a model based approach and we specifically use Alternating-Least-Squares with Weighted-λ-Regularization (ALSWR) algorithm.

In this section, we will often refer to items as "movies" due to the fact that one of our datasets consist of ratings on movies. Assuming we have n_u users and n_m movies, and R is the $n_u \times n_m$ matrix of input ratings. Usually each user can rate only few movies. Therefore the matrix R will initially have many missing values or loosely speaking it will be sparse. The problem is to predict the unknown elements of R from the known elements.

We model the preferences of users by assuming they have simple numeric level of preference for each of a number n_f of features to be found in movies; thus the behaviour of user i is modelled by a vector $\mathbf{u}_i$ of length n_f. Similarly each movie is assumed to have each these features to a simple numeric degree so each movie j is modelled by a vector $\mathbf{m}_j$ of the same size. The predicted preference of user i for movie j is the dot product $\mathbf{u}_i \cdot \mathbf{m}_j$. The vectors are conveniently collected together in matrices U and M of size $n_u \times n_f$ and $n_m \times n_f$ respectively. To fit the model to the known elements of R we use a least squares approach, adding a regularization term parameter λ to the sum of square deviations to prevent the

model from overfitting the data. The penalty function ALSWR strives to minimize is:

$$f(U,M) = \sum_{i,j}(r_{ij} - \mathbf{u}_i \cdot \mathbf{m}_j)^2 + \lambda \left(\sum_i n_{u_i}\mathbf{u}_i^2 + \sum_j n_{m_j}\mathbf{m}_j^2 \right) \tag{1}$$

where the first sum goes over i, j values where the element r_{ij} of R is known in advance, n_{u_i} is the number of items rated by a user i, and n_{m_j} is the number of users who have rated a given movie j.

ALSWR is an iterative algorithm. It shifts between fixing two different matrices. While one is fixed, the other one is updated hence solving a matrix factorization problem. The same process goes through a certain number of iterations until a convergence is reached which implies that there is little or no more change on either users and movies matrices. The ALSWR algorithm as explained by Zhou et al [20] is as follows:

- Step 1: Initialize matrix M in a pseudorandom way.
- Step 2: Fix M, Solve U by minimizing the objective function (the sum of squared errors);
- Step 3: Fix U, Solve M by minimizing the objective function similarly;

Steps 2 and 3 are repeated until a stopping criterion is satisfied. Step 2 is implemented by Equation 2 where M_{I_i} is the sub matrix of M, representing the selection of any column j in the set of movies rated by a user i, H is a unit matrix of rank equal to n_f and $R(i, I_i)$ is the row vector where columns j are chosen

$$\mathtt{u}_i = (M_{I_i} M_{I_i}^T + \lambda n_{u_i} H)^{-1} M_{I_i} R^T(i, I_i) \tag{2}$$

Step 3 is implemented by a similar formula exchanging the roles of U and M.

3 MPJ Implementation of ALSWR

The basic strategy for distributing the ALSWR algorithm to run in parallel was already described by the original proposers in [20]. All nodes of a cluster contain a certain subset of the large, sparse, recommendations array, R. In particular it is convenient for the R array to be stored in two ways across the cluster as a whole – divided across nodes by columns and also by rows. This is illustrated in figure 1, where i is the subscript identifying users and j is the subscript identifying items, and the two different forms of decomposition of R are used in the two different steps. Step 2, as defined in equation 2, conveniently uses locally held R decomposed by i to update locally owned elements u_i of the user model. B is a block size for the locally held subset of elements, approximately constant across the cluster for good load balancing.

Because update of u_i potentially involves *any* element of the item model $\mathbf{m}$, to simplify this step all elements of $\mathbf{m}$ should be stored locally, in globally replicated fashion. Step 2 has a complementary structure, but now update of m_j may require access to any element of $\mathbf{u}$. So between steps 1 and 2 all the locally computed elements of $\mathbf{u}$ must be gathered together and broadcast to processing nodes. Similarly between step 2 and step 3 in the *next* iteration of the algorithm, the locally computed elements of $\mathbf{u}$ must be gathered and broadcast. A great benefit of the MPI style of programming is the use of *collective communication* This is embodied here in the use of MPI_Allgather, that allows data to be gathered from each process then to be distributed to all processes. In our program the data that we used for the implementation of the ALSWR code consists of a sparse matrix of ratings, partitioned by

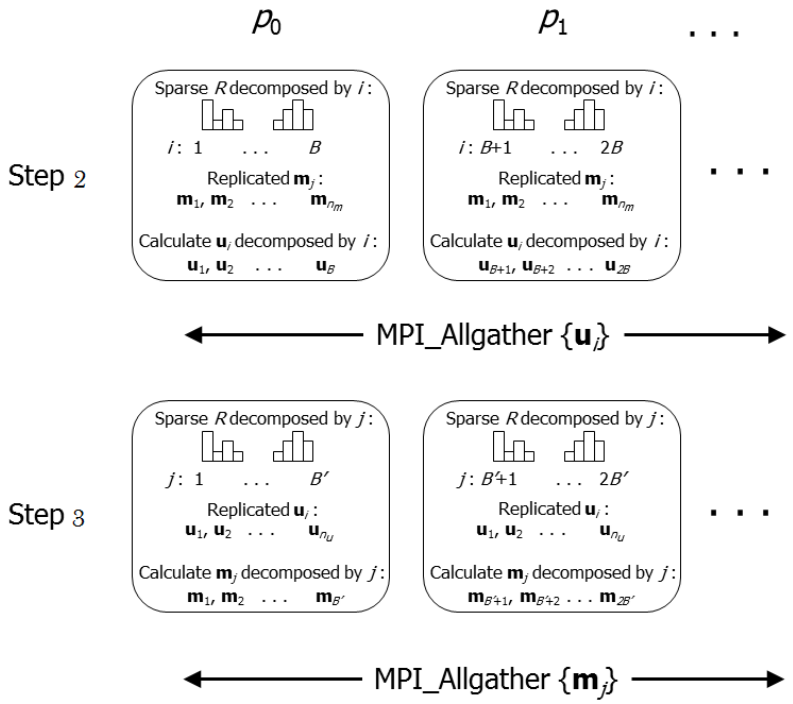


Figure 1 Visualization of an iteration of distributed ALSWR algorithm. "Processor space" runs across the page, processes are labelled $p_0, p_1, \ldots$ and so on. Time runs down the pages with distributed computational steps labelled as on page 3. Between computational stages there are collective synchronizations in the form of "allgather" operations.

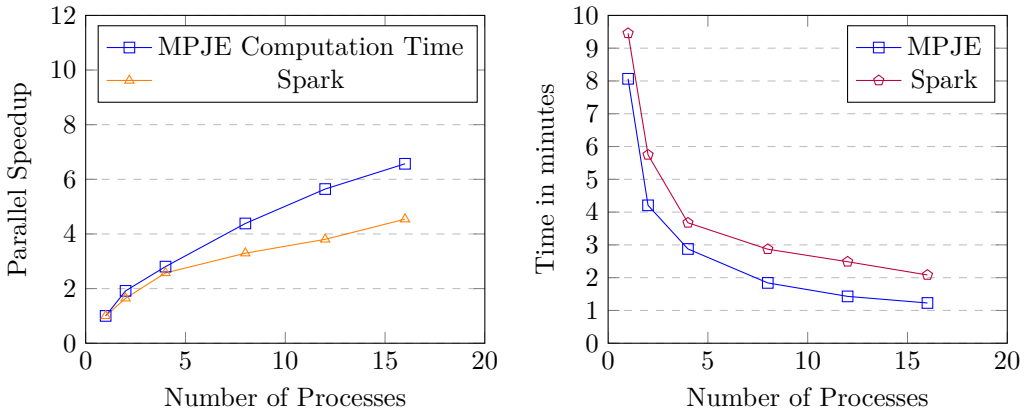
user or by item. Figure 6 in the appendices section illustrates the organization of the data. In order to solve the symmetric positive definite matrix we use Cholesky decomposition from the Intel Data Analytics Acceleration Library (DAAL) [8].

The code assumes each node holds `numLocal` elements of the distributed user model. Within a node we run `NUM_THREADS` long lived threads (they are started at the beginning of the program), where the `NUM_THREADS` parameter will be related to the number of cores on the node. The variable `me` identifies a thread within the local node (not to be confused with the MPI *rank* which identifies a node). Threads will be synchronized before MPI collective operations using barriers implemented by `java.util.concurrent.CyclicBarrier`. The MPI operations themselves are only executed by the `me` = 0 thread.

The ratings data for our MPJ code are read from the same HDFS text files as used by the third party implementations of ALS discussed below. We use HDFS API to determine the blocks that have replicas on nodes running MPJ processes. A heuristic is used to choose a load balanced set of local replicas to read. The locally read ratings are then partitioned to destination nodes using a variant of the CARI communication schedules introduced in [1].

4 Performance Evaluation and Comparison of MPJ Express, Mahout, and Spark

This section details our experiments focusing on the comparative performance evaluation of MPJ Express against well-known platforms including Hadoop, Mahout and Spark. The performance evaluation compares their parallel speedup. More information on Apache



(a) Parallel Speedup MPJE vs Spark. **(b)** MPJE vs Spark.

Figure 2 Frameworks Performance Comparison MPJ Express with MovieLens dataset.

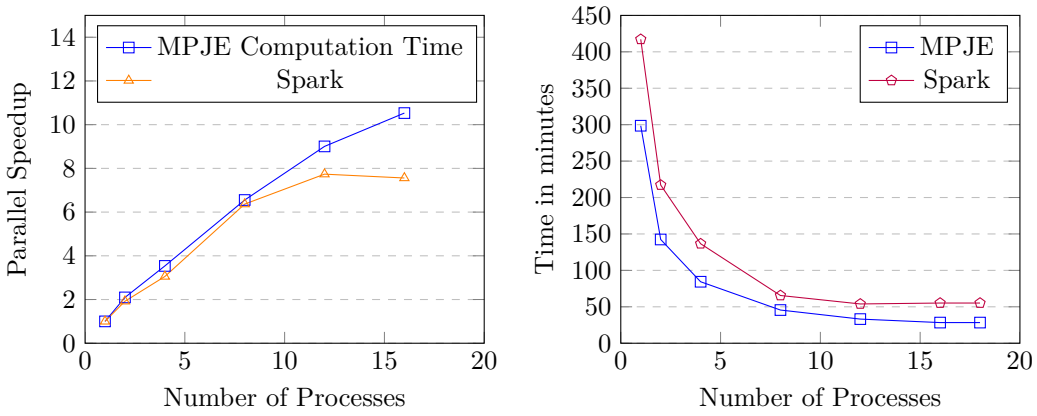
Mahout and Spark is provided in appendix B. For the purpose of performance evaluation, we acquired our datasets from public domains. These consist of anonymous user ratings from two different sources: MovieLens and Yahoo Music. Details on our datasets and test environment are provided in Appendix C.

4.1 MovieLens 20M Ratings Experiments

Our ALSWR code is tested with 50 features, 10 iterations, 0.01 for the regularization parameter lambda λ and 0.01 for the parameter epsilon ϵ that is used in the initial guess for the item model. MPJ Express and Spark have both a good performance and parallel speed up: as the number of cores increases the time decreases; Mahout does not show much variances from four cores and above. Refer to Figure 5 in the Appendices section which compares the performances between MPJ Express, Spark and Mahout on different number of processes. Figure 2b focus on MPJ and Spark. MPJ Express has the best performance amongst the 3 frameworks. It is, on average, 13.19 times faster than Mahout and on average 1.4 faster than Spark. Figure 2a represents the parallel speedup of MPJ Express and Spark. With sixteen cores MPJ Express is almost 10 times faster than when it is run in sequence while Spark is just about 4.5 times faster than its result with one process.

4.2 Yahoo Webscope 700M Ratings Experiments

Mahout was unable to cope with the large Yahoo dataset. For this reason, we have evaluated only MPJ Express and Spark versions of the code for this dataset. Figure 3b shows a pattern quite similar to figure 2b although this time our dataset is about 35 times bigger. A closer look at figure 3a demonstrates a significant improvement regarding the parallel speedup of MPJ Express which now runs more than 10.5 times faster on 16 cores than its sequential time. The parallel speedup of Spark has also improved. It implements the ALS on Yahoo dataset 7.5 times faster with 16 cores than when it is run in sequence. However from 16 cores onwards, the performance of the Spark version starts decreasing.



(a) Parallel Speedup MPJE vs Spark.

(b) MPJE vs Spark.

Figure 3 Frameworks Performance Comparison MPJ Express with Yahoo dataset.

4.3 Analysis of the results

The Mahout implementation of ALS – not necessarily representative of the wider Mahout project – is based on MapReduce. The performance limitations of MapReduce on iterative algorithms are well documented, see for example [5]. According to pseudocode given in [19], the Spark implementation uses a combination of its *parallelize* and *collect* operations to reproduce the communication operation called *MPI_Allgather* here. We assume that the MPI collective algorithms can implement this pattern more efficiently. There is a discussion of efficient implementations of Allgather in [14] for example. Additionally there may be some degradation of the performance of Spark when there is not enough memory (RAM) as the storage has to be on disk when the program is running out of space.

5 Conclusion

Various computational frameworks have been adopted over the last few years for running compute-intensive kernels of recommender algorithms on Hadoop platforms. These include Apache Mahout, Apache Spark and Apache Giraph. In this paper we have added our MPJ Express framework to this list, and provided evidence that it can outperform other implementations of the central optimization algorithm. This additional performance certainly comes at some cost in terms of programming complexity. For example the MPJ programmer has to spend more time orchestrating communication between Hadoop nodes. Nevertheless we argue that for some intensive and often used kernels, the extra investment in programming may be justified by the potential performance gains. We see MPI-based processing stages as one more resource in the armoury of big data frameworks that may be used in processing pipelines run on Hadoop clusters. We also suggest that in this setting MPJ Express may be a natural choice of MPI-like platform, given that many other such processing stages will be coded in Java or JVM-based languages. On our future work we need to evaluate alternative parallel organizations of the recommender code, like the rotational hybrid approach described in [10]. Preliminary analysis suggests that implementation of similar schemes in MPI style may benefit from extensions to the standard set of MPI collectives, currently embodied in MPJ Express. Again such an extended library could form part of a future data centric version of MPJ Express that builds on experiences of MPI processing in the Hadoop environment.

References

1 Wakeel Ahmad, Bryan Carpenter, and Aamir Shafi. Collective Asynchronous Remote Invocation (CARI): A High-Level and Efficient Communication API for Irregular Applications. *Procedia Computer Science*, 4:26–35, 2011. International Conference On Computational Science, ICCS 2011.

2 Apache Giraph. http://giraph.apache.org/, 2014. [accessed 19-January-2018].

3 Apache Mahout. https://mahout.apache.org/, 2017. [accessed 30-January-2018].

4 Spark RDD Operations-Transformation & Action with Example. https://data-flair.training/blogs/spark-rdd-operations-transformations-actions/. [accessed 11-June-2018].

5 Rui Maximo Esteves, Rui Pais, and Chunming Rong. K-means clustering in the cloud–a Mahout test. In *Advanced Information Networking and Applications (WAINA), 2011 IEEE Workshops of International Conference on*, pages 514–519. IEEE, 2011.

6 Datasets | GroupLens. http://grouplens.org/datasets/, 2015. [accessed 14-December-2016].

7 Rong Gu, Xiaoliang Yang, Jinshuang Yan, Yuanhao Sun, Bing Wang, Chunfeng Yuan, and Yihua Huang. SHadoop: Improving MapReduce performance by optimizing job execution mechanism in Hadoop clusters. *Journal of parallel and distributed computing*, 74(3):2166–2179, 2014.

8 Data Analytics Acceleration Library. https://software.intel.com/en-us/Intel-daal, 2017. [accessed 21-October-2017].

9 Michael Isard, Mihai Budiu, Yuan Yu, Andrew Birrell, and Dennis Fetterly. Dryad: Distributed Data-parallel Programs from Sequential Building Blocks. In *Proceedings of the 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems 2007*, EuroSys '07, pages 59–72, New York, NY, USA, 2007. ACM.

10 Maja Kabiljo and Aleksandar Ilic. Recommending items to more than a billion people. https://code.facebook.com, 2015. [accessed 30-December-2017].

11 Introduction to ALS Recommendations with Hadoop. https://mahout.apache.org/users/recom-mender/intro-als-hadoop.html. [accessed 22-June-2018].

12 MPJ Express. http://mpjexpress.org, 2015. [accessed 18-January-2018].

13 Bikas Saha, Hitesh Shah, Siddharth Seth, Gopal Vijayaraghavan, Arun Murthy, and Carlo Curino. Apache tez: A unifying framework for modeling and building data processing applications. In *Proceedings of the 2015 ACM SIGMOD international conference on Management of Data*, pages 1357–1369. ACM, 2015.

14 Rajeev Thakur, Rolf Rabenseifner, and William Gropp. Optimization of collective communication operations in MPICH. *The International Journal of High Performance Computing Applications*, 19(1):49–66, 2005.

15 Tom White. *Hadoop: The Definitive Guide*. O'Reilly Media, Inc., 4th edition, 2015.

16 Webscope. https://research.yahoo.com/, 2006. [accessed 14-December-2016].

17 Hamza Zafar, Farrukh Aftab Khan, Bryan Carpenter, Aamir Shafi, and Asad Waqar Malik. MPJ Express Meets YARN: Towards Java HPC on Hadoop Systems. *Procedia Computer Science*, 51:2678–2682, 2015. International Conference On Computational Science, ICCS 2015. doi:10.1016/j.procs.2015.05.379.

18 Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauley, Michael J Franklin, Scott Shenker, and Ion Stoica. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*, pages 2–2. USENIX Association, 2012.

19 Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, and Ion Stoica. Spark: Cluster Computing with Working Sets. In *Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing*, HotCloud'10, pages 10–10, Berkeley, CA, USA, 2010. USENIX Association. URL: `http://dl.acm.org/citation.cfm?id=1863103.1863113`.

20 Yunhong Zhou, Dennis Wilkinson, Robert Schreiber, and Rong Pan. Large-scale parallel collaborative filtering for the netflix prize. *Lecture Notes in Computer Science*, 5034:337–348, 2008.

A Description of Hadoop and MPJ Express

A.1 Hadoop Overview

Hadoop is a framework that stores and processes voluminous amounts of data in a reliable, fault-tolerant manner [15].

Since Hadoop 2, YARN (Yet Another Resource Negotiator) has been integrated in the infrastructure as the resource manager, enabling many other distributed frameworks besides MapReduce to process their data on Hadoop cluster. YARN depends on three main components to complete a task: a Resource Manager (RM), Node Managers (NMs), and an Application Master (AM). The RM is responsible for managing and allocating the resources across the cluster. NMs run on all nodes available in a cluster and report all the tasks to the RM such as the number of cores and memory space. Each job that is started has an AM specific to the processing framework that manages operation within containers and ensures there are sufficient containers for the task. The communication between the master nodes and slave nodes is achieved through the Heart Beat Mechanism [7].

A.2 MPJ Express

MPJ Express [12] is an open source Java MPI-like library that allows application developers to write and execute parallel applications on multicore processors and compute clusters. The MPJ Express software can be configured in cluster or multicore. Under the cluster configuration, the MPJ Express software provides different communication devices that are suitable for the underlying interconnect. Currently, there are four communication devices available:

1. niodev - uses Java New I/O (NIO) Sockets
2. mxdev - uses Myrinet eXpress (MX) library for Myrinet networks
3. hybdev - for clusters of multicore processors
4. native - uses a native MPI library (like MPICH, MVAPICH, OpenMPI)

Since 2015, the MPJ Express software provides a YARN-based runtime that exploits the niodev communication device to execute parallel Java code on Hadoop clusters. Under this setting, HDFS is used as the distributed file system where application datasets, MPJ Express libraries, and application programs are loaded to allow all processes to access the material.

B Description of Apache Mahout and Spark

Apache Mahout is a distributed linear algebra framework [3], widely used for its distributed implementation on Apache Hadoop. This essentially means that datasets are stored on the HDFS and various machine learning algorithms such as collaborative filtering can be

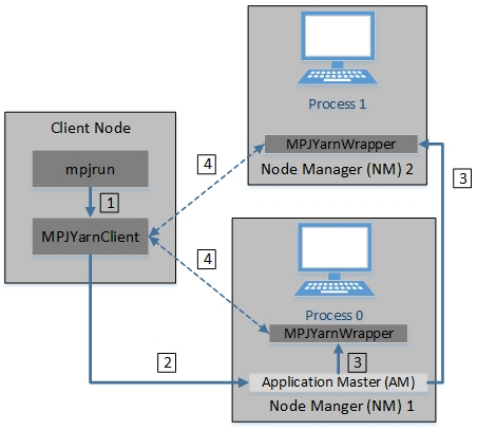


Figure 4 MPJ Express Integrated in YARN- 1) Submit YARN application- 2) Request container allocation for AM- 3) AM generates a CLC and allocates container to each node- 4) Each mpj-yarn-wrapper send outputs and error streams of the program to the MPJYarnClient.

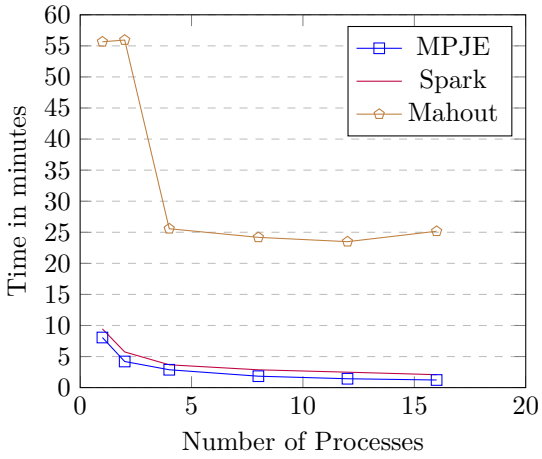


Figure 5 MPJ Express, Spark and Mahout on MovieLens Dataset.

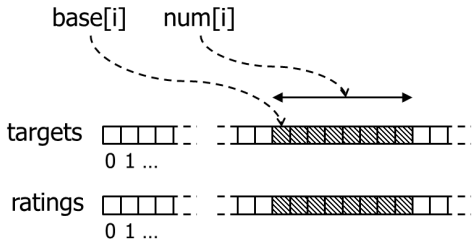


Figure 6 Sparse data structure to represent locally held ratings. This whole structure is duplicated, once for ratings distributed by user and once for ratings distributed by items. In the "by user" case the size of the `base` and `num` arrays is the total number of locally held users, with `num[`i`]` being the number of ratings by user i; `targets` elements hold a *global* index of the rated item (index in the gathered array of item models). In the "by item" case the size of the top arrays is the number of locally held items, with `num` holding the number of ratings per item; a `target` element now holds the global index of the user who *made* the rating.

applied to the data. The ALSWR implementation with Apache Mahout is done through its machine learning library and more specifically the map-reduce implementation of ALS. This last consists of two stages: a parallel matrix factorization phase followed up by some recommendations. Both phases are detailed in [11].

Apache Spark is an open-source cluster-computing framework suitable for large scale data processing. Since Hadoop 2, Spark has been integrated with Hadoop allowing its programs to run on YARN. Spark can use memory and disk processing through its Resilient Distributed Datasets (RDD). As explained in [18], the default is to keep the RDD in memory; when there is no more space in the RAM, Spark stores the rest on disk. Shared variables and parallel operations available in Spark are detailed in [19] and [4]. We have implemented ALS on Spark through its standard machine learning library (MLlib).

C Description of Datasets and testing environment

The dataset obtained from MovieLens contains $20,000,263$ ratings for $27,278$ movies, created by $138,493$ users [6]. The dataset from Yahoo Music – that is much larger – contains over 717 millions ratings for 136 thousands songs rated by 1.8 million users [16]. The data from Yahoo has been separated in training and test datasets. Our test environment includes a Linux cluster composed of 4 nodes with 20 cores in total. The software used for the tests consist of:

- Java 1.7
- Apache ant 1.6.2
- Hadoop-2.7.3
- MPJ Express (version 0.44), Mahout (version 0.12.2), and Spark (version 2.2.0)
- Intel Data Analytics Acceleration Library (DAAL) 2017

Towards Context-Aware Syntax Parsing and Tagging

Alaa Mohasseb
School of Computing, University of Portsmouth, United Kingdom
alaa.mohasseb@port.ac.uk

Mohamed Bader-El-Den
School of Computing, University of Portsmouth, United Kingdom
mohamed.bader@port.ac.uk

Mihaela Cocea
School of Computing, University of Portsmouth, United Kingdom
mihaela.cocea@port.ac.uk

Abstract

Information retrieval (IR) has become one of the most popular Natural Language Processing (NLP) applications. Part of speech (PoS) parsing and tagging plays an important role in IR systems. A broad range of PoS parsers and taggers tools have been proposed with the aim of helping to find a solution for the information retrieval problems, but most of these are tools based on generic NLP tags which do not capture domain-related information. In this research, we present a domain-specific parsing and tagging approach that uses not only generic PoS tags but also domain-specific PoS tags, grammatical rules, and domain knowledge. Experimental results show that our approach has a good level of accuracy when applying it to different domains.

2012 ACM Subject Classification Computing methodologies → Natural language processing

Keywords and phrases Information Retrieval, Natural Language Processing, PoS Tagging, PoS Parsing, Machine Learning

Digital Object Identifier 10.4230/OASIcs.ICCSW.2018.5

Category Main Track

1 Introduction

Parts-of-speech (PoS) tags play an important role in Natural Language Processing (NLP). PoS tagging provides a large amount of information about words. PoS parsing and tagging is one of the fundamental phases in text processing. Parsing has been used as a way to identify the sentence structure by adding mark-ups which helps in organizing a sentence, while tagging represent classes and features of words, in which each word will receive a tag based upon its word class and the feature it holds.

A broad range of PoS parsing and tagging tools and approaches have been developed; most of these tools and approaches are based on natural language. Furthermore, parsers and taggers still suffer from the problem of domain adaptation [21],[13] since most of them are based just on generic NLP tags which have a limited use in domains such as search engines, question answering systems and social networks; knowing only the generic PoS tags will not assist in identifying and retrieving relevant information since a lot of knowledge related to most of these domains cannot be captured with generic PoS tags. Moreover, most parser and tagger methods do not take inconsideration the syntax and grammatical structure of the given text.

In this paper, we propose a Domain Specific Syntax-based Parsing and Tagging (DSSPT) approach. The aim of the research presented in this paper is to evaluate the influence of using domain-specific grammatical rules categories on the the parsing and tagging process and the classification performance. In addition, we aim to evaluate the use of DSSPT on two different domains: query classification and question classification.

The rest of the paper is organized as follows. Section 2 outlines previous work in parsing and tagging, including different proposed tools and approaches. Section 3 describes the proposed parsing and tagging framework. The experiments setup and results are presented in Section 4. Finally, Section 5 concludes the paper and outlines directions for future work.

2 Background

In this section we review previous work on parsing and tagging. Different methods of parsing are outlined in Section 2.1, while Section 2.2 reviews previous work on tagging methods.

2.1 Parsing

Many recent studies proposed different parsing methods and models; some of these are based on dependency parsing. Authors in [21] developed distant-supervised algorithms that use a dependency grammar for Community Question Answering (CQA). In [32] authors developed a graph-based and a transition-based dependency parser using beam-search, while in [8] a simple semi-supervised method for training dependency parsers was presented. Authors in [19] introduced MaltParser, a data-driven parser generator for dependency parsing. Some works used machine learning algorithms. Authors in [3] proposed a dependency parser using neural networks, while authors in [27] introduced algorithms to derive a query's syntactic structure from the dependency trees. Furthermore, in [26] authors proposed a general compositional vector framework for transition based dependency parsing. Other works introduced a semantic-based parser model. Works in [9] presented a semantic parsing model for answering compositional questions. Moreover, in [30] authors presented a statistical natural language semantic parsing modeling, while in [31] authors proposed a semantic parsing framework for question answering. Authors in [24] introduced a Compositional Vector Grammar (CVG), which combines probabilistic context-free grammar (PCFGs). In [11] authors proposed an algorithm of text parsing which was demonstrated on data from Twitter, while in [25] a recursive neural network architecture was introduced. Finally, authors in [28] proposed a technique for improving parser portability.

2.2 Tagging

Most taggers and tagging approaches have been developed for general PoS tagging. Authors in [20] proposed a tag-set that consists of twelve universal PoS categories. In [1] the authors proposed a Trigrams'n'Tags (TnT) statistical PoS tagger. Moreover, work in [4] proposed a PoS tagger based on Support Vector Machines (SVMT). Other works like [29] proposed a PoS tagger using dependency network representation. In [10] authors presented a method for unsupervised PoS tagging that considers a word type. Furthermore, few taggers have been developed for specific domains. In [5] authors addressed the problem of PoS tagging for English data from Twitter. In [7] a PoS tagging method for web search queries was proposed using the sentence level morphological analysis, while in [22] a probabilistic tagging method was proposed, which avoids the problems of Markov model based taggers. Finally, authors in [12] introduced an approach for deep parsing of web search queries using a context-free multiset generating grammar.

3 Proposed Approach

3.1 Tag-set

The tag-set was developed by [18] and updated by [17]. It was mainly created for the purpose of identifying search queries by labelling each word in the query with its PoS tag and name entity to help in the classification of the users' intent. The tag-set has been tested on different search engines' queries datasets [17], [15], i.e. AOL 2006 data-set[1] and the TREC 2009 Million Query Track data-set[2]. Furthermore, it has been used in other domains such as question classification [16] and also has been tested on different questions datasets, i.e. Yahoo Non-Factoid Question Dataset[3], TREC 2007 question answering data[4] and a Wikipedia dataset[5] that was generated by [23].

The tag-set consists of 10,440 different words that have been labelled with PoS tags (categories) which include three levels of details from our grammar taxonomy: (1) Level 1 includes the seven major word classes in English, which are *Verb (V), Noun (N), Determiner (D), Adjective (Adj), Adverb (Adv), Preposition (P)* and *Conjunction (Conj)* ; (2) Level 2 consists of sub-categories of level 1 – for example, *Common Nouns (CN), Proper Nouns (PN)* and *Action Verbs (AV)*; the six main question words: *How, Who, When, Where, What* and *Which* have also been added to this level; (3) Level 3 consists of all the domain-specific categories – for example, *Proper Noun Celebrity (PN_C)* and *Proper Noun Geographical Areas (PN_G)*. A list of all the syntactic categories and corresponding acronyms is displayed in Appendix A.

3.2 Domain-specific syntax-based parsing and tagging

We proposed a Domain-Specific Syntax-based Parsing and Tagging (DSSPT), shown in Figure 1, for the objective of assigning not just PoS tags but also domain specific ones to help in the categorization and classification of text in different domains. The aim of this approach is to create a simple parser and tagger that could easily be applied to different domains by creating domain specific grammatical rules, in which each text is transformed to a domain-specific category using these rules. The grammatical rules contain in addition to typical categories of English grammar, domain-related grammatical categories. The domain specific syntax based parsing and tagging (DSSPT) is described below.

Phase 1: Grammar: In this phase input text is analyzed using domain knowledge and term taxonomy; this is done by identifying each keywords and phrases using the proposed tag-set. Next, the grammar is generated by identifying terminal and non-terminals nodes; the grammar in this phase is based on the Context-Free Grammar (CFG) which capture and combine two different components, i.e. the sentence structure and domain knowledge.

The target in this paper is to use a simple version of the English grammar combined with domain-specific syntactic categories since most domains do not follow entirely the formal English grammar and natural language.

Creating the grammatical rules helps with the identification of ambiguous terms since two different sentences may have similar terms but different structures, each having a different

[1] http://www.researchpipeline.com/mediawiki/index.php?title=AOL_Search_Query_Logs
[2] http://trec.nist.gov/data/million.query09.html
[3] https://ciir.cs.umass.edu/downloads/nfL6/
[4] http://trec.nist.gov/data/qa/t2007_qadata.html
[5] https://www.cs.cmu.edu/~ark/QA-data

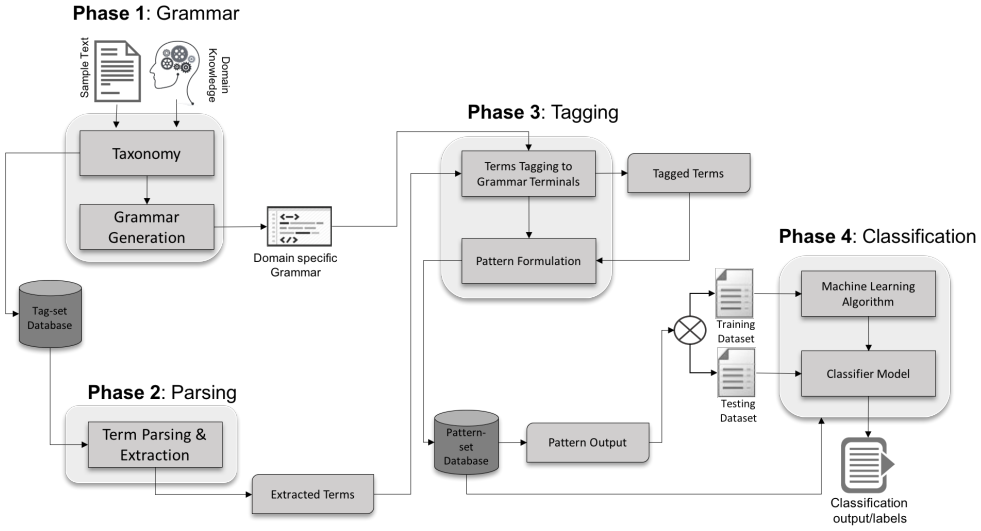


Figure 1 Framework.

meaning, which may lead to different intents. For the given examples *"Order Ed Sheeran Albums"* and *"Ed Sheeran Albums Order"*, the grammatical rules will identify the structure of the sentence at three levels: (1) at phrase level, (2) at words level which includes word classes and sub-classes and (3) domain-specific level. At phrase level, *"Order Ed Sheeran Albums"* consists of *Verb Phrase* and *Noun Phrases*, while at word level, it consists of *Verb (Action Verb)* and *Nouns (Proper Noun and Common Noun)*. At the domain specific level it consists of *Action Verb - Interact* (AV_I), *Proper Noun - Celebrity* (PN_C) and *Common Noun - Other - Plural* (CN_{OP}). On the contrary, at phrase level, *"Ed Sheeran Albums Order"* consists of *Noun Phrases*; at word level, it consists of *Nouns (Proper Noun and Common Nouns)*. At the domain-specific level it consists of *Proper Noun - Celebrity* (PN_C), *Common Noun - Other - Plural* (CN_{OP}) and *Common Noun - Other - singular* (CN_{OS}). The different syntactical structure of the two sentence leads to different syntactical patterns, which result in different meaning, intent and search results.

Phase 2: Parsing: This step is mainly responsible for extracting terms in the text to help generate the grammar structure in the next phase to facilitate the tagging of each word to the right term category. This is done by using the keywords and phrases that have been identified from the previous phase; first, compound words will be parsed and extracted, followed by single words.

Phase 3: Tagging: In this phase the text is transformed into a pattern of grammatical terms by mapping each term to its grammar terminals; each term will be mapped to its highest level of abstraction (word class, sub-class or domain-specific) and after mapping each terms the grammatical pattern is formulated. Using the domain-specific grammar that has been generated in Phase 1 (Grammar), terms will be tagged to their terminals.

Phase 4: Classification: In this phase the patterns generated in the tagging phase are used for machine learning; the aim of this phase is to build a model for automatic classification. The classification is done by following the standard process for machine learning, which involves the splitting of the dataset into a training dataset and a test dataset. The training dataset is used for building the model, and the test dataset is used to evaluate the performance of the model.

4 Experimental Study and Results

The objective of the experimental study is to investigate the ability of our proposed parsing and tagging approach to work on different domains. Two domains were used: classification of search queries and classification of questions (for question-answering systems). To assess the performance of the machine learning classifiers, the Weka[6] software [6] was used. The experiments were set up using the typical 10-fold cross validation and the effectiveness of the classification was evaluated based on Precision, Recall and F-Measure. The results are presented in the next sub-sections for the two domains.

4.1 Queries Classification

1953 labelled queries from [14] were used, and 4,047 queries were randomly selected from AOL 2006 dataset. Queries were classified and labelled to three different categories; these categories are based on Broder's [2] classification of web queries, which are informational, navigational and transactional.

4.1.1 Results

Table 1 presents the classification performance details (Precision, Recall and F-Measure) of the Support Vector Machine (SVM) and Naive Bayes (NB) classifiers for query classification. Results show that $DSSPT_{SVM}$ identified correctly (i.e. Recall) 99.6% of the questions, while $DSSPT_{NB}$ correctly classified 95.5% of the query. $DSSPT_{SVM}$ misclassified 0.5% of transactional queries as informational, while informational and navigational queries were 100% correctly classified. Furthermore, $DSSPT_{NB}$ incorrectly classified 4.5% of the queries – 3.4% of the informational queries were classified as transactional, and 8.5% of the transactional queries were classified as informational.

Table 1 Performance of the classifiers for Query Classification.

	$DSSPT_{SVM}$			$DSSPT_{NB}$		
Accuracy	99.6%			95.5%		
Precision	0.996			0.955		
Recall	0.996			0.955		
F-score	0.996			0.955		
Class:	P	R	F	P	R	F
Info.	0.997	0.998	0.998	0.955	0.966	0.96
Nav.	1.00	1.00	1.00	0.999	1.00	1.00
Trans.	0.972	0.955	0.964	0.935	0.915	0.925

4.2 Questions Classification

We used 1,160 questions that were randomly selected from Yahoo Non-Factoid Question Dataset[7], TREC 2007 Question Answering Data[8] and a Wikipedia dataset[9]. Questions were classified and labelled to six different categories, namely: causal, choice, confirmation (Yes-No Questions), factoid (Wh-Questions), hypothetical and list. These classifications were proposed by [16].

[6] http://www.cs.waikato.ac.nz/ml/weka/
[7] https://ciir.cs.umass.edu/downloads/nfL6/
[8] http://trec.nist.gov/data/qa/t2007_qadata.html
[9] https://www.cs.cmu.edu/~ark/QA-data

4.2.1 Results

Table 2 presents the classification performance details (Precision, Recall and F-Measure) of the SVM and NB classifiers for question classification. Results show that $DSSPT_{SVM}$ identified correctly (i.e. Recall) 88.6% of the questions, while $DSSPT_{NB}$ identified correctly 83.5% of the questions.

More specifically, looking at where the errors occur, when using $DSSPT_{SVM}$, 3.2% of the causal questions were misclassified as confirmation and 32.2% were misclassified as factoid. From the choice questions, 41.7% were misclassified as confirmation and 33.3% were misclassified as factoid. Similarly, 4% of the list questions were misclassified as confirmation and 45.5% were misclassified as factoid. These results indicate that $DSSPT_{SVM}$ could not distinguish between causal, choice and list types of questions and incorrectly classified most of them as confirmation and factoid questions. Moreover, 1.6% of confirmation questions were misclassified as factoid and less than 1% were misclassified as choice or list. For the factoid questions 4.6% were misclassified as list, 1.2% were misclassified as causal, 1% were misclassified as confirmation and less than 1% were misclassified as choice. In addition, most of the hypothetical questions, i.e. 57.1%, were misclassified as factoid.

The $DSSPT_{NB}$ classifier incorrectly classified 6.5% of the causal questions as confirmation, 80.6% as factoid and 3.2% as list. Similar to $DSSPT_{SVM}$ classifier, $DSSPT_{NB}$ could not identify choice questions and misclassified 41.7% as confirmation and 58.3% as factoid. Furthermore, 0.9% of the confirmation questions were misclassified as choice, 3.4% as factoid, 2% as hypothetical and 0.9% as list. For the factoid questions, 1.3% were misclassified as causal, 0.43% as choice, 2.5% as confirmation, 0.87% as hypothetical and 2.2% as list. Moreover, 14.3% of the hypothetical questions were misclassified as causal and 57.1% as factoid. For the list type of question $DSSPT_{NB}$ incorrectly classified 7% as confirmation and 65.3% as factoid.

Table 2 Performance of the classifiers for Question Classification.

	$DSSPT_{SVM}$			$DSSPT_{NB}$		
Accuracy:	88.6%			83.5%		
Precision:	0.88			0.814		
Recall:	0.886			0.835		
F-score:	0.881			0.818		
Class:	P	R	F	P	R	F
Causal	0.714	0.645	0.678	0.231	0.097	0.136
Choice	0.429	0.25	0.316	0.00	0.00	0.00
Conf.	0.948	0.972	0.96	0.906	0.928	0.917
Factoid	0.903	0.929	0.915	0.85	0.927	0.887
Hypo.	1.00	0.429	0.6	0.133	0.286	0.182
List	0.6	0.505	0.548	0.609	0.277	0.381

Unlike the previous approaches which focus only on the type of domain, our proposed Domain-Specific Syntax-based Parsing and Tagging (DSSPT) is a general approach for incorporating domain-specific tags, which exploits the structure of the text through using domain-specific grammatical categories and rules. Moreover, the domain-specific grammar could be easily integrated in different platforms. In addition, using syntactic categories related to different domain-specific types enable the machine learning algorithms to better differentiate between different queries/question types.

5 Conclusion and Future Work

In this paper, we proposed a domain specific syntax-based Parsing and tagging (DSSPT) approach. The grammatical rules contain in addition to typical categories of English grammar, domain-related grammatical categories. The results show that our solution led to a good performance when applying it on two different domains.

The proposed framework can be applied to other domains with similar classification problems, such as Twitter, which will be investigated in future work. In addition, we aim at examining and analyzing more datasets from different domains to enrich the tag-set which will extend the ability of our framework to be used in more domains.

References

1 Thorsten Brants. TnT: a statistical part-of-speech tagger. In *Proceedings of the sixth conference on Applied natural language processing*, pages 224–231. Association for Computational Linguistics, 2000.

2 Andrei Broder. A taxonomy of web search. In *ACM Sigir forum*, volume 36(2), pages 3–10. ACM, 2002.

3 Danqi Chen and Christopher Manning. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 740–750, 2014.

4 Jesús Giménez and Lluis Marquez. Fast and accurate part-of-speech tagging: The SVM approach revisited. *Recent Advances in Natural Language Processing III*, pages 153–162, 2004.

5 Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, and Mills. Part-of-speech tagging for twitter: Annotation, features, and experiments. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 42–47. Association for Computational Linguistics, 2011.

6 Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. The WEKA data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18, 2009.

7 Atsushi Keyaki and Jun Miyazaki. Part-of-speech tagging for web search queries using a large-scale web corpus. In *Proceedings of the Symposium on Applied Computing*, pages 931–937. ACM, 2017.

8 Terry Koo, Xavier Carreras Pérez, and Michael Collins. Simple semi-supervised dependency parsing. In *46th Annual Meeting of the Association for Computational Linguistics*, pages 595–603, 2008.

9 Jayant Krishnamurthy, Pradeep Dasigi, and Matt Gardner. Neural semantic parsing with type constraints for semi-structured tables. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1517–1527, 2017.

10 Yoong Keok Lee, Aria Haghighi, and Regina Barzilay. Simple type-level unsupervised POS tagging. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 853–861. Association for Computational Linguistics, 2010.

11 EE Luneva, PI Banokin, VS Zamyatina, and SV Ivantsov. Natural language text parsing for social network user sentiment analysis based on fuzzy sets. In *Mechanical Engineering, Automation and Control Systems (MEACS), 2015 International Conference on*, pages 1–5. IEEE, 2015.

12 Mehdi Manshadi and Xiao Li. Semantic tagging of web search queries. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint*

Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2, pages 861–869. Association for Computational Linguistics, 2009.

13 David McClosky, Eugene Charniak, and Mark Johnson. Automatic domain adaptation for parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 28–36. Association for Computational Linguistics, 2010.

14 Marcelo Mendoza and Juan Zamora. Identifying the intent of a user query using support vector machines. In *International Symposium on String Processing and Information Retrieval*, pages 131–142. Springer, 2009.

15 A. Mohasseb, M. Bader-El-Den, H. Liu, and M. Cocea. Domain specific syntax based approach for text classification in machine learning context. In *2017 International Conference on Machine Learning and Cybernetics (ICMLC)*, volume 2, pages 658–663. IEEE Systems, Man and Cybernetics, 2017.

16 Alaa Mohasseb, Mohamed Bader-El-Den, and Mihaela Cocea. Question categorization and classification using grammar based approach. *Information Processing & Management*, 2018.

17 Alaa Mohasseb, Mohamed Bader-El-Den, Andreas Kanavos, and Mihaela Cocea. Web Queries Classification Based on the Syntactical Patterns of Search Types. In *International Conference on Speech and Computer*, pages 809–819. Springer, 2017.

18 Alaa Mohasseb, Maged El-Sayed, and Khaled Mahar. Automated Identification of Web Queries using Search Type Patterns. In *WEBIST (2)*, pages 295–304, 2014.

19 Joakim Nivre, Johan Hall, and Jens Nilsson. Maltparser: A data-driven parser-generator for dependency parsing. In *Proceedings of LREC*, volume 6, pages 2216–2219, 2006.

20 Slav Petrov, Dipanjan Das, and Ryan McDonald. A universal part-of-speech tagset. *arXiv preprint arXiv:1104.2086*, 2011.

21 Yuval Pinter, Roi Reichart, and Idan Szpektor. Syntactic Parsing of Web Queries with Question Intent. In *HLT-NAACL*, pages 670–680, 2016.

22 Helmut Schmid. Probabilistic part-ofispeech tagging using decision trees. In *New methods in language processing*, page 154, 2013.

23 Noah A Smith, Michael Heilman, and Rebecca Hwa. Question generation as a competitive undergraduate course project. In *Proceedings of the NSF Workshop on the Question Generation Shared Task and Evaluation Challenge*, 2008.

24 Richard Socher, John Bauer, Christopher D Manning, et al. Parsing with compositional vector grammars. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 455–465, 2013.

25 Richard Socher, Christopher D Manning, and Andrew Y Ng. Learning continuous phrase representations and syntactic parsing with recursive neural networks. In *Proceedings of the NIPS-2010 Deep Learning and Unsupervised Feature Learning Workshop*, pages 1–9, 2010.

26 Pontus Stenetorp. Transition-based dependency parsing using recursive neural networks. In *NIPS Workshop on Deep Learning*. Citeseer, 2013.

27 Xiangyan Sun, Haixun Wang, Yanghua Xiao, and Zhongyuan Wang. Syntactic Parsing of Web Queries. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1787–1796, 2016.

28 Ivan Titov and James Henderson. Porting statistical parsers with data-defined kernels. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, pages 6–13. Association for Computational Linguistics, 2006.

29 Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics*

on Human Language Technology-Volume 1, pages 173–180. Association for Computational Linguistics, 2003.

30 Gökhan Tür, Minwoo Jeong, Ye-Yi Wang, Dilek Hakkani-Tür, and Larry P. Heck. Exploiting the semantic web for unsupervised natural language semantic parsing. In *INTERSPEECH 2012, 13th Annual Conference of the International Speech Communication Association, Portland, Oregon, USA, September 9-13, 2012*, pages 338–341. ISCA, 2012.

31 Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. Semantic Parsing via Staged Query Graph Generation: Question Answering with Knowledge Base. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 1321–1331, 2015.

32 Yue Zhang and Stephen Clark. A tale of two parsers: investigating and combining graph-based and transition-based dependency parsing using beam-search. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 562–571. Association for Computational Linguistics, 2008.

A Appendix: Grammar terms and corresponding abbreviations

Category Name	Abbreviation	Category Name	Abbreviation
Verbs	V	Action Verbs	AV
Action Verb-Interact terms	AV_I	Action Verb-Locate	AV_L
Action Verb- Download	AV_D	Auxiliary Verb	$AuxV$
Linking Verbs	LV	Adjective Free	Adj_F
Adjective Online	Adj_O	Adjective	Adj
Adverb	Adv	Determiner	D
Conjunction	$Conj$	Preposition	P
Domain Suffix	DS	Domain Prefix	DP
Noun	N	Pronoun	$Pron$
Numeral Numbers	NN	Ordinal Numbers	NN_O
Cardinal Numbers	NN_C	Proper Nouns	PN
Celebrities Name	PN_C	Entertainment	PN_{Ent}
Newspapers, Magazines, Documents, Books	PN_{BDN}	Events	PN_E
Companies Name	PN_{CO}	Geographical Areas	PN_G
Places and Buildings	PN_{PB}	Institutions, Associations, Clubs, Parties, Foundations and Organizations	PN_{IOG}
Brand Names	PN_{BN}	Software and Applications	PN_{SA}
Products	PN_P	History and News	PN_{HN}
Religious Terms	PN_R	Holidays, Days, Months	PN_{HMD}
Health Terms	PN_{HLT}	Science Terms	PN_S
Common Noun	CN	Common Noun – Other- Singular	CN_{OS}
Common Noun- Other- Plural	CN_{OP}	Database and Servers	CN_{DBS}
Advice	CN_A	Download	CN_D
Entertainment	CN_{Ent}	File Type	CN_{File}
Informational Terms	CN_{IFT}	Obtain Offline	CN_{OF}
Obtain Online	CN_{OO}	History and News	CN_{HN}
Interact terms	CN_I	Locate	CN_L
Site, Website, URL	CN_{SWU}	Question Words	QW
How	QW_{How}	What	QW_{What}
When	QW_{When}	Where	QW_{Where}
Who	QW_{Who}	Which	QW_{Which}

Evaluation of Rule-Based Learning and Feature Selection Approaches For Classification

Fatima Chiroma
School of Computing, University of Portsmouth, United Kingdom
fatima.chiroma@port.ac.uk

Mihaela Cocea
School of Computing, University of Portsmouth, United Kingdom
mihaela.cocea@port.ac.uk

Han Liu
School of Computer Science and Informatics, Cardiff University, United Kingdom
LiuH48@cardiff.ac.uk

Abstract

Feature selection is typically employed before or in conjunction with classification algorithms to reduce the feature dimensionality and improve the classification performance, as well as reduce processing time. While particular approaches have been developed for feature selection, such as filter and wrapper approaches, some algorithms perform feature selection through their learning strategy. In this paper, we are investigating the effect of the implicit feature selection of the PRISM algorithm, which is rule-based, when compared with the wrapper feature selection approach employing four popular algorithms: decision trees, naïve bayes, k-nearest neighbors and support vector machine. Moreover, we investigate the performance of the algorithms on target classes, i.e. where the aim is to identify one or more phenomena and distinguish them from their absence (i.e. non-target classes), such as when identifying benign and malign cancer (two target classes) vs. non-cancer (the non-target class).

2012 ACM Subject Classification Computing methodologies → Feature selection

Keywords and phrases Feature Selection, Prism, Rule-based Learning, Wrapper Approach

Digital Object Identifier 10.4230/OASIcs.ICCSW.2018.6

Category Main Track

1 Introduction

The application of machine learning has been on the rise in recent years [10] as its various techniques have been applied to different problem domains successfully. For example, in medicine, machine learning techniques have been used to predict the effectiveness of drugs in patients with depression [16] while in finance it was used to detect fraudulent activities on credit cards [16], to mention a few. Furthermore, approaches used by machine learning techniques differ. For example, rule-based learning is a machine learning technique that makes its decisions based on a number of rules [15] - a popular rule-based algorithm is Prism [8, 5, 18, 3], which works with the concept of target class and is capable of selecting attributes based on their importance to a particular class.

Another machine learning technique is Feature selection. Its strategy is to select only the attributes that are relevant and effective from a large number of features or attributes in a data-set where the selected attribute determines the performance of the classification [6, 12]. These approaches will be explored in this study, especially their performance when applied

to classification problems; more specifically, we will investigate the performance of Prism, which has implicit feature selection, in comparison with other feature selection approaches.

This paper is organized as follows: Section 1 introduces the background of the study; section 2 reviews the related works that have been carried out by various researchers; Section 3 discusses the experimental approach; Section 4 comprises of the results and discussion; and Section 5 concludes the study and presents the future work.

2 Related Work

Classification is one of the most popular machine learning tasks which typically involves the training of an algorithm to build a model which is subsequently used to identify the category of an unseen instance [8]. Various research relating to classification has been carried out using several approaches. Rule-based learning and Feature selection are some of the approaches that have been applied to classification problems.

Rule-based learning is an approach in which the model consists of a set of rules which were learned from the data [15]. For example, Prism, which is a rule learning algorithm learns from a set of rules that separates a specific class i.e. the target class from other classes [8, 13]. Prism has been used by several researchers in classification problems. For example, [5] who developed Prism, used it to identify types of contact lenses and the results have shown that Prism has a higher classification accuracy than ID3, a decision tree learning algorithm. Another study showed a 92% classification accuracy was achieved by Prism on image segmentation data set for multi-task feature selection [13].

Feature selection can be done by following one of three approaches, i.e. filter, wrapper or embedded approaches [17].The filter method does not require the application of a classification algorithm to evaluate the quality of the features selected while the wrapper method is the opposite [14], i.e. it is dependent on the classification algorithm to evaluate the quality of selected features. The embedded method, on the other hand, performs its feature selection as the optimal parameters are being learned [14].

A study was done by [17], where they compared the Naïve Bayes wrapper feature selection with other filter feature selection algorithms on Human Activity Recognition machine learning problem. The result of their study showed that the wrapper method outperformed all the filter algorithms and they were also able to discover that features selected by the wrapper method are efficiently usable with other machine learning algorithms.

The research that is most related to this study is the research done by [15]. They used feature selection with wrapper approach based on ensemble learning on 13 data-sets using two base learners: Decision Tree and Naïve Bayes. They were able to identify which wrapper approach has better classification accuracy and their results showed that the forward selection when applied to Decision Tree had the highest accuracy in their study.

Therefore, the aim of this study is to explore how the implicit feature selection within Prism compares with the wrapper feature selection approach.

3 Data and Experimental Setup

The experiment was carried out using seven classification data-sets which were acquired through the UCI Machine Learning repository [9] and Knowledge Extraction based on Evolutionary Learning (KEEL) data-set repository [4].

Table 1 lists the data-sets used, as well as their properties, i.e. the number of instances, the number of attributes, the type of attributes and the number of classes. We chose data-sets

Table 1 Data-sets Description.

S/N	Name	Instances	Attributes	Type	Classes
1	Balance Scale	625	4	Integer	3
2	Breast Tissue	106	10	Real	4
3	Forest Type	198	27	Integer, Real	4
4	Heart Disease Cleveland	303	75	Integer, Real	5
5	Lymphography	148	18	Integer	4
6	Soybean	47	35	Integer	4
7	Website Phishing	1353	9	Integer	3

with at least 3 classes, as we focused our investigation on non-binary data-sets, where one or more target classes need to be distinguished from one or more non-target classes.

All data-sets in Table 1 are classification data-sets with numeric data, that have already been pre-processed before acquisition. However, due to the importance of pre-processing for classification [8] additional pre-processing was done to ensure the data is clean, compatible and ready for classification. These pre-processing includes the conversion of the label or class attribute to string, filtration of attributes that are not relevant for the study, renaming of important attribute names for better readability and easier identification, and also the concatenation of data-sets with multiple files.

These data-sets were classified using five machine learning algorithms: Prism, Decision Tree (DT), Naïve Bayes (NB), Library of Support Vector Machine (LibSVM) and K-Nearest Neighbors (KNN). Prism is the target-classifier for this experiment and the remaining four are subsequently going to be referred to as the other-classifiers.

Furthermore, the forward selection and backward elimination algorithms which are based on the wrapper feature selection method, were applied to the data-sets using the other-classifiers. The reason for using both the forward selection and the backward elimination algorithm is due to the fact that the forward selection algorithm is known to improve accuracy but only on some data-sets as it may not have any effect on others [10], while the backward elimination allows for backtracking when it removes features therefore allowing for the inclusion of previously eliminated features [15, 2].

For the evaluation, the 10-fold cross validation was applied to both the target-classifier and other-classifiers. This validation technique was applied due to its ability to limit the level of influence of randomly selected training sets on the overall results [8].

4 Results and Discussion

The results of the experiment have been presented in three tables for better comparison across the machine classifiers and data-sets. Thus, we present the results across all classes (Table 2) and across the target classes (Table 3), as well as the number of features selected (Table 4). In terms of the performance of the algorithms, we report the F-measure (which is the harmonic mean of precision [1] and recall [2]) rather than accuracy, as it is less influenced by an unbalanced distribution of instances across classes.

Table 2 shows the performance of the machine classifiers for each data-set.

[1] Precision is the number of correctly identified instances from all instances
[2] Recall is the number of correctly identified instances from the subset of relevant instances

Table 2 All Classes F-Measure Results.

Data-sets	Prism	DT		NB		KNN		LibSVM	
		FS	*BE*	*FS*	*BE*	*FS*	*BE*	*FS*	*BE*
Balance Scale	0.62	0.59	0.44	0.63	0.43	0.62	0.37	0.86	0.43
Breast Tissue	0.82	1.00	1.00	1.00	1.00	0.78	0.73	0.78	0.73
Forest Type	0.88	0.95	0.73	0.93	0.55	0.94	0.54	0.98	0.99
Heart Disease Cleveland	0.46	0.28	0.22	0.31	0.14	0.23	0.14	0.30	0.17
Lymphography	0.43	0.47	0.36	0.45	0.36	0.49	0.18	0.59	0.36
Soybean	0.98	0.98	0.84	0.94	0.84	1.00	0.67	1.00	0.73
Website Phishing	0.88	0.87	0.56	0.65	0.56	0.81	0.23	0.61	0.55

Table 3 Target Class F-Measure Results.

Data-sets	Prism	DT		NB		KNN		LibSVM	
		FS	*BE*	*FS*	*BE*	*FS*	*BE*	*FS*	*BE*
Balance Scale	0.93	0.87	0.66	0.95	0.64	0.92	0.56	0.95	0.675
Breast Tissue	0.83	1.00	1.00	1.00	1.00	0.72	0.44	0.67	0.39
Forest Type	0.89	0.95	0.74	0.94	0.47	0.96	0.48	0.98	0.69
Heart Disease Cleveland	0.37	0.17	0.09	0.18	0	0.12	0	0.17	0.03
Lymphography	0.58	0.62	0.62	0.67	0.48	0.65	0.65	0.79	0.79
Soybean	1.00	1.00	0.79	0.97	0.79	1.00	0.83	1.00	0.64
Website Phishing	0.86	0.86	0.41	0.53	0.41	0.76	0	0.47	0.40

The results show that Prism has the highest performance on two of the data-sets: Website Phishing and Heart Disease Cleveland with an F-measure of 0.88 and 0.46, respectively. Moreover, for the Heart Disease Cleveland data-set which has the highest number of attributes (75) and classes (5), we notice that Prism outperforms all other wrapper approaches by a very high margin.

For the other 5 data-sets, the results show that: (a) LibSVM is best on three of the data-sets; (b) DT and NB are equally best on one data-set; (c) KNN and LibSVM are equally best on one data-set.

The results also show that on the used data-sets the forward selection performance is higher than the performance of the backward elimination approach – this is likely to be due to the simplicity of forward selection and the ability to add only feature with the highest performance[15].

Table 3 shows the performance results for the target classes i.e. the F-measure for only the target classes. It shows that Prism also has the highest performance for the Heart Disease Cleveland data-set with an F-measure of 0.37 but equal performance with Decision Tree for the Website Phishing with an F-measure of 0.86 as well as the soybean data-set which has an F-measure of 1.0 for Prism, Decision Tree, K-Nearest Neighbor and LibSVM. The 1.0 performance on the soybean data-sets obtained by the wrapper approaches was achieved using the forward selection algorithm.

For the other four data-sets, we observe the following: (a) LibSVM and NB are equally best on one data-set; (b) DT and NB are equally best on one data-set; (c) LibSVM is best for two data-sets.

For classification, attributes can be redundant, irrelevant or problematic [15]. Therefore, applying feature selection approaches ensures the selection of attributes that are relevant

Table 4 Number of Attributes.

Data-sets	Total Attributes	Prism	DT		NB		KNN		LibSVM	
			FS	*BE*	*FS*	*BE*	*FS*	*BE*	*FS*	*BE*
Balance Scale	4	4	4	4	4	4	4	4	4	4
Breast Tissue	10	9	2	2	2	2	7	4	2	4
Forest Type	27	22	4	4	10	11	8	6	17	9
Heart Disease Cleveland	75	13	4	3	4	3	4	9	4	7
Lymphography	18	17	4	8	10	10	16	11	11	15
Soybean	35	4	2	2	2	2	2	2	2	2
Website Phishing	9	9	6	7	4	8	9	9	9	5

or important. Table 4 shows the total number of attributes selected to achieve the highest performance for each classifier. These selected attributes are considered to be the most relevant for the classification; however, these may vary across the different approaches.

Prism used the most number of attributes across all data-sets when compared with the wrapper approaches. This seems to be an advantage in some situations, e.g. on the Heart Disease Cleveland and the Website Phishing data-sets, but not in others.

Thus, Prism performed better with data-sets that have large instances or high number of attributes.

Furthermore, the other-classifiers performed better with the forward selection algorithm than the backward elimination. Also, Prism had higher performance than all the classifiers for the backward elimination algorithm, except for LibSVM for the target classes, which has an F-measure of 0.98 for the Lymphography data-sets.

Additionally, according to [15] one of the benefits of feature selection is the reduction of run-time for large and multidimensional data-sets as well as increased accuracy. However, on the used data-sets, LibSVM which is a library of Support Vector Machine [7], had the longest processing time.

5 Conclusion and Future Work

In this paper, we explored how the implicit feature selection within prism compares with the wrapper feature selection approach using four popular machine learning algorithms: Decision Tree, Naïve Bayes, LibSVM and K-Nearest Neighbour. The results of the experiments have shown that both Prism and the other-classifiers have varying performance. Therefore, we will further extend this study by exploring the same approach and algorithms on text data-sets to measure its performance for text classification. We will also further investigate what properties of data make Prism more suitable for some classification problems than others.

Acknowledgements The authors would like to extend their gratitude to the Petroleum Technology Development Fund for their support. Additionally, some of the data-sets used in this study: the lymphography data-set was obtained by M. Zwitter and M. Soklic from the University Medical Centre, Institute of Oncology, Ljubljana, Yugoslavia; the website phishing data-set was compiled by [1]; the forest type by [11] and the Heart Disease Cleveland data-set was provided by Robert Detrano, M.D., Ph.D. from the Cleveland Clinic Foundation.

References

1 Neda Abdelhamid, Aladdin Ayesh, and Fadi Thabtah. Phishing detection based Associative Classification data mining. *Expert Systems with Applications*, 41(13):5948–5959, 2014.

2 Shigeo Abe. Modified backward feature selection by cross validation. In *ESANN*, pages 163–168, 2005.

3 Maher Aburrous, M Alamgir Hossain, Keshav Dahal, and Fadi Thabtah. Intelligent phishing detection system for e-banking using fuzzy data mining. *Expert systems with applications*, 37(12):7913–7921, 2010.

4 Jesús Alcalá-Fdez, Alberto Fernández, Julián Luengo, Joaquín Derrac, Salvador García, Luciano Sánchez, and Francisco Herrera. Keel data-mining software tool: data set repository, integration of algorithms and experimental analysis framework. *Journal of Multiple-Valued Logic & Soft Computing*, 17, 2011.

5 Jadzia Cendrowska. PRISM: An algorithm for inducing modular rules. *International Journal of Man-Machine Studies*, 27(4):349–370, 1987. doi:10.1016/S0020-7373(87)80003-2.

6 Girish Chandrashekar and Ferat Sahin. A survey on feature selection methods. *Computers & Electrical Engineering*, 40(1):16–28, 2014.

7 Chih-Chung Chang and Chih-Jen Lin. LIBSVM: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):27, 2011.

8 Fatima Chiroma, Han Liu, and Mihaela Cocea. Suicide Related Text Classification with Prism Algorithm. *International Conference on Machine Learning and Cybernetics (ICMLC)*, pages 1–6, 2018.

9 Dua Dheeru and Efi Karra Taniskidou. UCI machine learning repository, 2017. URL: http://archive.ics.uci.edu/ml.

10 Pedro Domingos. A few useful things to know about machine learning. *Communications of the ACM*, 55(10):78–87, 2012.

11 Brian Johnson, Ryutaro Tateishi, and Zhixiao Xie. Using geographically weighted variables for image classification. *Remote sensing letters*, 3(6):491–499, 2012.

12 Vipin Kumar and Sonajharia Minz. Feature selection. *SmartCR*, 4(3):211–229, 2014.

13 Han Liu, Mihaela Cocea, and Weili Ding. Multi-task learning for intelligent data processing in granular computing context. *Granular Computing*, 3(3):257–273, 2017.

14 Mehdi Naseriparsa, Amir-Masoud Bidgoli, and Touraj Varaee. A hybrid feature selection method to improve performance of a group of classification algorithms. *International Journal of Computer Applications*, 69(17):28–35, 2013.

15 Rattanawadee Panthong and Anongnart Srivihok. Wrapper feature subset selection for dimension reduction based on ensemble learning algorithm. *Procedia Computer Science*, 72:162–169, 2015.

16 The Royal Society. Machine learning: the power and promise of computers that learn by example. Online, April 2017. URL: https://royalsociety.org/~/media/policy/projects/machine-learning/publications/machine-learning-report.pdf.

17 Jozsef Suto, Stefan Oniga, and Petrica Pop Sitar. Comparison of wrapper and filter feature selection algorithms on human activity recognition. In *Computers Communications and Control (ICCCC), 2016 6th International Conference on*, pages 124–129. IEEE, 2016.

18 Ian H Witten, Eibe Frank, Mark A Hall, and Christopher J Pal. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.

The iBUG Eye Segmentation Dataset

Bingnan Luo
Intellignet Behaviour Understanding Group, Imperial College London, United Kingdom
bingnan.luo16@imperial.ac.uk

Jie Shen
Intellignet Behaviour Understanding Group, Imperial College London, United Kingdom
jie.shen07@imeprail.ac.uk

Yujiang Wang
Intellignet Behaviour Understanding Group, Imperial College London, United Kingdom
yujiang.wang14@imeprail.ac.uk

Maja Pantic
Intellignet Behaviour Understanding Group, Imperial College London, United Kingdom
m.pantic@imeprail.ac.uk

Abstract

This paper presents the first dataset for eye segmentation in low resolution images. Although eye segmentation has long been a vital preprocessing step in biometric applications, this work is the first to focus on low resolutions image that can be expected from a consumer-grade camera under conventional human-computer interaction and/or video-chat scenarios. Existing eye datasets have multiple limitations, including: (a) datasets only contain high resolution images; (b) datasets did not include enough pose variations; (c) a utility landmark ground truth did not be provided; (d) high accurate pixel-level ground truths had not be given. Our dataset meets all the above conditions and requirements for different segmentation methods. Besides, a baseline experiment has been performed on our dataset to evaluate the performances of landmark models (Active Appearance Model, Ensemble Regression Tree and Supervised Descent Method) and deep semantic segmentation models (Atrous convolutional neural network with conditional random field). Since the novelty of our dataset is to segment the iris and the sclera areas, we evaluate above models on sclera and iris only respectively in order to indicate the feasibility on eye-partial segmentation tasks. In conclusion, based on our dataset, deep segmentation methods performed better in terms of IOU-based ROC curves and it showed potential abilities on low-resolution eye segmentation task.

2012 ACM Subject Classification Computing methodologies → Image segmentation

Keywords and phrases dataset, eye, segmentation, landmark, pixel-level

Digital Object Identifier 10.4230/OASIcs.ICCSW.2018.7

Category Main Track

1 Introduction

Eyes not only are the most vital sensory organ but also play a crucial role in conveying a person's emotional state and mental wellbeing [5]. Although there have been numerous works on blink detection [1, 8, 10], we argue that accurate segmentation of sclera and iris can provide much more information than blinks alone, thus allowing us to study the finer details of eye movement such as cascade, fixation and other gaze patterns. As a pre-processing step in iris recognition, iris segmentation in high resolution expression – less frontal face

images have been well studied by the biometric community. However, the commonly used Hough-transform-based method [14] does not work well on low-resolution images captured under normal human-computer interaction (HCI) and/or video-chat scenarios, when the boundary of eyes and iris are blurry and the shape of the eye can differ greatly due to pose variation and facial expression. To our knowledge, this work presents the first effort in solving the eye segmentation problem under such challenging conditions.

To investigate the topic of eye segmentation in low-resolution images, the first problem we need to address is the lack of data. Albeit both biometric community and facial analysis community published an abundance of datasets over the years, none can be used as is for our purpose because the former category only contains high resolution eye scans while the latter category lacks annotation of segmentation masks for sclera and iris. Therefore, during the course this work, we created a sizable eye segmentation dataset by manually annotating images selected from HELEN[9], 300VW[12], CVL[11] and Columbia Gaze[13] datasets.

After establish our dataset, it is necessary to evaluate how good performances are based on two types of ground truths. Therefore, deformable and deep segmentation models were chosen. Active Appearance Model(AAM)[3], Ensemble Regression Tree(ERT)[7] and Supervised Descent Method(SDM)[15] were compared with deep semantic segmentation methods(DeepLab[2] proposed): Atrous Neural Network with Conditional Random Field (ACNN+CRF). For deformable models, the segmentation of non-frontal faces is a big challenge because of occlusion, shape deformation and initializations. Therefore, the deep segmentation methods can relatively compensate this shortcoming, whose performance can be more stable especially on non-frontal faces. Otherwise, since we also want to know performances on iris-only and sclera-only segmentations, all models were utilized and trained by iris-background and sclera-background data samples. Finally, all segmentation results are evaluated and discussed based on Interaction over Union(IOU) and Receiver operating characteristic(ROC) curves. Based on that, the model with the best performance will be considered as the potential model in eye segmentation researches.

2 Relative Works

Image segmentation is one of the oldest computer vision problems studied by the community. Early approaches often rely on finding edges and/or specific shapes in the image or hand-craft feature maps. Albeit dated, this kind of simple methods are still being used for eye segmentation as a pre-processing step for iris recognition [14]. In this case, the eye and iris are modeled respectively, by two parabolic curves and an ellipse. The parameters of the curves are then determined using Hough transform performed on the image's edge map. Because this approach is sensitive to image noise and even slight shape variation caused by head-pose and/or facial expressions, it cannot be use for eye segmentation in images captured by consumer-grade camera under normal HCI/video-chat conditions.

On the other hand, various sparse 2D deformable-model-based methods, such as AAM, SDM or ERT, have shown promising results in image segmentation, and in particular, facial landmark localization. These methods work by finding a local minimum in the parametric space that can optimally describe the object's shapes and appearance. Since image intensity space contains multiple local minimums so that it is uneasy to control which local minimum it lies on. Moreover, the optimizing process of deformable models starts with the mean shape of object's intensity and geometry, thus the transformation of the landmark depends on the initialization and integrity of objects. Therefore, they share many common limitations, including sensitivities to initialization, occlusion, and out-of-plane rotation. In [3, 15, 7], the profile face landmark tracking is still a challenge. Thus deformable models can experienced as a candidate in our research, but the performance can be expectedly poor to profile faces.

Table 1 dataset statistical information.

Name	Value
Total number	3161
Non-frontal faces proportion	18.35%
low-resolution image proportion	66.97%
Number of bad illumination samples	10
Number of samples with glasses	185

The methods mentioned above rely on the prior knowledges (such as the number of point, landmark shapes or curves expressions) so that they cannot adapt variant images and head poses of eyes (profile faces or occlusions). To lighten the influences in the wild (illuminations and etc.) and adapt to multiple situations (tricky head poses or occlusions), more recently, various deep learning techniques have achieved impressive results in semantic segmentation of natural images, which is widely-used because of its better adaptability of performances in variant environments. In particular, DeepLab[2, 4] uses atrous convolutional neural network based on VGG-16 and ResNet-101 architectures to generate segmentation masked, refined by a fully-connected CRF layer with mean-field approximation for fast inference. The atrous convolutional neural network is the innovation of DeepLab in order to increase the ability of extracting global image features. They not only reduced the computational cost but also improved the generalization.

In previous works, there is no existed low-resolution eye dataset. However, facial datasets provide us a good sources to obtain eye images in different illuminations and pose variations. HELEN dataset[9] contains high resolution facial images in different situations (multi-faces, indoor/outdoor and etc.). 300VW[12] is a low-resolution facial video dataset captured in the wild. CVL[11] and Columbia Gaze[13] are two facial dataset technically captured in lab environment. Although these datasets cannot be utilized directly in our research, they can also regard as sources of our proposed dataset.

3 Data Description

We create the iBUG eye segmentation dataset by annotating a total of 3161 images selected from HELEN[9], 300VW[12], CVL[11] and Columbia Gaze[13] datasets. The dataset contains faces under various poses. Specifically, faces who look ahead and with slight rotations are frontal, the others are annotated sas non-frontal faces. We primarily focus on low resolution images, but a small number of high resolution images are also included for completeness. The distribution of eye-patch height is illustrated in Figure 1. Note that we use eye-patch height as a measure for image resolution because the widely-used interocular distance measure can be easily biased by face yaw. Last but not least, the dataset contains a small number of examples featuring partial occlusion and bad illumination. The detailed statistics can be found in Table 1.

Some examples of the annotated images are shown in Figure 2. The first row shows the source image and the location of the control points, while the second row visualizes the segmentation mask. Some extra statistical information has been presented in Table 1.

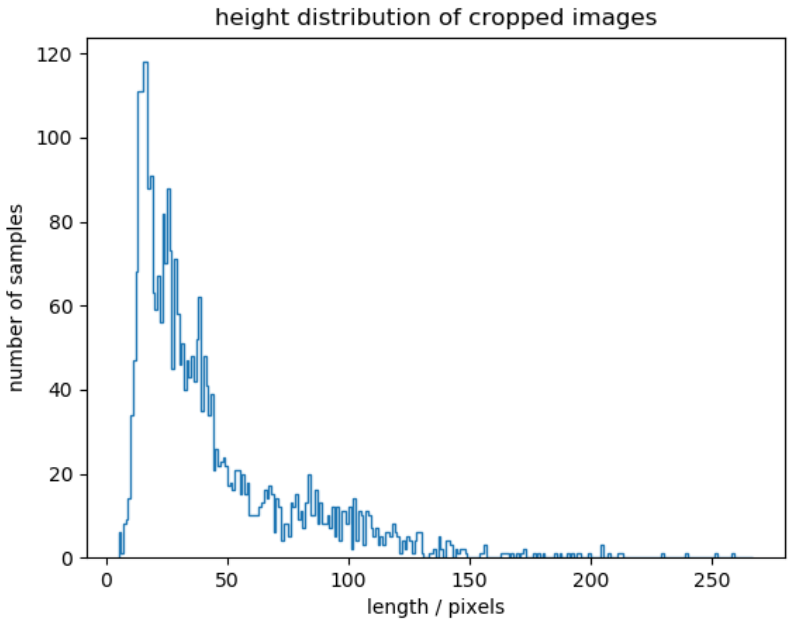


Figure 1 Height distribution of eye regions.

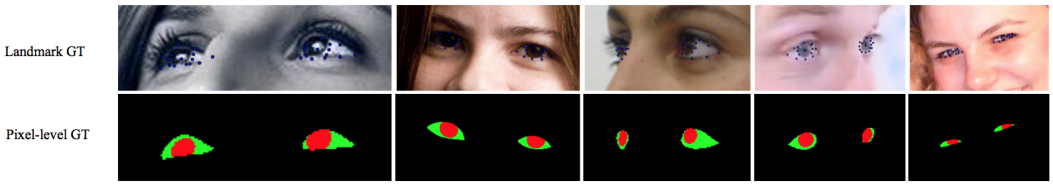


Figure 2 Some annotated images in our dataset.

4 Baseline Methods

In this work, we utilized deformable model-based methods (AAM, ERT and SDM) and DeepLab proposed Atrous CNN+CRF (CGG16+CRF and ResNet101+CRF) as the baseline method.

4.1 Landmark Models

Before we discuss details of utilizations of baseline methods, we are going to introduce some concepts about them. AAM, ERT and SDM are deformable statistic models which were generally used in object localization and alignment. The shape and texture will be transformed with a specific transformation function. Assume $x = \{x_1, x_2, ..., x_i\}$ indicates shapes of images and $g = \{g_1, g_2, ..., g_i\}$ presents textures of images. $\mathbf{x} = \bar{\mathbf{x}} + \mathbf{Q_s c_s}$ and $\mathbf{g} = \bar{\mathbf{g}} + \mathbf{Q_g c_g}$ demonstrate the transformation methods in deformable model-based methods. Q_s and Q_g are matrices describing the modes of variation derived from the training set. c_s is the shape model parameter and c_g is the appearance model parameter, which control the shape and texture gradient and the transforming directions. Thus, the aim of deformable model-based methods is to find the optimized local minimum between current image and the mean shape.

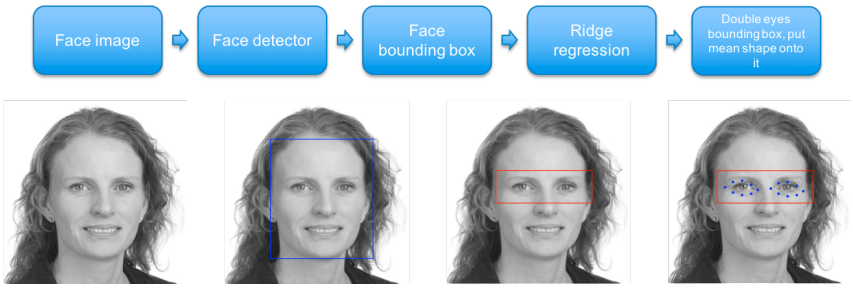


Figure 3 eye initialization generation.

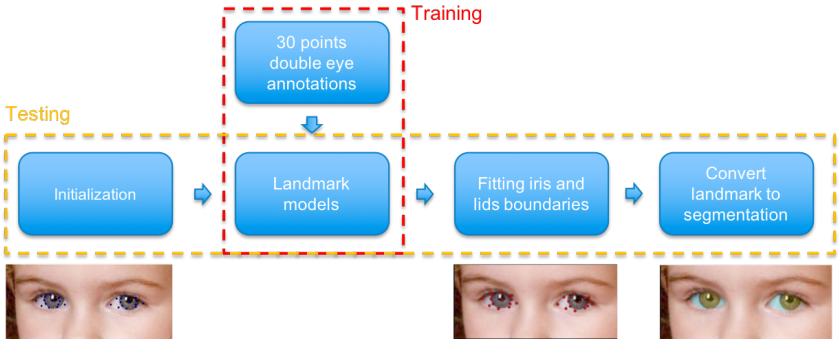


Figure 4 Flowchart of landmark models methodology.

Since deformable model-based methods are sensitive to initialization, the accuracy of initializations affects algorithms' performances. To generate an appropriate initialization, the first step is eye localization. The procedure is shown in Figure 3. Firstly, face detection method (fast RCNN[6]) was utilized to obtain the bounding box of each faces. Secondly, a ridge regression model was trained to predict eyes' bounding boxes based on facial bounding boxes. Finally, landmark models can use mean shape as initialization lying into above located bounding boxes. The procedure flowchart of deformable model-based method is shown in Figure 4.

4.2 Deep Segmentation Method

Atrous convolutional neural network (ACNN) effectively enlarge the field of view of filters without increasing the number of parameters or the amount of computation[2]. The method adds 'holes' to the convolution filter mask to better model the relationship between distant pixels. With our dataset, we fine-tune the ACNN network to produce the initial per-pixel probability score map from the input eye region RGB image. Note that at this stage, the entire eye region (which contains both left and right eyes) are fed to the ACNN. Because the shape and orientation of left and right eyes are highly correlated, the correlation can be learned and exploited by ACNN to produce good score map for both eyes when face is not in a frontal position so that one eye is more blurry/smaller than the other. Since the boundaries of the sclera and the iris were too blur to accurately be segmented, a CRF was utilized at the end of ACNN as post-processing in order to sharp boundaries. The procedure of the experiment is as Figure 5.

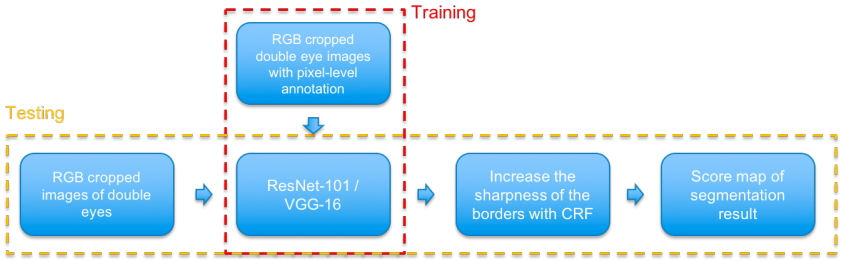
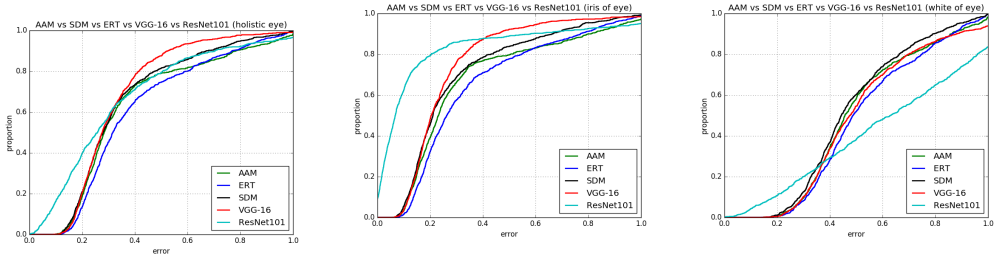


Figure 5 Flowchart of deep model methodology.



(a) Holistic eye. **(b)** Iris of eye. **(c)** Sclera of eye.

Figure 6 Evaluation of all models.

5 Experiments

In this section, we evaluated deformable model-based methods and deep models according to two criteria: (a) the performance of holistic, iris-only and sclera-only segmentation; (b) robustness through comparing performances of frontal faces and non-frontal faces. Performances of landmark and deep models are evaluated in Figure 6. It is obvious that performances of deep models on holistic eye and iris-only segmentation were better than deformable models. For VGG-16 and ResNet101, ResNet101 performed better than VGG16, since ResNet101 has larger size of architecture in order to gain wispy features of the eye. On the other hand, in sclera-only segmentation, the performance of ResNet101 was relatively worse than other methods, because the dataset we built was not big enough for large-scale ResNet101. Meanwhile, the overfitting was happened during ResNet101 training.

According to Figure 7 in appendix, the segmentation of profile faces is worse than frontal faces. Even so, deep segmentation models still got higher performances than deformable models on profile faces. On another aspect, the accuracy reduction of deep models is milder than deformable methods, which means that deep models are more robust than deformable methods under pose variation. Theoretically, for non-frontal faces, the face shape and texture need to be transformed further than frontal faces during predictions of deformable methods, so that it is difficult to find optimized local minimum from image intensity space. With inaccuracy initializations, non-frontal faces are still challenges for landmark tracking. On the other hand, the baseline methods we utilized in this research aims to evaluate the availability of our dataset, meanwhile, the performance of experiments provide a preliminary research on low-resolution eye segmentation. Although methods above are widely-used and may not be state-of-art currently, it is enough for us to present the effectiveness of our dataset. Therefore, this research indicates: (a) eye segmentation research can reasonably work on our dataset; (b) deep models are more potential for eye segmentation compared with deformable model-based methods.

6 Conclusion

In conclusion, there are two contributions in this research. Firstly, we proposed a new dataset for low-resolution eye segmentation. Our dataset provides two types of ground truths: 30-point landmarks and pixel-level ground truth. In terms of contents, the dataset contains frontal and non-frontal faces in low resolution of eye region, under variant illuminations and with/without glasses. Secondly, in order to evaluate the usability of our dataset and provide a preliminary eye segmentation investigation on low-resolution eye segmentation, we applied deformable model-based methods (AAM, SDM and ERT) and deep semantic segmentation models (VGG16+CRF and ResNet101+CRF) as baseline methods. According to the ROC curves of IOU accuracy, deep models got a better robustness than deformable methods. Moreover, especially for non-frontal faces, performances of deep models can adapt head poses variation. Otherwise, our dataset can be utilized for iris-only and sclera-only segmentation. Based on experiments, deep models got better performances on our dataset as well. Therefore, this research indicates that researchers can put more efforts to use deep segmentation methods instead of deformable model-based methods in eye segmentation task. Otherwise, existed models did not consider the shape refinement and shape prior of the eye, thus in future researches plugging in shape prior and post-processing shape model can extremely improve segmentation performance.

References

1. Michael Chau and Margrit Betke. Real time eye tracking and blink detection with usb cameras. Technical report, Boston University Computer Science Department, 2005.
2. Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2018.
3. Timothy F Cootes, Gareth J Edwards, and Christopher J Taylor. Active appearance models. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, pages 681–685, 2001.
4. Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.
5. Bruce M Hood, J Douglas Willen, and Jon Driver. Adult's eyes trigger shifts of visual attention in human infants. *Psychological Science*, 9(2):131–134, 1998.
6. Huaizu Jiang and Erik Learned-Miller. Face detection with the faster R-CNN. In *Automatic Face & Gesture Recognition (FG 2017), 2017 12th IEEE International Conference on*, pages 650–657. IEEE, 2017.
7. Vahid Kazemi and Josephine Sullivan. One millisecond face alignment with an ensemble of regression trees. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1867–1874, 2014.
8. Marc Lalonde, David Byrns, Langis Gagnon, Normand Teasdale, and Denis Laurendeau. Real-time eye blink detection with GPU-based SIFT tracking. In *Computer and Robot Vision, 2007. CRV'07. Fourth Canadian Conference on*, pages 481–487. IEEE, 2007.
9. Vuong Le, Jonathan Brandt, Zhe Lin, Lubomir Bourdev, and Thomas S Huang. Interactive facial feature localization. In *European conference on computer vision*, pages 679–692. Springer, 2012.
10. Yuezun Li, Ming-Ching Chang, Hany Farid, and Siwei Lyu. In Ictu Oculi: Exposing AI Generated Fake Face Videos by Detecting Eye Blinking. *arXiv preprint arXiv:1806.02877*, 2018.

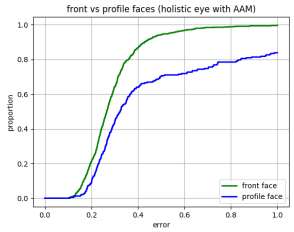
11 Peter Peer. Cvl face database. *Computer vision lab., faculty of computer and information science, University of Ljubljana, Slovenia. Available at http://www. lrv. fri. uni-lj. si/facedb. html*, 2005.

12 Jie Shen, Stefanos Zafeiriou, Grigoris G Chrysos, Jean Kossaifi, Georgios Tzimiropoulos, and Maja Pantic. The first facial landmark tracking in-the-wild challenge: Benchmark and results. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 50–58, 2015.

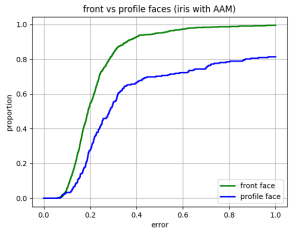
13 Brian A Smith, Qi Yin, Steven K Feiner, and Shree K Nayar. Gaze locking: passive eye contact detection for human-object interaction. In *Proceedings of the 26th annual ACM symposium on User interface software and technology*, pages 271–280. ACM, 2013.

14 Qi-Chuan Tian, Quan Pan, Yong-Mei Cheng, and Quan-Xue Gao. Fast algorithm and application of hough transform in iris segmentation. In *Machine Learning and Cybernetics, 2004. Proceedings of 2004 International Conference on*, volume 7, pages 3977–3980. IEEE, 2004.

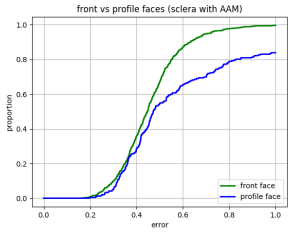
15 Xuehan Xiong and Fernando De la Torre. Supervised descent method and its applications to face alignment. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 532–539, 2013.



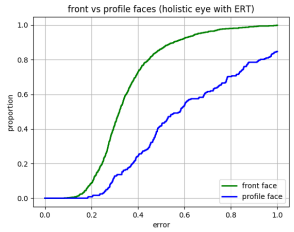
(a) Holistic eye of AAM



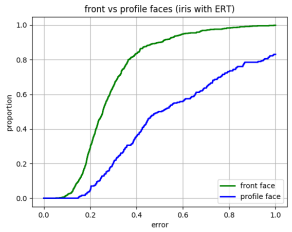
(b) Iris of eye of AAM



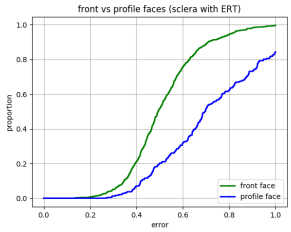
(c) Sclera of eye of AAM



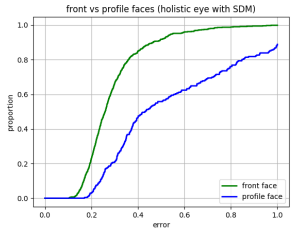
(d) Holistic eye of ERT



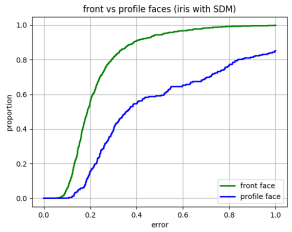
(e) Iris of eye of ERT



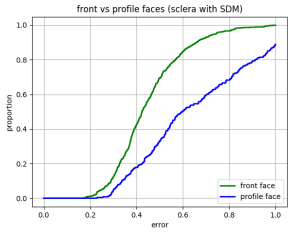
(f) Sclera of eye of ERT



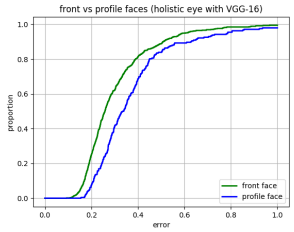
(g) Holistic eye of SDM



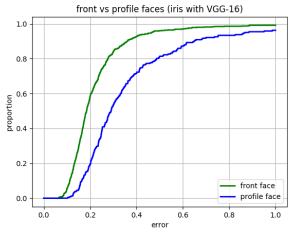
(h) Iris of eye of SDM



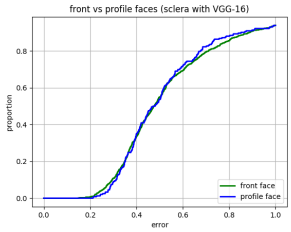
(i) Sclera of eye of SDM



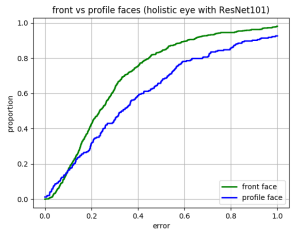
(j) Holistic eye of VGG-16



(k) Iris of eye of VGG-16

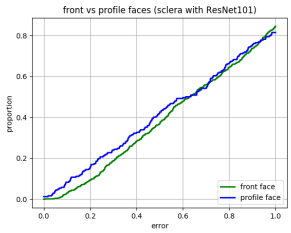


(l) Sclera of eye of VGG-16



(m) Holistic eye of ResNet101

(n) Iris of eye of ResNet101



(o) Sclera of eye of ResNet101

Figure 7 Appendix: Robustness evaluation compared between profile and frontal faces.

Anomaly Detection for Big Data Technologies

Ahmad Alnafessah
Department of Computing, Imperial College London, United Kingdom
a.alnafessah17@imperial.ac.uk

Giuliano Casale
Department of Computing, Imperial College London, United Kingdom
g.casale@imperial.ac.uk

Abstract

The main goal of this research is to contribute to automated performance anomaly detection for large-scale and complex distributed systems, especially for Big Data applications within cloud computing. The main points that we will investigate are:

- Automated detection of anomalous performance behaviors by finding the relevant performance metrics with which to characterize behavior of systems.
- Performance anomaly localization: To pinpoint the cause of a performance anomaly due to internal or external faults.
- Investigation of the possibility of anomaly prediction. Failure prediction aims to determine the possible occurrences of catastrophic events in the near future and will enable system developers to utilize effective monitoring solutions to guarantee system availability.
- Assessment for the potential of hybrid methods that combine machine learning with traditional methods used in performance for anomaly detection.

The topic of this research proposal will offer me the opportunity to more deeply apply my interest in the field of performance anomaly detection and prediction by investigating and using novel optimization strategies. In addition, this research provides a very interesting case of utilizing the anomaly detection techniques in a large-scale Big Data and cloud computing environment. Among the various Big Data technologies, in-memory processing technology like Apache Spark has become widely adopted by industries as result of its speed, generality, ease of use, and compatibility with other Big Data systems. Although Spark is developing gradually, currently there are still shortages in comprehensive performance analyses that specifically build for Spark and are used to detect performance anomalies. Therefore, this raises my interest in addressing this challenge by investigating new hybrid learning techniques for anomaly detection in large-scale and complex systems, especially for in-memory processing Big Data platforms within cloud computing.

2012 ACM Subject Classification Computing methodologies → Anomaly detection

Keywords and phrases Performance anomalies, Apache Spark, Neural Network, Resilient Distributed Dataset (RDD)

Digital Object Identifier 10.4230/OASIcs.ICCSW.2018.8

Category Poster Track

2018 Imperial College Computing Student Workshop (ICCSW 2018).
Editors: Edoardo Pirovano and Eva Graversen; Article No. 8; pp. 8:1–8:1

OpenAccess Series in Informatics
OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

A Novel Method for Event Detection using Wireless Sensor Networks

Ameer A. Al-Shammaa[1]
Department of Engineering, University of Leicester, United Kingdom
aambas2@le.ac.uk

A. J. Stocker[2]
Department of Engineering, University of Leicester, United Kingdom
sto@le.ac.uk

Abstract

Reliable event detection is one of the hottest research areas in the wireless sensor networks field these days. Battlefield monitoring, fire detection, nuclear and chemical attack, and gas leak detection are examples of the event detection applications. One of the main goals to WSNs is transmitting the sensed data to the sink (Base station) in an efficient way with minimum energy usage to achieve high degree of event detection reliability. Thus, Its very important to determine the reliability degree to know the number of data that are required to receive at the sink to achieve the desired reliability.

Most of the previous research works proposed different solutions for reliable event detection. The idea of all these solutions is based on increasing the amount of the transmitted data to the sink by controlling the sources reporting rate. However, rising the reporting rate may lead to losing the transmitted data due to the network congestion and packets collision, and this is related to the restricted resources capacity of the network's sensor nodes.

Therefore, in this paper, a new indoor method to achieve quality based event reliability for critical event detection have been implemented using hardware sensor nodes (Waspmote). The idea of this method is depending on sending the sensed data to the sink using a node called Cluster Head (CH) in a sequence according to their priority from the high to the low. The network nodes have been deployed in the experiment area into clusters, and each cluster have a CH node which work on collecting the cluster members readings and reorder it in descending order to send it next to the sink. The probability to deliver the important data to detect the event to the sink will increase by using this new method. The proposed mechanism intends to improve the event detection reliability, minimize the end-to-end delay, and increase the network lifetime. Experiments results show that the proposed method achieved a good the performance in terms of packets delivery, event detection, and end-to-end delay.

2012 ACM Subject Classification Networks

Keywords and phrases Waspmote nodes, Critical events, Wireless Sensor Networks, Clustering

Digital Object Identifier 10.4230/OASIcs.ICCSW.2018.9

Category Poster Track

[1] The work of Ameer A. Al-Shammaa is sponsored by the Ministry of Higher Education and Scientific Research, Iraq. Ameer is currently pursuing a PhD in the Aerospace and Computational Engineering research group at the University of Leicester, UK. He is also a staff member in the Training Department at IT-RDC at the University of Kufa, Iraq (e-mail: ameer.alshammaa@uokufa.edu.iq).
[2] Alan Stocker is an associate professor in the Department of Engineering at University of Leicester, UK. He is a staff member in the Aerospace and Computational Engineering research group.

Context-Aware Adaptive Biometrics System using Multiagents

Fatina Shukur
School of Computing, University of Buckingham, United Kingdom
fatinat.shukur@uokufa.edu.iq

Harin Sellahewa
School of Computing, University of Buckingham, United Kingdom
harin.sellahewa@buckingham.ac.uk

Abstract

Traditional biometric systems are designed and configured to operate in predefined circumstances to address the needs of a particular application. The performance of such biometrics systems tend to decrease because when they encounter varying conditions as they are unable to adapt to such variations. Many real-life scenarios require identification systems to recognise uncooperative people in uncontrolled environments. Therefore, there is a real need to design biometric systems that are aware of their context and be able to adapt to changing conditions.

The context-awareness and adaptation of a biometric system are based on a set of factors that include: the application (e.g. healthcare system, border control, unlock smart devices), environment (e.g. quiet/noisy, indoor/outdoor), desired and pre-defined requirements (e.g. speed, usability, reliability, accuracy, robustness to high/low quality samples), user of the system (e.g. cooperative or non-cooperative), the chosen modality (e.g. face, speech, gesture signature), and used techniques (e.g. pre-processing to normalise and clean biometrics data, feature extraction and classification). These factors are linked and might affect each other, hence the system has to work adaptively to meet its overall aim based to its operational context.

The aim of this research is to develop a multiagent based framework to represent a context-aware adaptive biometric system. This is to improve the decision making process at each processing step of traditional biometric identification systems. Agents will be used to provide the system with intelligence, adaptation, flexibility, automation, and reliability during the identification process. The framework will accommodate at least five agents, one for each of the five main processing steps of a typical biometric system (i.e. data capture, pre-processing, feature extraction, classification and decision). Each agent can contribute differently towards its designated goal to achieve the best possible solution by selecting/ applying the best technique. For example, an agent can be used to assess the quality of the input biometric sample to ensure the important features can be extracted and processed in further steps. Another agent can be used to pre-process the biometric sample if necessary. A third agent is used to select the appropriate set of features followed by another to select a suitable classifier that works well in a given condition.

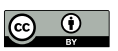
2012 ACM Subject Classification Security and privacy → Biometrics

Keywords and phrases Biometrics, Multiagents, Context-Aware

Digital Object Identifier 10.4230/OASIcs.ICCSW.2018.10

Category Poster Track

Acknowledgements The first author would like to thank HCED for supporting her PhD study.



2018 Imperial College Computing Student Workshop (ICCSW 2018).
Editors: Edoardo Pirovano and Eva Graversen; Article No. 10; pp. 10:1–10:1

OpenAccess Series in Informatics
OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Harnessing AI For Research

Matthew Johnson
Microsoft Research, Cambridge, United Kingdom
matjoh@microsoft.com

Abstract

Artificial Intelligence is increasingly being used to both augment existing fields of research and open up new avenues of discovery. From quality control for imaging flow cytometry to computational musicology, modern AI is an exciting new tool for research and thus knowing how to engineer AI systems in a research context is a vital new skill for RSEs to acquire. In this talk, I will outline four different areas of AI: supervised learning, unsupervised learning, interactive learning, and Bayesian learning. For each of these approaches, I will discuss how they typically map to different research problems and explore best practices for RSEs via specific use cases. At the end of the talk, you will have received a high-level overview of AI technologies and their use in research, have seen some cool examples of how AI has been used in a wide range of research areas, and have a good sense of where to go to learn more.

2012 ACM Subject Classification Computing methodologies → Artificial intelligence

Keywords and phrases Artificial intelligence

Digital Object Identifier 10.4230/OASIcs.ICCSW.2018.11

Category Invited Talk

2018 Imperial College Computing Student Workshop (ICCSW 2018).
Editors: Edoardo Pirovano and Eva Graversen; Article No. 11; pp. 11:1–11:1

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany