THE WEB 2014


Stephen Cranefield, Andrew Trotman and Jian Yang, Eds.

Conferences in Research and Practice in Information Technology, Volume 155.

Copyright ©2014, Australian Computer Society. Reproduction for academic, not-for-profit purposes permitted provided the copyright text at the foot of the first page of each paper is included.

Editors:

Andrew Trotman
Department of Computer Science
University of Otago
PO Box 56
Dunedin
New Zealand
Email: andrew@cs.otago.ac.nz

Stephen Cranefield
Department of Information Science
School of Business
University of Otago
Dunedin 9054
New Zealand
Email: stephen.cranefield@otago.ac.nz

Jian Yang
Department of Computing
Macquarie University
Sydney, NSW 2109
Australia
Email: jian.yang@mq.edu.au

Series Editors:
Vladimir Estivill-Castro, Griffith University, Queensland
Simeon J. Simoff, University of Western Sydney, NSW
Email: crpit@scem.uws.edu.au

Publisher: Australian Computer Society Inc.
PO Box Q534, QVB Post Office
Sydney 1230
New South Wales
Australia.

Conferences in Research and Practice in Information Technology, Volume 155.
ISSN 1445-1336.

Document engineering, January 2014 by CRPIT
On-line proceedings, January 2014 by the University of Western Sydney
Electronic media production, January 2014 by the AUT University

The Conferences in Research and Practice in Information Technology series disseminates the results of peer-reviewed research in all areas of Information Technology. Further details can be found at http://crpit.com/.
# Table of Contents


<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preface</td>
<td>vii</td>
</tr>
<tr>
<td>Conference Organisation</td>
<td>viii</td>
</tr>
<tr>
<td>Organising Committee</td>
<td>ix</td>
</tr>
<tr>
<td>Welcome from the Organising Committee</td>
<td>x</td>
</tr>
<tr>
<td>CORE - Computing Research &amp; Education</td>
<td>xi</td>
</tr>
<tr>
<td>ACSW Conferences and the Australian Computer Science Communications</td>
<td>xiii</td>
</tr>
<tr>
<td>ACSW and AWC 2014 Sponsors</td>
<td>xv</td>
</tr>
</tbody>
</table>

## Contributed Papers

- **Difference Computation for Grammar-Compressed XML Data**  
  *Stefan Böttcher, Rita Hartel and Sebastian Thiele*  
  Page 3

- **Keyword Search on DAG-compressed XML Data**  
  *Stefan Böttcher, Marc Brandenburg and Rita Hartel*  
  Page 13

- **Scalability in Recursively Stored Delta Compressed Collections of Files**  
  *Angelos Molfetas, Anthony Wirth and Justin Zobel*  
  Page 21

- **Extracting Crime Information from Online Newspaper Articles**  
  *Rexy Arulanandam, Bastin Tony Roy Savarimuthu and Maryam A. Purvis*  
  Page 31

- **Axiometrics: Axioms of Information Retrieval Effectiveness Metrics**  
  *Eddy Maddalena and Stefano Mizzaro*  
  Page 39

- **The Efficiency of Corpus-based Distributional Models for Literature-based Discovery on Large Data Sets**  
  *Michael Symonds, Peter Bruza and Laurianne Sitbon*  
  Page 49

- **A Systematic Approach to Measuring Advertising Transparency Online: An Australian Case Study**  
  *Paul A. Watters*  
  Page 59

## Author Index

Page 69
The next decade holds the prospect of remarkable progress in a wide range of innovations around the way people, organisations and society work and interact, due, in part, to continued advancements in Web technologies. The Australasian Web Conference (AWC), now in its second year, aims to provide a forum for researchers to share knowledge and present work in the field of Web research.

AWC 2014 was held as part of the Australasian Computer Science Week (ACSW) in Auckland, New Zealand. ACSW is the largest annual gathering of computing educators and researchers in Australasia, and in 2014 it comprised 13 conferences and workshops.

This year we received nine submissions with authors from six countries. The full written version of each submission received three anonymous reviews from independent members of the Programme Committee, which comprised 30 experts from 10 countries. The submissions were evaluated based on originality (novelty), problem significance, technical and scientific quality, and relevance to AWC 2014. The Programme Committee eventually selected seven full papers for presentation and inclusion in this volume. The selected papers cover a range of topics in the Web.

We thank all authors for their submissions and their active participation, and the Programme Committee members for their excellent work. We hope that you find the papers in the proceedings interesting and stimulating.

Stephen Cranefield
University of Otago

Jian Yang
Macquarie University

AWC 2014 Programme Chairs
January 2014
Conference Organisation

General Chair

Andrew Trotman, University of Otago, New Zealand

Programme Chairs

Stephen Cranefield, University of Otago, New Zealand
Jian Yang, Macquarie University, Australia

Program Committee

Helen Ashman, University of South Australia, Australia
Boualem Benatallah, University of New South Wales, Australia
Djamal Benlimane, Université Claude Bernard Lyon 1, France
Athman Bouguettaya, RMIT University, Australia
Steve Cassidy, Macquarie University, Australia
Florian Daniel, University of Trento, Italy
Jia Tina Du, University of South Australia
David Eyers, University of Otago, New Zealand
Dragan Gasevic, Athabasca University, Canada
Aditya Ghose, University of Wollongong, Australia
David Hawking, Funnelback Ltd. and Australian National University
Renato Iannella, Semantic Identity, Australia
Xiang-Fei Jia, University of Otago, New Zealand
Dong Seong Kim, University of Canterbury, New Zealand
Qing Li, City University of Hong Kong, Hong Kong, China
Xitong Li, MIT Sloan School of Management, USA
Lin Liu, Tsinghua University, China
Jiangang Ma, University of Adelaide, Australia
Zakaria Maamar, Zayed University, United Arab Emirates
Alistair Moffat, University of Melbourne, Australia
Hamid Motahari, IBM Almaden Research Center, USA
Chaoyi Pang, CSIRO, Australia
Russel Pears, Auckland University of Technology, New Zealand
Yongrui Qin, University of Adelaide, Australia
Falk Scholer, RMIT University, Australia
Quan Sheng, University of Adelaide, Australia
Hua Wang, University of Southern Queensland, Australia
Muhammad Younas, Oxford Brookes University, UK
Jian Yu, Swinburne University of Technology, Australia
Qi Yu, Rochester Institute of Technology, USA
Weiliang Zhao, University of Wollongong, Australia
Yong Zheng, DePaul University, USA
Zibin Zheng, Chinese University of Hong Kong, Hong Kong, China

Additional Reviewer

Yongrui Qin, University of Adelaide, Australia
Organising Committee

Chairs
Tony Clear and Russel Pears

Venue
Tony Clear

Communications
Russel Pears, Hui Ling Tan, Melanie Curry-Irons and Ryan Butler

Finance
Alison Clear and Eva Ihaia

Sponsorship
Stephen Thorpe

Operations
Adam Winship and Eva Ihaia

Programme, proceedings and booklet
Alison Clear

Catering and registration
AUT Hospitality Services
Welcome from the Organising Committee

On behalf of the Organising Committee, it is our pleasure to welcome you to Auckland and to the 2014 Australasian Computer Science Week (ACSW 2014). Auckland is New Zealand’s largest urban area with a population of nearly one and a half million people. As the centre of commerce and industry, Auckland is the most vibrant, bustling and multicultural city in New Zealand. With the largest Polynesian population in the world, this cultural influence is reflected in many different aspects of city life. ACSW 2014 will be hosted at the City Campus of Auckland University of Technology (AUT), which is situated just up from the Town Hall and the Auckland central business district. ACSW is the premier event for Computer Science researchers in Australasia. ACSW2014 consists of conferences covering a wide range of topics in Computer Science and related areas, including:

- Australasian Computer Science Conference (ACSC) (Chaired by Bruce Thomas and Dave Parry)
- Australasian Computing Education Conference (ACE) (Chaired by Jacqueline Whalley and Daryl D’Souza)
- Australasian Information Security Conference (AISC) (Chaired by Udaya Parampalli and Ian Welch)
- Australasian User Interface Conference (AUIC) (Chaired by Burkhard C. Wünsche and Stefan Marks)
- Australasian Symposium on Parallel and Distributed Computing (AusPDC) (Chaired by Bahman Javadi and Saurabh Kumar Garg)
- Australasian Workshop on Health Informatics and Knowledge Management (HIKM) (Chaired by James Warren)
- Asia-Pacific Conference on Conceptual Modelling (APCCM) (Chaired by Georg Grossmann and Motoshi Saeki)
- Australasian Web Conference (AWC2013) (Chaired by Andrew Trotman and Michael Sheng)

This year reflects an increased emphasis for ACSW on community building. Complementing these published technical volumes therefore, ACSW also hosts two doctoral consortia and a number of associated workshops, including those for the Heads and Professors of Computer Science, plus for the first time the ‘Australasian Women in Computing Celebration’. Naturally in addition to the technical program, there are a range of events, which aim to provide the opportunity for interactions among our participants. A welcome reception will be held in the atrium of the award winning newly built Sir Paul Reeves Building, which has integrated the city campus as a hub for student activity and provides a wonderful showcase for this year’s ACSW. The conference banquet will be held on campus in one of the reception rooms in this impressive complex.

Organising a multi-conference event such as ACSW is a challenging process even with many hands helping to distribute the workload, and actively cooperating to bring the events to fruition. This year has been no exception. We would like to share with you our gratitude towards all members of the organising committee for their combined efforts and dedication to the success of ACSW2014. We also thank all conference co-chairs and reviewers, for putting together the conference programs which are the heart of ACSW, and to the organisers of the symposia, workshops, poster sessions and accompanying conferences. Special thanks to Alex Potanin, as the steering committee chair who shared valuable experiences in organising ACSW and to John Grundy as chair of CoRE for his support for the innovations we have introduced this year. We’d also like to thank Hospitality Services from AUT, for their dedication and their efforts in conference registration, venue, catering and event organisation. This year we have secured generous support from several sponsors to help defray the costs of the event and we thank them for their welcome contributions. Last, but not least, we would like to thank all speakers, participants and attendees, and we look forward to several days of stimulating presentations, debates, friendly interactions and thoughtful discussions.

We hope your stay here will be both rewarding and memorable, and encourage you to take the time while in New Zealand to see some more of our beautiful country.

Tony Clear
Russel Pears
School of Computer & Mathematical Sciences

ACSW2014 General Co-Chairs
January, 2014
CORE welcomes all delegates to ACSW2014 in Auckland. CORE, the peak body representing academic computer science in Australia and New Zealand, is responsible for the annual ACSW series of meetings, which are a unique opportunity for our community to network and to discuss research and topics of mutual interest. The component conferences of ACSW have changed over time with additions and subtractions ACSC, ACE, AISC, AUIC, AusPDC, HIKM, ACDC, APCCM, CATS and AWC have now been joined by the Australasian women in computing celebration (AWIC), two doctoral consortia (ACDC and ACE-DC) and an Australasian Early Career Researchers Workshop (AECRW) which reflect the evolving dimensions of ACSW and build on the diversity of the Australasian computing community.

In 2014, we have again chosen to feature a small number of keynote speakers from across the discipline: Anthony Robins (ACE), John Mylopoulos (APCCM), and Peter Gutmann (AISC). I thank them for their contributions to ACSW2014. The efforts of the conference chairs and their program committees have led to strong programs in all the conferences, thanks very much for all your efforts. Thanks are particularly due to Tony Clear, Russel Pears and their colleagues for organising what promises to be a vibrant event.

Below I outline some of CORE’s activities in 2012/13.

I welcome feedback on these including other activities you think CORE should be active in.

The major sponsor of Australian Computer Science Week:
- The venue for the annual Heads and Professors meeting
- An opportunity for Australian & NZ computing staff and postgrads to network and help develop their research and teaching
- Substantial discounts for attendees from member departments
- A doctoral consortium at which postgrads can seek external expertise for their research
- An Early Career Research forum to provide ECRs input into their development

Sponsor of several research, teaching and service awards:
- Chris Wallace award for Distinguished Research Contribution
- CORE Teaching Award
- Australasian Distinguished Doctoral Dissertation
- John Hughes Distinguished Service Award
- Various Best Student Paper awards at ACSW

Development, maintenance, and publication of the CORE conference and journal rankings. In 2013 this includes a new portal with a range of holistic venue information and a community update of the CORE 2009 conference rankings.

Input into a number of community resources and issues of interest:
- Development of an agreed national curriculum defining Computer Science, Software Engineering, and Information Technology
- A central point for discussion of community issues such as research standards
- Various submissions on behalf of Computer Science Departments and Academics to relevant government and industry bodies, including recently on Australian Workplace ICT Skills development, the Schools Technology Curriculum and the Mathematics decadal plan

Coordination with other sector groups:
- Work with the ACS on curriculum and accreditation
- Work with groups such as ACDICT and government on issues such as CS staff performance metrics and appraisal, and recruitment of students into computing
- A member of CRA (Computing Research Association) and Informatics Europe. These organisations are the North American and European equivalents of CORE.
- A member of Science & Technology Australia, which provides eligibility for Science Meets Parliament and opportunity for input into government policy, and involvement with Science Meets Policymakers

A new Executive Committee from 2013 has been looking at a range of activities that CORE can lead or contribute to, including more developmental activities for CORE members. This has also included a revamp of the mailing lists, creation of discussion forums, identification of key issues for commentary and lobbying, and working with other groups to attract high aptitude students into ICT courses and careers.

Again, I welcome your active input into the direction of CORE in order to give our community improved visibility and impact.
CORE’s existence is due to the support of the member departments in Australia and New Zealand, and I thank them for their ongoing contributions, in commitment and in financial support. Finally, I am grateful to all those who gave their time to CORE in 2013, and look forward to the continuing shaping and development of CORE in 2014.

John Grundy
President, CORE
January, 2014
The Australasian Computer Science Week of conferences has been running in some form continuously since 1978. This makes it one of the longest running conferences in computer science. The proceedings of the week have been published as the Australian Computer Science Communications since 1979 (with the 1978 proceedings often referred to as Volume 0). Thus the sequence number of the Australasian Computer Science Conference is always one greater than the volume of the Communications. Below is a list of the conferences, their locations and hosts.

2015. Volume 37. Host and Venue - University of Western Sydney, NSW.


2013. Volume 35. Host and Venue - University of South Australia, Adelaide, SA.
2012. Volume 34. Host and Venue - RMIT University, Melbourne, VIC.
2011. Volume 33. Host and Venue - Curtin University of Technology, Perth, WA.
2010. Volume 32. Host and Venue - Queensland University of Technology, Brisbane, QLD.
2008. Volume 30. Host and Venue - University of Wollongong, NSW.
2007. Volume 29. Host and Venue - University of Ballarat, VIC. First running of HDKM.
2006. Volume 28. Host and Venue - University of Tasmania, TAS.
1998. Volume 20. Hosts - University of Western Australia, Murdoch University, Edith Cowan University and Curtin University. Venue - Perth, WA.
1995. Volume 17. Hosts - Flinders University, University of Adelaide and University of South Australia. Venue - Glenelg, SA.
1990. Volume 12. Host and Venue - Monash University, Melbourne, VIC. Joined by Database and Information Systems Conference which in 1992 became ADC (which stayed with ACSW) and ACIS (which now operates independently).
1989. Volume 11. Host and Venue - University of Wollongong, NSW.
1987. Volume 9. Host and Venue - Deakin University, VIC.
1986. Volume 8. Host and Venue - Australian National University, Canberra, ACT.
1983. Volume 5. Host and Venue - University of Sydney, NSW.
1982. Volume 4. Host and Venue - University of Western Australia, WA.
1981. Volume 3. Host and Venue - University of Queensland, QLD.
1980. Volume 2. Host and Venue - Australian National University, Canberra, ACT.
1979. Volume 1. Host and Venue - University of Tasmania, TAS.
1978. Volume 0. Host and Venue - University of New South Wales, NSW.
**Conference Acronyms**

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACDC</td>
<td>Australasian Computing Doctoral Consortium</td>
</tr>
<tr>
<td>ACE</td>
<td>Australasian Computing Education Conference</td>
</tr>
<tr>
<td>ACSC</td>
<td>Australasian Computer Science Conference</td>
</tr>
<tr>
<td>ACSW</td>
<td>Australasian Computer Science Week</td>
</tr>
<tr>
<td>ADC</td>
<td>Australasian Database Conference</td>
</tr>
<tr>
<td>AISC</td>
<td>Australasian Information Security Conference</td>
</tr>
<tr>
<td>APCCM</td>
<td>Asia-Pacific Conference on Conceptual Modelling</td>
</tr>
<tr>
<td>AUIC</td>
<td>Australasian User Interface Conference</td>
</tr>
<tr>
<td>AusPDC</td>
<td>Australasian Symposium on Parallel and Distributed Computing (replaces AusGrid)</td>
</tr>
<tr>
<td>AWC</td>
<td>Australasian Web Conference</td>
</tr>
<tr>
<td>CATS</td>
<td>Computing: Australasian Theory Symposium</td>
</tr>
<tr>
<td>HIKM</td>
<td>Australasian Workshop on Health Informatics and Knowledge Management</td>
</tr>
</tbody>
</table>

Note that various name changes have occurred, which have been indicated in the Conference Acronyms sections in respective CRPIT volumes.
ACSW and AWC 2014 Sponsors

We wish to thank the following sponsors for their contribution towards this conference.

Host Sponsor

Auckland University of Technology, www.aut.ac.nz

Platinum Sponsor


Gold Sponsor

The University of Auckland, www.auckland.ac.nz

Silver Sponsors

COLAB - AUT Design+Creative Technologies, colab.aut.ac.nz

Institute of IT Professionals, New Zealand, www.iitp.org.nz

Bronze Sponsor

SERL - AUT Software Engineering Research Laboratory, www.serl.aut.ac.nz

Australian Computer Society, www.acs.org.au

Publication Sponsor

University of Western Sydney, www.uws.edu.au
CONTRIBUTED PAPERS
Difference Computation for Grammar-Compressed XML Data

Stefan Böttcher, Rita Hartel, Sebastian Thiele
University of Paderborn (Germany), Computer Science,
Fürstenallee 11,
D-33102 Paderborn
{stb@, rst@, sthiele2@mail.}uni-paderborn.de

Abstract
Whenever web data processing requires the storage or the exchange of multiple versions of big XML data collections and the pure size of big XML data becomes a bottleneck in storage or fast data exchange over the web, XML compression and XML version control may become significant contributions to avoid such a bottleneck. Grammar-based compression is of increasing importance for big XML data collections in the web as it allows fast queries and updates on compressed data without full decompression. However, merging different versions of grammar-based compressed XML data collections is a challenge, because small differences in two given uncompressed XML files may lead to significant differences in the grammar-based compressed data formats of these files. Therefore, when multiple versions of an XML file have to be stored in compressed form, the different compressed formats may be difficult to combine, which weakens the benefit achieved by the compression. To overcome this weakening, we present a technique to compute the common part and the difference of two compressed XML documents without the need to fully decompress the documents. Our approach computes a compressed common prefix and parameters representing the difference of two compressed XML documents in polynomial time in the size of the grammar compressed documents, even if the common part of the documents is hidden in completely different sets of compressed grammar rules.

Keywords: Difference computation, grammar-based XML compression, XML data versions

1 Introduction
1.1 Motivation
Nowadays, in many industry branches, collections of data and documents in the web are stored in form of XML documents, as e.g. office documents, payment data (SEPA), touristic flight data (OTA), product catalogues, linked data, and many more are based on XML data formats. Whenever industry branches are required to archive and to exchange over the web daily versions of their documents for longer terms, a large amount of data has to be archived, yielding a large amount of storage, bandwidth, and energy costs. To overcome this problem, there exist mainly two different approaches: data compression on the one hand, and incremental difference computation, i.e. storing and exchanging one base version and difference files for all other versions, on the other hand. Both approaches reduce data sizes and thus storage and energy costs, but when the data and the number of versions grow, more data reduction will be required.

Therefore, it is an interesting task to combine both techniques in such a way that a stronger compression is reached by a combination of both, XML compression and XML versioning, than by XML compression or XML versioning alone, and that furthermore desired features of XML data like e.g. the queryability or updateability of the compressed data are maintained. Hereby, a key challenge is the fast computation of difference XML fragments on compressed XML files without decompressing too much data.

1.2 Contributions
In this paper, we propose an approach to combine grammar-based XML compression (like e.g. used in BPLEX(Busatto et al., 2005), CluX(Böttcher et al., 2010), TreeRePAIR(Lohrey et al., 2011)) with difference-based storing of multiple versions of the same XML document. In other words, our approach reads two compressed input tree grammars, $TG_1$ and $TG_2$, representing two versions of an XML document and computes one common output tree grammar $CTG$, representing the commonalities of $TG_1$ and $TG_2$ and two output difference tree grammars, $DTG_1$ and $DTG_2$, representing the differences between $TG_1$ and $TG_2$.

To the best of our knowledge, our approach is the first to combine the following advantages:

- It allows to compute the first difference of the two compressed XML trees that can be found in a pre-order walk through both trees in runtime that is polynomial in the size of the compressed tree grammars (i.e., without the need to decompress and materialize both XML trees).
- It computes a compressed format of a common prefix tree represented by $TG_1$ and $TG_2$, even if the grammar rules of $TG_1$ and $TG_2$ are completely different.
- It computes the differences as parameters of this common prefix trees, such that all operations that
can be performed on grammar-compressed data like e.g. query evaluation, transformation, or updates can be performed on the compressed versioned XML data too. It yields compressed versions that are in total 25-40% smaller than the already compressed input documents.

1.3 Paper Organization
This paper is organized as follows: Section 2 summarizes the basic idea of grammar-based compression followed by a description of how grammar based compression is being used for compressing XML data. Section 3 describes the fundamental concepts used by our approach for computing the commonalities and differences of two tree grammars in polynomial time. The fourth section outlines some of the experiments that compare our prototype with other text and XML versioning tools. Section 5 gives an overview of related work and is followed by the Summary and Conclusions.

2 A Summary of Grammar-Based Compression and the Paper's Example

2.1 The Paper’s Example Document
The simplest forms of grammar-based XML compressors are those compressors that combine identical sub-tree structures, such that the compressed grammar represents the minimal DAG of the XML tree (Buneman et al., 2003). Whenever a sub-tree occurs repeatedly within an XML document, a pointer to the first occurrence is stored instead of storing the repeated sub-tree another time. Our approach goes beyond the idea of DAG compression and uses a parameterized grammar for not only sharing identical sub-tree structures, but even similar sub-trees. It follows the idea of grammar-based compression as it was introduced in BPLEX (Busatto et al., 2005).

![Figure 1: (a) Example XML document D and (b) XML tree of D](image)

Figure 1 shows an example XML document D represented as a binary tree (without nullpointers representing empty sub-trees). It represents a book database where each book (b) has a title (t) and optionally an author (a) and/or an editor (e). The XML tree D can be generated by the following grammar, Grammar 1, using the non-terminal $S$ as the start symbol and using the symbol $#$ for the empty sub-tree, i.e., the right-hand side of the grammar rule is a term representing the pre-order notation of the binary tree given in Figure 1:

$$S \rightarrow bs(b(t(#,a(#,#))), b(t(#,a(#,#))), b(t(#,a(#,#))), b(t(#,e(#,#))))$$

Grammar 1: Grammar corresponding to the binary XML tree shown in Figure 1 (b).

2.2 The Idea of Grammar-Based Compression
Approaches like binary DAG compression, that share identical sub-trees $T$ in an XML document $D$, replace each occurrence of $T$ in $D$ with a non-terminal $N$ and add a grammar rule that defines $N$ to be a non-terminal that represents $T$. For example, in Grammar 1, there are four occurrences of the pattern $t(#,a(#,#))$. These occurrences are replaced by the nonterminal $TA$, such that we get the following grammar, Grammar 2:

$$TA \rightarrow t(#,a(#,#))$$

Grammar 2: Grammar corresponding to the binary DAG of the XML tree of Figure 1.

If we were only able to share identical sub-trees, we would only find sharable sub-trees consisting of title and author; i.e., we would not find bigger patterns of whole books, as they all differ in their next-sibling.

![Figure 2: Example document of Figure 1 with repeated patterns](image)

However, if we want to share structures that are not identical sub-trees, but similar sub-trees besides small differences, we find e.g. patterns consisting of book (‘b’), title (‘t’) and author (‘a’) as shown in Figure 2. For each of these patterns, there exist several matches in $D$ which are enclosed in boxes in Figure 2. Although the matches of the patterns have identical inner nodes, they cannot be shared in a DAG because the next-siblings of the ‘b’-nodes differ from each other. Figure 2 highlights 4 occurrences of the pattern $TA$ consisting of a ‘t’-node with an ‘a’-node as next-sibling, 4 occurrences of the pattern $BTA$ consisting of a ‘b’-node with the $TA$ pattern as first-child and different next-siblings of the ‘b’-node, and 2 occurrences of the $BTA2$ pattern consisting of a $BTA$ pattern with a second $BTA$ pattern as next-sibling of the ‘b’-node.

This compression can be represented as a parameterized grammar as follows. Within Grammar 3 below, we express e.g. the pattern $BTA(X)$ by one grammar rule with the left hand side $BTA(X)$ and the right-hand side $b(TA,X)$, where the parameter ‘X’ is being used for referring the different next-sibling nodes of the ‘b’-nodes occurring in the example document. When applying the grammar rule e.g. to the 4th ‘b’-node, $b(TA, b(t(#,e(#,#))))$, the empty sub-tree, i.e., the right-hand side of the grammar rule is a term representing the pre-order notation of the binary tree (without nullpointers representing empty sub-trees). It represents a book database where each book (b) has a title (t) and optionally an author (a) and/or an editor (e). The XML tree D can be generated by the following grammar, Grammar 1, using the non-terminal $S$ as the start symbol and using the symbol $#$ for the empty sub-tree, i.e., the right-hand side of the grammar rule is a term representing the pre-order notation of the binary tree given in Figure 1:
is called a match of the grammar rule’s right-hand side \( b(TA,X) \), with actual parameter \( X = b(t(#,e(#,#)), #) \), and \( BTA(b(t(#,e(#,#)), #)) \) is called a corresponding instantiation of the grammar rule’s pattern \( BTA(X) \). By replacing each match of a grammar rule’s right-hand side with a corresponding instantiation of the grammar rule’s pattern, we get the following grammar, Grammar 3, which is more compact than Grammar 2:

\[
\begin{align*}
S & \rightarrow \text{bs} \left( \text{BTA}^2 \left( \text{BTA}^2 \left( t(\#, e(\#, \#)), \#, \# \right) \right) \right), \# \\
\text{BTA}^2(X) & \rightarrow \text{BTA} \left( \text{BTA}(X) \right) \\
\text{BTA}(X) & \rightarrow b(TA,X) \\
\text{TA} & \rightarrow t(\#, a(\#, \#))
\end{align*}
\]

Grammar 3: A grammar sharing patterns by using parameterized rules and representing the XML tree of Figure 1.

All terminal nodes except \# have two parameters, i.e. the first-child and the next-sibling. However, non-terminal nodes may have an arbitrary number of parameters.

2.3 Grammars for Different Document Versions can Differ Significantly

The XML document shown in Figure 3 below differs from the one in Figure 1, as here, the third book has an editor child instead of an author child.

![Figure 3: A document version replacing the third author with an editor element](image)

Grammar 3b: A grammar representing the XML tree of Figure 3 (b).

Compressing the XML document of Figure 3 may lead to Grammar 3b. Note that compressed Grammar 3b significantly differs from Grammar 3, although the document in Figure 3 (a) differs only in one element from the document in Figure 1 (a). Given two grammars like Grammar 3 and Grammar 3b, our goal is to find a minimal grammar-based representation of both document versions, i.e. in Fig 1(a) and Fig 3(a), without a full de-compression of Grammar 3 and Grammar 3b.

3 Difference Computation for Grammar Compressed Data

3.1 Overview of the Difference Computation Algorithm

An algorithm contributed by Plandowski (Plandowski, 1994) allows to decide in polynomial time whether or not two String grammars \( S_1 \) and \( S_2 \) in Chomsky Normal Form (CNF) are equivalent, i.e., whether a unique word \( w(S_1) = w(S_2) \) can be derived from both grammars. Given two input tree grammars \( TG_1 \) and \( TG_2 \), we extend the idea of Plandowski in the following way. We compute one common tree grammar \( CTG \), representing the commonalities of \( TG_1 \) and \( TG_2 \), and two output difference tree grammars, \( DTG_1 \) and \( DTG_2 \), representing the differences between \( TG_1 \) and \( TG_2 \). The start rules of both grammars, \( DTG_1 \) and \( DTG_2 \), call the same top rule of \( CTG \), but with different parameters representing the differences between \( TG_1 \) and \( TG_2 \).

Our algorithm consists of the following steps.

In Step 1, we compute equivalent String grammars \( SG_1 \) and \( SG_2 \) of \( TG_1 \) and \( TG_2 \) respectively.

In Step 2, we use an extended version of the algorithm of Plandowski in order to compute the left-most difference of \( SG_1 \) and \( SG_2 \), and we mark the left-most commonalities computed for \( SG_1 \) and \( SG_2 \) in \( TG_1 \) and \( TG_2 \). In Step 3, we isolate from \( TG_1 \) those rules of the common tree grammar \( CTG \) which represent the left-most commonalities, and we construct a difference tree grammar \( DTG_1 \) representing where \( TG_1 \) differs from \( TG_2 \). For \( TG_2 \), we proceed correspondingly.

We finally repeat the steps 2 and 3 recursively on all corresponding parameters of the computed differences \( DTG_1 \) and \( DTG_2 \) to identify more common parts of \( TG_1 \) and \( TG_2 \), i.e., common parts that occur after the first difference, and to extend \( CTG \) to these commonalities.

Let \( D_1 \) and \( D_2 \) denote the document trees represented by the grammars \( TG_1 \) and \( TG_2 \). As a result of this computation, we get a grammar \( CTG \) representing the common prefix tree of \( D_1 \) and \( D_2 \), and a grammar \( DTG \) representing the remaining fragments of \( D_1 \) below the common prefix tree, and we get \( DTG_2 \), i.e. the same kind of grammar for \( D_2 \). Both, \( DTG_1 \) and \( DTG_2 \), call grammar rules of \( CTG \), but with different parameters.

These steps are described in more detail in the next sections.

3.2 Step 1: Linearization of a Tree Grammar into a CNF String Grammar

In the first step, we follow an idea presented in (Busatto et al., 2005) that computes an equivalent String grammar \( SG \) for a given tree grammar \( TG \). In order to linearize a grammar rule \( TGR \) of \( TG \), we apply the following two steps to all symbols having parameters:

1. For each terminal symbol \( t \), we compute the terminal symbols \( t_{0,1}, t_{1,2}, \) and \( t_{2,0} \). These new terminal symbols represent the traversing of the sub-tree rooted by symbol \( t \). Thereby, \( t_{0,1} \) represents the traversing from \( t \) to its first-child, \( t_{1,2} \) represents the traversing from the
first-child to the next-sibling, and $t_{20}$ represents the traversing from the next-sibling back to $t$.

2. Similarly, for each non-terminal symbol NT of rank $n$, i.e., that has $n$ parameters, we compute the new non-terminal symbols NT$_{01}$, NT$_{12}$, ..., NT$_{n-1n}$, NT$_{n0}$. The non-terminal NT$_{01}$ represents the traversing of the subtree NT($X_1$, ..., $X_n$) from its root NT to its first parameter $X_1$, a non-terminal NT$_{12}$ represents the traversing of the subtree from its $i^{th}$ parameter $X_i$ to its $i+1^{th}$ parameter $X_{i+1}$, and the non-terminal NT$_{n0}$ represents the traversing of the subtree from its $n^{th}$ parameter $X_n$ back to the sub-tree’s root.

Grammar 4 shows the String grammar representation that is equivalent to the tree grammar shown in Grammar 3. For a terminal symbol $t$(fc,ns), $t_{01}$ corresponds to ‘(‘, $t_{12}$ corresponds to the comma between fc and ns, and $t_{20}$ corresponds to the closing parenthesis. Therefore, $e(#,#)$ in the start rule $S$ of Grammar 3 is replaced with $e_{01} # e_{12} # e_{20}$ in the start rule $S$ of Grammar 4.

$$S \rightarrow b_{01} BTA_{20} b_{11} t_{01} # t_{12} e_{01} # e_{12} # e_{20} t_{20} b_{12} # b_{20} BTA_{20} BTA_{10}$$

$$BTA_{20} ightarrow BTA_{01} BTA_{10}$$

$$BTA_{01} ightarrow b_{01} T A b_{12}$$

$$T A ightarrow t_{01} # t_{12} a_{01} # a_{12} # a_{20} t_{20}$$

**Grammar 4: String grammar representation equivalent to the tree grammar Grammar 3**

Similarly, for the tree grammar rule defining BT($X$,Y) in Grammar 3b, the three grammar rules BT$_{01}$, BT$_{12}$, and BT$_{20}$ are defined in Grammar 4b, such that BT$_{01}$ corresponds to the non-terminal symbol BT and the opening parenthesis, BT$_{12}$ corresponds to the comma between X and Y, and BT$_{20}$ corresponds to the closing parenthesis. Therefore, the call of BT(a(#,#)) in rule BT$_{a}$ of Grammar 3b is replaced with the call of BT$_{a1}$ a$_{01} # a_{12} # a_{20}$ BT$_{12}$ in the rule BT$_{a01}$ of Grammar 4b.

Similarly, linearization applied to Grammar 3b results in the following Grammar 4b:

$$S^* \rightarrow b_{01} BTA_{01} BTEBTA_{01}$$

$$BTEBTA_{01} \rightarrow BTEBTA_{10}$$

$$BTEBTA_{10} \rightarrow BTEBTA_{01} BTA_{51} BTA_{10} b_{12} # b_{20}$$

$$BTEBTA_{10} \rightarrow BTA_{01} BTA_{10} e_{01} # e_{12} # e_{20} BTA_{12}$$

$$BTEBTA_{10} \rightarrow BTA_{10} BTA_{01}$$

$$BTA_{01} \rightarrow BTA_{01} a_{01} # a_{12} # a_{20} BTA_{12}$$

$$BTA_{10} \rightarrow BTA_{20} BTA_{10}$$

$$BTA_{01} \rightarrow BTA_{01} t_{01} # t_{12}$$

$$BTA_{12} \rightarrow BTA_{20} BTA_{12}$$

**Grammar 4b: String grammar representation of the tree grammar Grammar 3b**

We finish the first step of our approach by computing the Chomsky Normal Form CNF of the String grammar SG that was the result of the linearization.

### 3.3 Step 2: Extended Plandowski Algorithm to Determine the First Difference

In this section, we only outline the basic concepts of the algorithm of Plandowski, before we describe our modifications in more detail. For further details on the algorithm of Plandowski, please refer to (Plandowski, 1994).

The algorithm of Plandowski checks for two non-terminal symbols A and B of a context-free grammar in CNF, whether they produce the same words, i.e. whether $w(A)=w(B)$. The basic idea of the algorithm is to reduce the comparison of the words generated by two non-terminal symbols to the comparison of smaller words generated by other non-terminal symbols, until the comparison task is reduced to the comparison of two terminal symbols. As we use each grammar for representing a single compressed XML document, we have to reduce the comparison of two words, $w(A)$ and $w(B)$, only.

Consider for example the grammars Grammar 5 and Grammar 6 shown below:

<table>
<thead>
<tr>
<th>Grammar 5</th>
<th>Grammar 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_1 \rightarrow AB_1 C_1$</td>
<td>$S_2 \rightarrow A_2 BC_2$</td>
</tr>
<tr>
<td>$AB_1 \rightarrow A_1 B_1$</td>
<td>$BC_2 \rightarrow B_2 C_2$</td>
</tr>
<tr>
<td>$A_1 \rightarrow a$</td>
<td>$A_2 \rightarrow a$</td>
</tr>
<tr>
<td>$B_1 \rightarrow b$</td>
<td>$B_2 \rightarrow b$</td>
</tr>
<tr>
<td>$C_1 \rightarrow c$</td>
<td>$C_2 \rightarrow c$</td>
</tr>
</tbody>
</table>

**Grammar 5 and Grammar 6: CNF String grammars producing the word ‘abc’**

In addition to the definition of the non-terminal symbols, we have to know the sizes of the words produced by each symbol, which can be efficiently computed bottom-up.

Let us assume that we want to compare the words produced by $S_1$ and $S_2$. As $S_1 \rightarrow AB_1 C_1$, we can reduce this comparison to first, comparing $AB_1$ to the beginning of $S_2$ until position 2, and second, comparing $C_1$ to $S_2$ starting at the third position (as $AB_1$ has length 2). Furthermore, comparing $AB_1$ to the beginning of $S_2$ can be reduced to first, comparing $A_2$ to the beginning of $AB_1$, and second, comparing $BC_2$ to $AB_1$ starting at position 2 (as $A_2$ has length 1). Finally, the result of the algorithm is that the words $w(S_1)$ and $w(S_2)$ produced by $S_1$ and $S_2$ are equivalent, if and only if the following holds: the words produced by $A_1$ and $A_2$ are equivalent, the words produced by $B_1$ and $B_2$ are equivalent, and the words produced by $C_1$ and $C_2$ are equivalent. As each of these pairs of non-terminals is defined by the same single terminal symbol each, this can be checked easily to be true.

Plandowski presents some additional optimizations that allow pruning certain branches, such that the number of comparisons becomes polynomial.

We extend the algorithm of Plandowski by an additional value $v$ counting the number of equal characters found in the prefixes of $S_1$ and $S_2$ and by an additional value $v$, where $v$ specifies the final position of the substring that is currently compared. The algorithm stops when the left-most difference is found during finally comparing terminal symbols, and sets $v$ to the position of
the left-most difference, i.e., \( v-1 \) indicates the size of the equal prefix produced by both words.

When comparing the words generated by two non-terminal symbols \( S_1 \) of \( SG_1 \) and \( S_2 \) of \( SG_2 \), initially \( u = 0 \) and \( v = \min(\text{size}(S_1), \text{size}(S_2)) \), where \( \text{size}(S) \) is the length of the word generated by \( S \). Then, as long as \( v > u \), we compare the words generated by two non-terminal symbols \( R_1 \) of \( SG_1 \) and \( R_2 \) of \( SG_2 \), and as there are rules \( R_1 \rightarrow R_{11} R_{12} \) in \( SG_1 \) and \( R_2 \rightarrow R_{21} R_{22} \) in \( SG_2 \), we apply a reduction step to the rules and to the comparisons considered as follows:

If \( R_{11} \) and \( R_{21} \) are both terminals and are equal, we increase \( u \) by 1 and continue with the comparison of \( R_{12} \) and \( R_{22} \). If \( R_{11} \) and \( R_{21} \) are both terminals and are not equal, we set \( v = u \) and stop the algorithm as we have found the difference. Otherwise, we reduce the rule for \( R_1 \) if \( \text{size}(R_{11}) > \text{size}(R_{21}) \), or the rule for \( R_2 \) if \( \text{size}(R_{11}) \leq \text{size}(R_{21}) \).

Whenever a reduction step for one rule \( R_1 \rightarrow R_{11} R_{12} \) is performed, the other rule \( R_2 \rightarrow R_{21} R_{22} \) remains unchanged. The comparison \( c_{12} \) comparing \( R_{12} \) to the beginning of \( R_2 \), and we set \( v = 1 \) initially to minimum(\( \text{size}(R_{11}), \text{size}(R_{21}) \)). Second, we apply the comparison \( c_{21} \) comparing \( R_{21} \) to the string produced by \( R_1 \) starting at the position \( \text{size}(R_{11})+1 \). The initial value \( v = 1 \) for the comparison \( c_{21} \) will be \( v_1 \) because the rightmost symbol to be compared does not change.

For example, the initial value of \( v \) for the comparison \( c_0 \) comparing \( S_1 \) of Grammar 5 to \( S_2 \) of Grammar 6 is 3. When we reduce the comparison \( c_0 \) to the comparison \( c_1 \) comparing \( AB_1 \) to the beginning of \( S_2 \) and to the comparison \( c_2 \) comparing \( C_1 \) to \( S_2 \) starting at position 2, then, \( v(c_1)=2 \) and \( v(c_2)=3 \).

The result of Step 2 will be the size of the common prefix of the non-terminal symbols \( S_1 \) and \( S_2 \). This size is then passed to Step 3.

When applying Step 2 to the Chomsky normal forms of Grammar 4 and Grammar 4b, we get the common prefix \( b_0_1 b_0_1 b_0_1 \# t_1_2 a_0_1 \# a_1_0 \# a_2_0 t_2_0 b_1_2 b_0_1 b_0_1 \# t_1_2 a_0_1 \# a_1_0 \# a_2_0 t_2_0 b_1_2 b_0_1 b_0_1 \# t_1_2 \) and \( v = 27 \).

### 3.4 Splitting Grammars into Common Grammar Tree and List of Actual Parameter Values

#### 3.4.1 The Goal of Step 3 and Step 4

In the previous step, we have computed the number of terminal symbols that are equal when reading the words produced by the start rules of the two linearized String grammars from the left to the right. Now, we want to transfer this result back to the two tree grammars that are the initial input of our approach.

The goal of Step 3 and Step 4 is to split each grammar \( G \) into two parts: On the one hand, a common tree grammar \( CTG \) that contains the common ‘prefix’ that precedes the occurrence of the first different terminal node of both trees. On the other hand, a list of actual parameters \( ap_1, \ldots, ap_m \), such that \( CTG \) invoked with \( ap_1, \ldots, ap_m \) is equivalent to \( G \), i.e., it produces the same tree as \( G \).

More precisely, let \( d \) be the first node that differs in the pre-order walk through the trees produced by both grammars, and let \( p(d) \) be the ‘predecessor’ of \( d \), i.e., either the parent node, if \( d \) is the first-child of \( p(d) \) or the previous-sibling, if \( d \) is the next-sibling of \( p(d) \). Let furthermore \( r \) be the root of the document represented by the grammar and \( v_1, \ldots, v_n \) be the nodes on the path from the root to \( p(d) \) in the unranked tree. Then, we want to split each grammar \( G \) into a common grammar \( CTG \) and parameters as follows:

CTG is a collection of grammar rules that represents the common tree containing all nodes that occur before \( d \) in document order. Furthermore, the subtrees of \( G \) rooted in \( d \) or in the next-siblings of the nodes \( r, v_1, \ldots, v_n \), \( p(d) \) are substituted by formal parameters, and these formal parameters will occur in the top-rule of CTG. As the top rule of CTG contains formal parameters, CTG not only defines a single document, but a set of documents depending on the actual parameters with which CTG is invoked.

On the other hand, we get a list of actual parameters \( ap_1, \ldots, ap_m \) such that \( CTG \) invoked with \( ap_1, \ldots, ap_m \) is equivalent to \( G \), i.e., it produces the same tree as \( G \). This list of parameters contains the sub-trees rooted by \( d \) or by the next-siblings of \( v_n, \ldots, v_1 \), or \( r \) respectively.

For example, to find the first difference between the grammars Grammar 3a and Grammar 3b, let us compare the tree representation given in Figure 3 with the tree representation given in Figure 3 (b). In the tree of Figure 2, the node \( d \) is the \( a\)-child of the \( 3^{rd} \) \( b\)-node, \( p(d) \) is the \( t\)-node that is the preceding-sibling of \( d \), \( v_1 \) is the \( 3^{rd} \) \( b\)-node, and \( r \) is the \( bs\)-node. The nodes belonging to CTG are colored white, whereas the three actual parameters (i.e., the sub-trees rooted by the next-siblings of \( p(d), v_1 \), and \( r \) respectively) that were split from CTG are colored in different shades of gray.

#### 3.4.2 Step 3: A Derivation Tree for the CTG

In Figure 2, each instantiation of a grammar rule is represented by an enclosing box. We can distinguish three categories of grammar rule instantiations:

- Instantiations of rules the corresponding match of which contains nodes on the fc-ms-path from the root node to the node \( p(d) \). Only for each of these rules, we create a copy in which we replace each sub-tree which is rooted in \( d \) or in a next-siblings of \( v_n, \ldots, v_1 \), or \( r \) by a formal parameter.

- Instantiations of rules the corresponding match of which contains no nodes of the root-to-\( p(d) \)-path and belongs to the common part preceding \( p(d) \). These rule calls can be skipped, and these matches are not decompressed during the run of our algorithm.
— Instantiations of rules the corresponding match of which contains no nodes of the root-to-p(d)-path and belongs to the nodes following p(d), i.e., to the difference part. The algorithm stops before these instantiations are reached and these matches are not decompressed during the run of our algorithm.

— In order to skip the instantiations of the second category, we process the grammar once bottom-up to calculate the size of each instantiation, i.e., the number of nodes of the corresponding match in the decompressed tree. This calculation is performed prior to the computation of the common tree grammar.

<table>
<thead>
<tr>
<th>Instandiation Inst</th>
<th>Size</th>
<th>of Inst</th>
</tr>
</thead>
<tbody>
<tr>
<td>TA</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>BTA(X)</td>
<td>12 +</td>
<td>[X]</td>
</tr>
<tr>
<td>BTA(BTA(X))</td>
<td>24 +</td>
<td>[X]</td>
</tr>
<tr>
<td>BTA2(X)</td>
<td>24 +</td>
<td>[X]</td>
</tr>
<tr>
<td>BTA2(#)</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>BTA2(b(#,e(#,#),BTA2(#)))</td>
<td>65</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Sizes of all grammar rules and all instantiations of Grammar 3

Table 1 shows the sizes of all grammar rules and all instantiations of rules occurring in Grammar 3. All sizes are computed in a single bottom-up run through the grammar. For each rule, the size of the rule is the number of terminals, null-pointers, commas that separate the first-child and the next-sibling of a terminal and closing parentheses that follow the next-sibling of a terminal. For each instantiation of a grammar rule RG, we add the size of RG plus the sizes of the actual parameters of RG.

In order to split the grammar into the common part CTG and into the list of parameters, we process the grammar top-down in order to compute the positions where to split actual parameter values from CTG. We compute a derivation tree (c.f. Figure 4) for the grammar rules belonging to CTG. The derivation tree contains rules that have instantiations of the first and of the second category. For each rule instantiation that belongs to the first category (i.e., one part of it belongs to the common tree and another part does not), the derivation tree contains an extracted copy of the rule. Within the grammar rules in this derivation tree, we mark all the symbols that represent the common nodes (i.e. nodes visualized as white nodes in Figure 2). These symbols include terminals, null-pointers, commas and closing-parentheses read and all non-terminals of rule instantiations belonging to the first or the second category. All marked items belong to CTG, whereas all non-marked items are replaced by formal parameters within CTG and their values have to be stored in the list of actual parameter values.

The calculation of the derivation tree gets as input the tree grammar and the number of symbols v before the first difference. Initially, the derivation tree contains a single node containing a copy of the start rule S of the grammar G, and the computation of the derivation tree starts at the first symbol of the right-hand-side of S and proceeds as follows:

— Whenever an instantiation I of a rule R with a non-terminal NT is read, we mark NT as read, and look up the size s of I. If s=v, I can be skipped and v=v-s. Otherwise: We extract R, attach a node N containing a copy of R as child-node of the current derivation tree node and proceed with the node N by processing the first symbol of N’s right-hand-side. We call I the source instantiation of N. As soon as we have processed all symbols of N, we continue processing at the first non-marked symbol of N’s parent in the derivation tree.

— Whenever a terminal-symbol t is read, we decrease v by 1 and mark t as read. Afterwards, we continue processing the first non-marked symbol of the current rule. The same holds for nullpointers, commas that separate the first-child and the next-sibling of a terminal, and closing parentheses that follow the next-sibling of a terminal.

— Whenever a formal parameter fp is read, we mark fp as read, and we go to the parent of the current node cn and read the actual parameter value, that is the first non-marked symbol av and proceed processing av. As soon as the processing of av is finished, we continue with the first non-marked symbol of cn.

— When v gets the value v=0, the algorithm is stopped.

Figure 4 shows the derivation tree for Grammar 3 and v=27. The marked symbols are underlined. The two left-most branches end with the extraction of BTA(X), as TA does not have to be extracted, as the size of TA(=9) is smaller than the value of v at that time.

```
[4
| S → l(x(BTA2(BTA2(x(t(#,#),e(#,#,#),BTA2(#))))), #)) #)
```

Figure 4: Result of Step 4 for Grammar 3 and v=27

The set of terminal symbols that are marked corresponds to the set of white nodes of Figure 2.

In the following step, the derivation tree is processed bottom-up in order to compute CTG and the list of parameters.

3.4.3 Step 4: Computing CTG and the Parameter List

In this step, we process the derivation tree, which was the result of the previous step, bottom-up and calculate CTG and the list of actual parameters.

Starting with the left-most leaf-node of the derivation tree, we generate a copy of the rule and replace each non-
marked instantiation of a non-terminals rule or each sub-tree rooted by a non-marked terminal or each non-marked nullpointer by a formal-parameter, and we store the value that was replaced by the formal parameter as the value of the actual parameter. Then, we compare the copy with the original version to check, whether anything was changed. If not, we just delete this node of the derivation tree. Otherwise, we repeatedly assign a new non-terminal to the modified rule, store the new rule in the grammar, and propagate the changes upwards in the tree, i.e., we change the non-terminal symbol of the source instantiation of this node and adapt the parameters of this instantiation. Whenever we read a non-terminal symbol that was marked, we store the information that the corresponding rule is part of CTG. The algorithm stops when the root of the syntax tree has been processed. All new rules and all rules a marked non-terminal of which was read belong to CTG. The actual parameters propagated to the root of the syntax trees of each of the grammars describe the different parts of both grammars.

When processing the derivation tree of Figure 4, we do not store modified rules for the two left-most leaf nodes, as these nodes are marked completely, i.e., the rules are not modified, such that no symbol is replaced by a parameter. For the right-most leaf node, we generate a rule $TA(X) \rightarrow t(#,X)$ with parameter binding $X=a(#,#)$. Then, we propagate this parameter binding upwards and generate the rule $BTA'(X,Y) \rightarrow b(TA'(X),Y)$ with $X=a(#,#)$ and $Y=X$ and the rule $BTA2'(X,Y) \rightarrow BTA'(X,Y)$ with $X=a(#,#)$ and $Y=BTA(X)$. Finally, the top rule is $S' \rightarrow bs(BTA2(BTA2'(X,Y)), Z)$ with $X=a(#,#)$, $Y=BTA(b((#,#,#),#))$ and $Z=#$.

As result of this step, we receive the CTG as given in Grammar 7 and the difference grammar as given in Grammar 7a, such that, if we combine grammars 7 and 7a and start with start symbol $S$, these grammars define the same XML tree as Grammar 3 starting with start symbol $S$. Furthermore, if we apply this approach to Grammar 3b and $v=27$, we get the second difference grammar as given in Grammar 7b, such that, if we combine grammars 7 and 7b and start with start symbol $S^*$, these grammars define the same XML tree as Grammar 3b starting with start symbol $S^*$.

### Grammar 7: CTG for Grammar 3 and Grammar 3b

$$S \rightarrow S'(a(#,#), BTA(b(t(#,e(#,#)),#)))$$

### Grammar 7a: DTG for Grammar 3 based on the CTG of Grammar 7

$$S^* \rightarrow BTA^*(b(t(#,#),#))$$

### Grammar 7b: DTG for Grammar 3b based on the CTG of Grammar 7

$$S^* \rightarrow BTA^*(BTA^*(a(#,#),#))$$

**Remark:** When computing the algorithm for Grammar 3b, we would get a second common tree grammar CTG’ which might differ in the textual representation, but which is equivalent to CTG, i.e., CTG and CTG’ started with the same parameters will create identical trees. If we chose CTG’ as common tree graph (as e.g. it is more compact than CTG), the difference grammar has to store besides the grammar rule $S$ the rules for BTA2, BTA and TA, as they are used directly or indirectly within $S$.

### Step 5: Recursive Computation of Further Commonalities

Up to now, we only have indentified the left-most identical terminal symbols of both grammars. Now, we recursively apply our approach to all the other corresponding pairs of actual parameter values $ap_1$ and $ap_2$ occurring at corresponding positions within the actual parameter list of DTG and DTG2 in order to extend CTG to the commonalities that are still contained within the actual parameter values of DTG and DTG2.

Continuing the previous example, we apply the difference computation to compare the first parameters of $S$ and $S^*$ with each other. As they differ already in the root node, no commonality can be found. When comparing the second pair of parameters with each other, we get the result, that both produce identical trees, such that these parameters can be integrated into CTG. The same holds for the third pair of parameters which are both a ‘#’, i.e., a null-pointer.

As shown in Grammars 8, 8a and 8b, this leads to the following extension of the CTG and smaller difference grammars DTG and DTG2.

### Grammar 8: CTG for Grammar 3 and Grammar 3b after having completed all recursive steps

$$S \rightarrow S'(a(#,#))$$

### Grammar 8a, 8b: DTGs for Grammars 3 and 3b respectively based on the CTG of Grammar 8 after having completed all recursive steps

$$S^* \rightarrow S'(e(#,#))$$
4 Evaluation of our Prototype Implementation

4.1 Evaluation Environment

The goal of our evaluation was to examine whether the combination of versioning and compression will lead to smaller file sizes than compression or versioning alone.

As test documents, we used documents created with the help of the XMark Benchmark (Schmidt et al., 2002) with factors of 0.002 to 0.064 yielding XML documents of sizes from 200 kB to more than 7 MB. In our tests, we only regarded the structure of these documents without any text nodes.

For each of these base documents XML1, we created 3 additional documents XML2 by applying modifications at random positions of XML1.

We compared our approach using 4 different scenarios:

- XMLDiff: We apply versioning, but we do not apply compression. We use javaxdelta an implementation of the VCDIFF approach as described in (Bell & Mcilroy, 1999) to create a delta version d2 from XML1 to XML2 and store XML1 as plain text and the delta in its own compressed format as provided by javaxdelta.
- CluX1 + CluX2: We apply compression, but we do not apply versioning. We compress XML1 to CluX1 and XML2 to CluX2 using grammar-based compression and store CluX1 and CluX2.
- CluXDiff: We apply compression and versioning. We apply our approach to CluX1 and CluX2 and store the resulting grammars CTG, DTG1 and DTG2.
- All tests were performed on an Intel Core2 Duo CPU P8700 @ 2.53 GHz. The tests were performed using Java 1.6.

4.2 Evaluation results

We performed two series of measurements to examine whether the combination of compression and versioning will lead to smaller file sizes than compression or versioning alone.

In the first series of measurements, we examined the effect of scaling file sizes. We increased the document size of XML1 from 200 kB to more than 7 MB and performed 1 relabel operation, 1 deletion, and 1 insertion at random positions within XML1 to create XML2.

Figure 5(a) shows the results of our measurements. The file sizes of XML1 and XML2 define 100%. If we apply versioning only (XMLDiff), we get file sizes of constantly about 42% of |XML1 + XML2| (as XML2 was larger than XML1). If we apply compression only (CluX1+CluX2), we get file sizes of 27% for small documents down to 8% for larger documents (as those contain more redundancies to be eliminated by compression). And we get in fact the smallest file sizes by our combination of compression and versioning (CluXDiff) reaching file sizes of 17% for smaller documents down to 6% for larger documents.

In the second series of measurements, the document XML1 has a roughly a size of 2 MB, and we scaled the number of update operations (X relabels, X deletions, and X insertions, with X ∈ {1,3,5}) performed on XML1 to compute XML2. Figure 5(b) shows the results. The number of update operations merely affects the compression ratio of all three approaches. The approach based on compression only shows the most stable behavior, whereas the approaches based on versioning show a slightly worse compression when there are more modifications (as the common part of both versions becomes more and more ragged).

The combination of compression and versioning not only yields smaller files than performing compression only or versioning only, but also has a further advantage: When searching for certain data within all versions (e.g. for the evaluation of a given XPath query), there is no need to restore all versions, as it was the case for the scenario using XMLDiff. Instead, we can evaluate the XPath query on the combination of DTG1, DTG2, and CTG, i.e., that part of the query evaluation that has to be done on the common part, CTG, is done only once, thereby inheriting the advantages of query evaluation on grammar-compressed data, which is in many cases even more efficient than query evaluation on XML tree data.

Figure 5: Compression ratio scaled by (a) document size and (b) number of modifications
5 Related Works

Regarding XML structure compression, there exist several approaches, which can be mainly divided into three categories: encoding-based compressors (e.g. XMill (Sun et al., 2007)) XGrind (Tolani & Haritsa, 2002), XPRESS (Min et al., 2003), XQueC (Arion et al., 2007), (Bayardo Jr. et al., 2004), (Cheney, 2001), (Girardot & Sundaresan, 2000), and (Zhang et al., 2004)), schema-based compressors (e.g. XCQ (Ng et al., 2006), XAUST (Subramanian & Shankar, 2005), Xenia (Werner et al., 2006) and DT Diagnosis (Böttcher et al., 2007)), and grammar-based compressors.

To the group of grammar-based compressors belong XMLZip (Cheng & Ng, 2004), the approaches presented in (Adiego et al., 2004) and (Buneman et al., 2003), and the BPLEX algorithm (Busatto et al., 2005). They compress the data structure of an XML document by combining identical or similar sub-trees. As they all follow the same idea as the compression technique on which the approach presented in this paper is based on, the ideas of this paper should be applicable to these compression approaches, too.

Versioning approaches that compute the difference of two files on byte-level should be applicable to the compressed XML documents, no matter to which group they belong. However, this combination of XML compression and versioning has the disadvantage that certain features of the compression technique, like e.g. the capability to evaluate queries on the compressed data directly, get lost, when it is combined with byte-level versioning.

There exist several approaches to compute the difference of text files which could be applied to XML files as well. The most commonly used approaches might probably be the GNU diff utility which is based on the Longest Common Subsequence algorithm (Myers, 1986), xDelta based on VXDIFF (Bell & Mcilroy, 1999) or CVS which is based on the GNU diff utility. However, they cannot be directly applied to tree-structured data.

For hierarchically structured data, there exist several approaches. Some of these approaches compare different versions of trees, whereas other approaches were especially designed to compute the differences of two XML files.

(Zhang et al., 1989) proposes an algorithm to find an optimal edit script for two ordered, labeled trees. (Cobena et al., 2002) proposes XyDiff, an algorithm that allows to detect changes in XML documents. XyDiff first computes bottom-up a hash-value and a weight for each document node of both documents. Then, in a top-down run, it matches nodes of equal signature and tries to propagate these matchings to the nodes’ parents. XyDiff does not guarantee to find the optimal result. X-Diff (Wang et al., 2003) extends the ideas of XyDiff. It computes bottom-up a hash-value and a so-called signature for each node, whereas the signature consists of the root-to-node path and the type of the node (e.g. root node, element node, text node, …). In the next step, it matches nodes of both documents that have the same hash value. Finally, it computes bottom-up the edit distance for all nodes with the same signature. It follows the idea of dynamic programming and computes a distance matrix that allows calculating a matching of minimal costs. X-Diff runs in polynomial time and always computes the optimal result.

Beside these approaches for ordered trees, there exist approaches to compare two versions of unordered trees (e.g. MH-Diff (Chawathe & Garcia-Molina, 1997)), which might as well be applied to XML documents depending on whether or not the order of the tree should be regarded. (Zhang et al., 2004) follows a different idea of XML versioning. It does not try to find changes in the structure of the document, but it defines an XML versioning based on the semantical changes of two XML documents.

To the best of our knowledge, there does not exist any approach that tries to combine the advantages of XML compression and versioning and that allows to evaluate queries on the results of the difference computation.

6 Summary and Conclusions

Whenever multiple versions of big XML data collections, e.g. linked data, office documents, payment data (SEPA), touristic flight data (OTA), product catalogues, etc., have to be stored and transferred over the web, and when the pure size of the big XML data is a bottleneck, it may be a significant advantage to use compression and difference computation. Using grammar-based compression prior to archiving each XML document has the additional advantage that queries and update operations on compressed documents can be run without full decompression. However, when grammar-based compression is used for compressing similar XML documents or similar versions of XML documents, the compressed formats of the similar XML documents may differ significantly, which makes it a challenge to compute the difference of two compressed XML documents without a huge amount of decompression.

Our approach searches and finds the commonalities of two different documents or document versions in their compressed representations without full decompression, even if these documents have been compressed individually and the compressed versions differ a lot. Our algorithm computes a new grammar for the common parts of multiple XML documents, and returns the differences as different parameters, even if the common part of the documents is hidden in completely different sets of compressed grammar rules.

Our evaluation has shown that the combination of compression and versioning yields smaller files than performing compression only or versioning only. Even more, we expect that when searching for certain data within all versions (e.g. the evaluation of a given XPath query), our approach will show a further advantage: We do not need to restore all versions as for traditional versioning. Instead, we can evaluate the XPath query on the combination of DTG₁, DTG₂ and CTG, thereby inheriting the advantages of query evaluation on grammar-compressed data, which is in many cases even more efficient than query evaluation on XML tree data.
7 References


Keyword Search on DAG-Compressed XML Data

Stefan Böttcher, Marc Brandenburg, Rita Hartel
University of Paderborn (Germany), Computer Science,
Fürstenallee 11,
D-33102 Paderborn
stb@uni-paderborn.de, mbrande3@gmail.com, rst@uni-paderborn.de

Abstract
With the growing size of publicly available XML document collections, fast keyword search becomes increasingly important. We present an indexing and keyword search technique that is suitable for DAG-compressed data and has the advantage that common sub-trees have to be searched only once. We also present a performance evaluation that shows that our DAG-compressed index and search technique is superior to the corresponding tree-oriented keyword search technique that has been used up to now.

Keywords: Keyword Search, XML, XML compression, DAG

1 Introduction

1.1 Motivation
Nowadays, increasing amounts of XML data, e.g. product catalogues and open linked data, are made publicly available also to the non-expert users. While query languages for XML data like e.g. XPath and XQuery are powerful search tools for expert users, the non-expert users who just want to retrieve information related to some given keywords do not have the technical knowledge to write XPath or XQuery search queries. Therefore, for these users which are the great majority of users, there exists a great demand for efficient keyword search for XML data, where a user can write his query as a list of keywords expressing his search query – similar as the user is used to do this, when he uses a search engine for searching information within the internet.

1.2 Contributions
Our paper presents DAG-Index, an approach to efficient keyword search within XML data that is based on a compressed keyword index. To the best of our knowledge, DAG-Index is the first approach that combines the following features and advantages:

- Prior to building of the index, DAG-Index transforms the document into a DAG (directed acyclic graph) which removes redundant sub-trees from the document, such that DAG-Index yields a search index that is smaller, and thus can be searched faster than an index for XML trees.
- DAG-Index indexes common sub-trees only once, such that repetitive sub-trees have to be searched only once, if all the searched keywords are found in the sub-tree.
- DAG-Index uses proxy nodes for grouping equal keywords within each repetitive sub-tree into a single keyword occurrence within a proxy node, which additionally speeds-up keyword search if not all searched keywords occur in a shared sub-tree.

1.3 Paper Organization
The paper is organized as follows: Section 2 summarizes the basic idea of keyword search in XML data, and introduces the example used in this paper for visualizing our ideas. Section 3 describes the fundamental concepts used by our approach. The fourth section outlines some of the experiments that compare our prototype with keyword search on a non-compressed index. Section 5 gives an overview of related work and is followed by the Summary and Conclusions.

2 Our Goal and the Paper’s Example

2.1 Goal of XML Keyword Search versus Text Keyword Search
Plain text keyword search is known to many users from using an internet search engine. The user provides a list of keywords, and the search engine returns a list of documents containing these keywords. The search results are ranked according to the calculated ‘quality’ which is based e.g. on the importance of keywords for the document or on the distance of the keywords within the document.

Similar to the idea of plain text keyword search is the idea of keyword search for semi-structured data as it is provided in form of XML documents. The user provides a list of keywords, and the search engine returns these sub-trees of the documents that contain all keywords. Similar as for the ranking of the plain text documents, in order to get high quality information, the user not only wants any sub-tree containing all keywords, but the user might want to retrieve all the ‘smallest sub-trees’ containing all keywords, i.e. those sub-trees that contain all keywords but do not contain a smaller sub-tree that also contains all keywords.
2.2 This Paper’s Example

The example used in this paper is an excerpt of an XML document representing soccer players of the German soccer Bundesliga together with the teams they played for. Our example contains information on the player named “Manuel Neuer” who played for the teams “FC Bayern München” and “FC Schalke 04” and the player named “Timo Hildebrand” who played for the teams “FC Schalke 04” and “Sporting Lissabon”.

Figure 1 shows the binary XML tree of this document where instead of edges from parent to children there exist edges from a node to its first-child and its next-sibling. The numbers in parentheses represent the preorder number of each node.

Whereas typically relational data does not contain redundancies, this is nearly unavoidable for XML data. As there is a many-to-many relationship from players to teams, the XML document used as example contains a redundancy: the node “team” with first-child “FC Schalke 04” exists twice. This redundancy is removed by DAG compression. The second occurrence of a node is replaced by a backpointer to the first occurrence. The DAG of the example is shown in Figure 2. The second occurrence of each of both nodes “team” and “FC Schalke 04” is removed, and the next-sibling pointer from the “team” node of player “Timo Hildebrand” is directed to the first occurrence of the nodes “team” and “FC Schalke 04”.

2.3 Standard XML Keyword Search on the Example Document

A user might ask for the keyword list (“FC Bayern München”, “Timo Hildebrand”) in order to check whether or not Hildebrand played for Bayern München. Although both keywords are contained within the document, both keywords are not very closely related. Similar, if a user asks for the keyword list (“FC Bayern München”, “player”) in order to find out, which players did play for Bayern München, the results might be the sub-tree containing the information on “Manuel Neuer” and the sub-tree containing the whole document (matches are e.g. the player-node of “Timo Hildebrand” and the “FC Bayern München”-node of the other player-sub-tree). Intuitively, only the first solution (“Manuel Neuer”) is a desired solution.

Therefore, besides the requirement that all keywords have to be contained within the sub-trees, an additional requirement is added to increase the quality of the search result. A solution must not contain another solution. This property is called the “shortest lowest common ancestor (SLCA)”. As the second solution with root node “bundesliga” contains the first one with the left player-node as root node, the second solution is not considered as a solution, as it does not fulfil the SLCA property.

3 XML Keyword Search With Uncompressed Index

3.1 Preliminaries

Our paper follows the idea of anchor-based keyword search as presented in (Sun et al., 2007). Before we explain our changes on the search index and on the search approach, we explain the ideas of anchor-based keyword search. The approach is based on inverted lists that store for each keyword occurring in the XML document, a list of references to the document nodes with the keyword as node label.
Let $V$ denote the set of nodes in an XML document tree $T$, and let $v.nl$ denote $v$'s node label for each node $v \in V$. We provide an index based on inverted lists that maps each node label $nl$ occurring in $T$ to an ordered list $L_{nl}$ of those nodes $v \in V$ in document order for which $v.nl=nl$. Given two nodes $v_1, v_2 \in V$, $v_1 <_p v_2$ denotes that $v_1$ is a preceding node of $v_2$ in document order in the document $T$; and $v_1 \leq _p v_2$ denotes that $v_1 < _p v_2$ or $v_1 = v_2$.

Let $K = \{w_1, \ldots, w_k\}$ denote a set of $k$ keywords given as input to the keyword search problem, i.e., the keyword search looks for all the smallest sub-trees of $T$ containing at least one node $v_i$ with label $v_i.nl= w_i$ for each $w_i \in K$.

A set $S=\{v_1, \ldots, v_i\} \subseteq V$ of nodes is defined to be a match for $K$ if for each keyword in $K$, $S$ contains exactly one node labeled with that keyword, i.e., if $|S| = |K|$ and for each $w_i \in K$, there is $v_i \in S$ such that $v_i.nl = w_i$.

We use $v_1 <_s v_2$ to denote that $v_1$ is a proper ancestor of $v_2$ in $T$, and $v_1 \leq _s v_2$ to denote that $v_1$ is an ancestor-or-self of $v_2$, i.e., $v_1 = v_2$ or $v_1 <_s v_2$. A node $v$ is a lowest common ancestor of $S \subseteq V$, if $v$ is a common ancestor of $S$, and there is no common ancestor $v_2 \in V$ of $S$ with $v_1 <_s v_2$. The function $lca(S,T)$ returns the LCA of $T$ in the set of nodes $S$ and returns null if $S$ is null. If $T$ is obvious, we write $lca(S)$ instead of $lca(S,T)$.

Furthermore, a node $v_1 \in V$ is a lowest common ancestor for $K$ (LCA for $K$) if $v_1$ is the lowest common ancestor node of at least one match $S$ for $K$. Moreover, $v_1$ is also the smallest lowest common ancestor (SLCA) for $K$ if no descendant $v_2$ of $v_1$ in $T$ is an LCA for any match for $K$.

For example, for the node set $S=\{4, 7\}$ in the document $T$ shown in Figure 1, $lca(S)$ is the node 2.

Consider a node $v$ and a set of nodes $S$. The function first($S$) returns that node $v'$ in $S$ with $v' \leq _p v_i$ for each $v_i \in S$. Similarly, the function last($S$) returns that node $v'$ in $S$ with $v_i \leq _p v'$ for each $v_i \in S$. Both functions return null if $S$ is null.

The function next($v$, $S$) returns the first node in $S$ that follows $v$ if it exists; otherwise, it returns null. The function pred($v$, $S$) returns the predecessor of $v$ in $S$, i.e., the last node in $S$ that precedes $v$ if it exists; otherwise, it returns null.

The function closest($v$, $S$) computes the closest node in $S$ to $v$ as follows:

\[
\text{closest}(v, S) = \{ \text{v, if } v \in S; \text{otherwise } \text{pred}(v, S), \text{if } lca(\{v, \text{next}(v, S)\}) < lca(\{v, \text{pred}(v, S)\}); \text{next}(v, S), \text{otherwise}. \}
\]

However, closest($S$, $S$) returns null if $v \notin S$ and both pred($v$, $S$) and next($v$, $S$) are null; and it returns the non-null value if $v \in S$ and exactly one of pred($v$, $S$) and next($v$, $S$) is null.

A match $S=\{v_1, \ldots, v_k\}$ is said to be anchored by a node $v_a \in S$ if for each $v_i \in S\{v_a\}$, $v_i = \text{closest}(v_a, L_i)$. $v_a$ is then called the anchor node of $S$.

### 3.2 XML Keyword Search

The anchor-based search on an XML document is then performed as follows (for details see (Sun et al., 2007)):

**Step 1: Chose an Anchor.** Initially chose that node $n$ as an anchor that occurs last in document order from all the first nodes of the inverted element lists $L_i$ of the keyword $w_i$, i.e. $n = \text{last(}\{\text{first}(L_i) \mid w_i \text{is a searched keyword}\})$.

If we consider a keyword search for $w_i$="FC Bayern München” and $w_2$="FC Schalke 04" in order to find out, when they did play against each other, we get the following two inverted element lists, where each list entry is the preorder position of the node contained in the list: $L_1=(7, 17, 33)$ and $L_2=(10, 26)$. Therefore, we chose the node $n=10$ with label “FC Schalke 04” as initial anchor.

**Step 2: Compute the SLCA Candidates.** Let $L_a$ be the inverted element list containing $n$. We compute an SLCA candidate for a list $M$ that contains all nodes $v_i = \text{closest}(n, L_a)$.

For this purpose, in each inverted element list $L_i \neq L_a$ of keyword $w_i$, we chose $v_i = \text{pred}(n, L_i)$ as current node. The node $n$ and all these nodes $v_i$ form the initial list $M$ containing the match being currently regarded.

Considering our example, $M=[\{7, 10\}]$.

Next, we repetitively check, whether $v_i = \text{first}(M)$, i.e. the first node of the list $M$, where $v_i = \text{pred}(n, L_i)$ could be replaced by a node $v'_i = \text{next}(n, L_i)$ being closer to $n$. As long as such a node $v'_i$ is found for the currently first node $v_i$ of $M$, $v_i$ is substituted with the closer node $v'_i$ until no replacement is possible anymore. Then, the next SLCA candidate is $A=lca(\{v_i, \text{last}(M)\})$.

In our example, we check whether $v_i$=7 could be replaced by the node $v'_i$=17.

The replacement check is to check whether a node $v'_i = \text{next}(n, L_i)$ exists, and if so, to compute $A = lca(\{v_i, \text{last}(M)\})$ and to check whether $A < v'_i$. In our example, $lca(\{7, 10\})=2$, $v'_i=17$, and $17 < v'_i=2$, i.e. 2 is replaced by 17.

Furthermore, whenever replacing a node $v_i$ by $v'_i$, we have to chose $v'_i$ as the next anchor if the following holds: For each keyword $w_i$ (i.e.), there exists a node that occurs after the old anchor $n$ and before $v_i$ in document order.

In our example, we have to replace the old anchor by the new anchor $n'=v'_i=17$.

Whenever an SLCA candidate $A$ has been computed, add $A$ to the set $C$ of result candidates. Furthermore, let $N=|n_i | n_i=\text{next}(A, L_a)|$ be the set of all the nodes that occur after $A$ in document order in one of the lists $L_i$ of nodes that have the keyword $w_i$ as their label, and repeat Step 2 taking $n=\text{last}(N)$ as new anchor until the end of the XML tree has been reached for at least one $i$, i.e., next($A, L_a)$ does not exist.

In our example, we compute the new SLCA candidate for nodes 10 and 17, which is the node 8.

As node 33 is not a descendant of node 8, we add node 8 to the set of result candidates and continue the computation with node 33 as the new anchor resulting in node 24 as result candidate.
Step 3: Compute the Result Set from the Set C of Candidates. Remove all nodes \( ca \in C \) from the set of candidates \( C \) for which a node \( cd \in C \) exists with \( ca < cd \). All remaining nodes form the result set \( R \).

As nodes 8 and 24 of our example are not in ancestor-descendant-relationship, we do not have to remove any node and return the set of candidates as final result.

4 XML Keyword Search Based on a Compressed Index

Instead of computing the inverted elements lists \( L_i \) for each keyword \( k_i \) based on the XML document (i.e., one list entry into the lists \( L_i \) for each XML document node that has a label \( k_i \)), we compute the keyword list based on the compressed DAG of the XML document. Besides keeping the index small, the goal of using DAG compression is to search shared sub-trees only once, and thereby achieve a faster search speed.

4.1 Compressed Index

Prior to computing the index, we transform the XML document into its minimal DAG by replacing each repeated occurrence of a sub-tree with a pointer to the sub-tree’s first occurrence.

Similarly as for the uncompressed index, our compressed index consists of inverted element lists \( L_i \) for each potential keyword \( k_i \) that occurs as an element label or as a text node within the DAG. We do a bottom-up search for DAG nodes with multiple incoming edges and split the DAG into multiple sub-DAGs as follows: Whenever a node \( v \) of the DAG \( D \) has more than one incoming edge, i.e., \( v \) has the incoming edges \( e_{v_1}, \ldots, e_{v_n} \), we remove \( v \) from \( D \) and start a new sub-DAG \( D_v \), where \( D_v \) is a copy of \( D \) with all nodes not being a descendant-or-self of \( v \) in \( D \) being removed from \( D_v \) and with all dangling edges being removed, such that \( v \) is the root node of \( D_v \). Let \( L_v \) be the set of labels occurring in \( D_v \). Each edge \( e_{v_j} \) gets a new (virtual) target node \( v_j \), called proxy node of \( v \), and for each \( k_j \in L_v \), \( v_j \) is added to the inverted element list \( L_i \) representing all the occurrences of keyword \( k_j \) in \( D \).

Additionally, the information that \( v_j \) is a proxy node for \( v \) is stored in a table of proxy references where each node \( v_j \) has a reference to the root node \( v \) of \( D_v \).

Figure 3 shows the document of our example where the DAG is split into two DAGs connected by the proxy nodes p1 and p2 (represented by white rectangles) and their references to the common team node (1’) which is the second DAG’s root node.

4.2 Keyword Search on the Compressed Index

Keyword search on the DAG-compressed index works similar to keyword search on the uncompressed index, with the following differences: Due to the introduction of proxy nodes that represent multiple keywords occurring in a sub-DAG, the same proxy node-ID may occur in multiple inverted element lists, and the same proxy node-ID may occur multiple times within the currently considered list \( M \) of actual nodes. Whenever during the computation of \( M \), all elements of \( M \) contain the same proxy node \( v_j \), where \( v_j \) refers to the root node \( v \) of a sub-DAG \( D_v \), the complete match is contained in \( D_v \) or in a sub-DAG of \( D_v \). In this case, first, we remove a possible SLCA candidate \( C \) in \( D \), second, if \( C \leq a v_j \), we perform the keyword search in \( D_v \), and third, we start a new keyword search within \( D \) with a new anchor among the nodes after \( v_j \), i.e., we continue after we have increased the pointer positions in all inverted keyword lists of \( D \) to next(\( v_j \)). In this case, we have the advantage of computing the SLCA s within \( D_v \) for all shared sub-trees represented by \( D_v \) only once. Whenever this optimization is possible, we yield a faster search compared to computing all these solutions individually.

In the example of Figure 3, the first anchor node is the proxy node p1 and \( v_i \) is the node 7. Then, during the computation, we set \( v_i = p1 \) (similar as it was the case for the non-compressed index). As now all nodes in \( M \) represent the same proxy node \( p1 \), we recursively start a new search at the node (1’), i.e. at the root node of the second DAG. Within this DAG, we find that the nodes with preorder positions 3’ and 10’ within the sub-DAG are SLCA s. Later, the second anchor node found in the first DAG is the proxy node p2 and a corresponding node...
vi is the same proxy node p2. As p2 also refers to node (1’) which now has already been investigated, no new search starting in (1’) is required. Thereby, we have computed the SLCAs for both shared sub-trees only once – whereas, when using the non-compressed XML tree index, we had to compute the SLCAs represented by 3’ and 10’ twice, i.e. for both sub-trees.

5 Evaluation of Prototype Implementation

5.1 Evaluation Environment

We have implemented a prototype of our approach using Java 1.6.0. We compared our prototype using the compressed index with a similar implementation of the anchor-based keyword search on XML trees as described in Chapter 3.

As test documents, we used DISCOGS (http://www.discogs.com/data/), a discography database containing information on the releases, including information on the artist, the style, the genre, the originating country, the release date and comments.

We split the database into several chunks yielding documents starting from 50,000 releases (D50), having a file size of 42MB up to 350,000 releases (D350), having a file size of 271MB.

5.2 Evaluation Results

![Figure 4: Searching the 4 most frequent keywords using scaling document sizes](image)

![Figure 5: Keyword query accessing frequent text nodes using scaling numbers of keywords](image)

In a first series of measurements, we assumed a sort of worst-case scenario for both approaches. We computed the 10 most frequent keywords contained within each document, and we searched for the n=1,2,...,10 most frequent keywords. This means that during the evaluation, the longest possible keyword lists have to be combined, yielding a sort of worst-case scenario for both approaches.

Figure 4 shows the results for the 4 most frequent keywords for documents with increasing file size. As we can see, for some documents the anchor-based approach (XML) was faster, whereas for other documents the DAG-based approach was faster. But neither approach completely outperforms the other one.

Figure 5 shows the results for document D250, containing 250,000 releases, and for an increasing number of keywords. We can see that for smaller numbers of keywords (3-6) the anchor-based approach is predominant, whereas for larger number of keywords (>7), i.e., for more complex queries, the DAG-based approach outperforms the anchor-based approach. i.e., the increase of the search times for increasing file size is much smaller for the DAG-based approach than for the anchor-based approach.

These queries form a worst-case scenario, but we consider them to occur only less frequently than the queries of the following experiments, as these queries search for high-level tag names only and do not search for XML text nodes. In a second series of measurements, we used a query, asking for the most frequent text node and its parent label (i.e., for all releases of genre “Electronic”).

![Figure 6: Keyword query accessing frequent text nodes using scaling document sizes](image)

Figure 6 shows the results for increasing file sizes. It can be seen that for this query, the DAG-based approach outperforms the anchor-based approach. Only for one document (D250), the DAG-based approach needs as long as the anchor-based approach, but on average the DAG-based approach takes only 88% of the computation time of the anchor-based approach (at least 75% within our measurements).

To summarize, in our measurements for worst-case scenarios, the DAG-based approach performs approximately as good as the anchor-based approach, whereas for keyword queries, which we expect to occur more frequently, the DAG-based approach is superior and outperforms the anchor-based approach.

6 Related Works

There exist several approaches that address the problem of keyword search in XML.
These approaches can be roughly divided into two categories: approaches that examine the semantics of the queries in order to achieve query results of higher relevance on the one hand and approaches that concentrate on a higher performance for the computation of the set of query results on the other hand.

Within the first category, (Guo et al., 2003) not only focus on an efficient algorithm for keyword search based on inverted element lists, but they aim to rank the search results in such a way, that the user gets the (probably) most interesting results prior to the other results. SUITS (Zhou et al., 2008) is a heuristic-based approach, and the approach presented in (Petkova et al., 2009) uses probabilistic scoring to rank the query results. In order to enhance the usability, (Li et al., 2010) propose an approach on how to group the query results by category.

Within the second category (efficient result computation) most approaches are based on finding a set of (S)LCA nodes for all matches of a given keyword list.

Early approaches were computing the LCA for a set of given keywords on the fly. (Schmidt et al., 2001) propose the meet-operator that computes the LCA for a pair of nodes that match two query strings without requiring additional knowledge on the document structure from the user.

In contrast, recent approaches try to enhance the query performance by using a pre-computed index. (Florescu et al., 2000) propose an extension of the XML query language XML-QL by keyword search. In order to speed-up the keyword search, they compute the so-called “inverted file” for the XML document – a set of inverted element lists – and store the contents within a relational database.

(Li et al., 2004) present two approaches to compute the Meaningful Lowest Common Ancestor (MLCA), a concept similar to the SLCA considered in our approach. Their first approach allows computing the MLCA with the help of standard XQuery operations, whereas their second approach is a more efficient approach that is based on a stack-based algorithm for structural joins.

XKSearch (Xu & Papakonstantinou, 2005) is another stack-based approach to compute the LCA. For each keyword k, they store two lists: one inverted element list Lk containing all nodes with label k, and an ancestor list Ak containing all nodes that have a descendant with label k. They process the nodes bottom-up and store the nodes that have not yet been completely examined on the stack. Whenever a node is found, the descendants of which have all the search keywords as labels, it is returned as a result. In this case, all its ancestors are removed from the stack, as they cannot form a result anymore.

JDeweyJoin (Chen & Papakonstantinou, 2010) returns the top-k most relevant results. They compute the results bottom-up by computing a kind of join on the list of DeweyIDs of the nodes in the inverted element list. Whenever they find a prefix that is contained in all relevant element lists, the node with this prefix as ID is a result candidate. In addition they use a weight function to sort the list entries in such a way that they can stop the computation after k results, returning the top-k most relevant results.

(Zhou et al., 2012) present a more efficient, but more space-consuming approach. The elements of their inverted element lists do not only contain the nodes that have the keyword as label, but also contain all ancestor-nodes of these nodes. Therefore, they can compute the SLCAs by intersecting the inverted element lists with the list of keywords and by finally removing each result candidate, the descendant of which is another result candidate.

Like the contributions of the second category, our paper focuses on efficient result computation. It follows the anchor-based approach as it was presented in (Sun et al., 2007). However, different from all other contributions, instead of computing an XML-index, we compute a DAG-Index. This helps to compute several keyword search results in parallel, and thereby speeds-up the SLCA computation. To the best of our knowledge, DAG-Index is the first approach that improves keyword search by using XML compression before computing the search index.

7 Summary and Conclusions
Keyword search is of increasing interest for searching relevant data within large XML document collections, especially for the huge majority of non-expert users. Due to the increasing amount of publicly available data in the XML format, there is an increasing interest in fast keyword search techniques. We have presented DAG-Index, an indexing and keyword search strategy for large XML documents that allows compressing an XML tree and the search index in such a way that common sub-trees have to be indexed only once. As a consequence, a repeated keyword search within a repeated sub-tree can be avoided. Therefore, we consider our DAG-Index-based keyword search to be a significant contribution to improve the search performance especially for the majority of the non-expert users.

8 References


Scalability in Recursively Stored Delta Compressed Collections of Files

Angelos Molfetas\textsuperscript{1}  \hspace{1cm}  Anthony Wirth\textsuperscript{1}  \hspace{1cm}  Justin Zobel\textsuperscript{1}

\textsuperscript{1}Department of Computing and Information Systems
The University of Melbourne

Email: \{angelos.molfetas, awirth, jzobel\}@unimelb.edu.au

Abstract

The archiving and maintenance of vast quantities of data is a key challenge for the current use of information technology. When storing large repositories, possibly mirrored at multiple sites, an archiving system aims to reduce both storage and transmission costs. Delta compression is a key component of many archiving and backup systems. A file may be stored succinctly as a sequence of references to other files in the collection, establishing a dependency relationship between files. On the one hand, exploiting large dependency chains provides excellent compression. On the other hand, if a file is stored compactly, so that it depends on hundreds of other files, then retrieving it from the archive may be very time and resource consuming.

This paper assesses the scalability of delta compression of typical data collections. We use experiments to model and examine the dependency relationship, and quantify the cost of full use of dependencies. We propose strategies to reduce dependencies and yet retain highly effective compression.

Keywords: Differential compression, repository compression, scalability, delta files.

1 Introduction

The proliferation of Web services and cloud computing necessitates centrally managed and distributed stored repositories of large collections of files. Examples of such large data collections are versioned textual content (e.g. in the case of wiki sites, such as Wikipedia\textsuperscript{1}), emails (in the case of Web-based email systems, such as gmail\textsuperscript{2}), and source code (in the case of Web-enabled repository sites, such as Github\textsuperscript{3} and Bitbucket\textsuperscript{4}).

Repositories of source code and executables are mirrored between countries; collections, such as genetic databases, are distributed to scientific institutions; companies duplicate data so that the people at each site have a local copy; applications such as search engines maintain their indexes at multiple regional data centres.

In enabling these Web-based applications, the repositories should be replicated, or partially replicated, in an efficient and scalable manner. Furthermore, these large datasets often have large amounts of redundancy, allowing for efficient transport and storage of data. The aim of scalable and efficient transport and storage raises numerous relevant questions. For example, given a repository and a query file, can we find (a specified number of) files in the repository that are the closest to the query, assuming some definition of file similarity? Or, given two repositories, can we estimate the communication required to compress the information required to synchronize the repositories? In Web scale systems, these types of questions are very difficult to answer in a manner that scales with the increasing quantities of data being produced.

The cost of sending, say, an executable to a machine over the Internet can be greatly reduced if it is possible to first identify large parts of the executable that have already been delivered as components of other programs.

Closely related to differential compression is data deduplication. A simple example of deduplication is the removal of an identified duplicate file from a collection, replacing it with a hard link, with the same name, pointing to the inode for the original file. Duplicate files are usually identified through some hashing technique.

Companies offering mass storage often deduplicating their data. For example, the online file hosting service Dropbox identifies identical files or pieces of files, and stores a single copy\textsuperscript{5} (presumably before initiating replication for redundancy). Though it is similar to deduplication, in contrast, differential compression may in a single file object interleave references to duplicate data with non-duplicate data.

Many organizations keep a record of a large-scale Web crawl for search and mining. Plausibly, several organizations may share the results of several crawls (Suel et al. 2004). The task of our algorithms, therefore, would be to ensure that the updates that each crawl finds are transmitted to the other sites. The Stanford WebBase project is an example of this approach (Hirai et al. 2000).

Copyright ©2014, Australian Computer Society, Inc. This paper appeared at the The Second Australasian Web Conference, AWC 2014, Auckland, New Zealand, January 2014. Conferences in Research and Practice in Information Technology (CRPIT), Vol. 155, S. Cranefield, A. Trotman, J. Yang, Ed. Reproduction for academic, not-for-profit purposes permitted provided this text is included.
Differential Compression  Differential compression is a method for concisely encoding files by taking advantage of their similarity to other files (Ajtai et al. 2002). It is a particularly effective, and a commonly applied, technique for the transfer of versioned data, for example by synchronization tools such as rsync\textsuperscript{6} (Tridgell & Mackerras 1996). Differential compression has been used in a cache-based technique for optimized Web transfers distribution of content to Web clients (Chan & Woo 1999), for IP-level network requests (Spring & Wetherall 2000), and for Web traffic (Mogul et al. 1997).

Simple two-file differential compression involves encoding a target file with respect to a reference file. The reference is typically (but not necessarily) an older version of the target file. This process looks for matching strings between the target and reference files and produces a delta file. When decoded in the presence of a reference file, the delta file spawns the target file. This delta file is composed of two types of instructions: copies that specify that a matching strings be copied from the reference file, and adds that specify verbatim a string (that was not found in the reference file) and should be spliced into the reconstructed target file. A standard approach to construct delta files is the Bentley & McIlroy (1999) hash-table dictionary scheme for finding long common strings.

In addition to encoding a single file against another file, differential compression can also be used to compress a collection of documents. This differs from the common combine and compress approach, which merges all files into a single file, with tar or similar, and then compresses the resultant file with a compression utility, such as gzip\textsuperscript{7} or 7zip\textsuperscript{8}. When compressing a collection using differential compression, the key objective is to exploit repeating occurrences in different files while at the same time being able to access them atomically, without having to decompress the whole collection (Peel et al. 2011). Though differential compression can make the most of limited storage space, it also poses scalability problems. Principally, a delta file could refer to several reference files, which themselves are stored as deltas, in turn linking to other reference files, and so forth. Therefore, to restore a single file to its original target form, the system might need to resolve a large number of dependencies, and thus access many many files.

Related Work  In the context that nearby files may be unrelated, Bhagwat et al. (2009) exploit file similarity, rather than locality, to build a scalable file-backup system. Here, chunks of size 4 to 8 kB are cryptographically hashed to detect inter-file duplication. Echoing some of the considerations in this paper, Min et al. (2011) consider how to divide an index of fingerprints, and which replacement policy to invoke, for the backup of multimedia files.

PRESIDIO is storage framework for immutable archives (You et al. 2011). It incorporates multiple different compression techniques, including chunking and delta compression, and develops alternatives to standard clustering techniques. You et al. explore the impact of delta chain length, one of the central issues in scalability, and a focus of this paper. They show that long delta chains, in which delta files refer to other delta files, provide massive increases in compression effectiveness. However, as we describe in detail in Section 3, resolving long delta chains burdens disk and CPU heavily. Different from those in this paper, You et al. present some strategies for reducing delta chain length.

Outline  This paper continues as follows. Section 2 presents a taxonomy of the different ways that differential compression can be applied. Section 3 discusses scalability problems of some differential compression schemes and some known solutions. Experiments demonstrating the problems with a simple differential compression approach follow in Section 4. Resolutions to these, as well as further experiments, appear in Section 5. The paper concludes in Section 6.

2 Types of Differential Compression  Differential compression schemes can be classified into four broad categories: two-file; serial; collection-based, single dependency; and collection-based, multiple dependencies.

Two-File This scheme is the simplest, involving only a single delta file, which encodes a target file with respect to a single reference file. The target file can be recreated from the delta file in the presence of the reference file (Figure 1). This scheme is a good choice when one wants to transfer a large target file to a remote location containing a copy of the reference file that is very similar to the target file; in this situation, one can send a much smaller delta file incorporating the differences between the two files.

Examples of tools of this kind are xdelta\textsuperscript{9}, vdelta (MacDonald 2000), and zdelta (Trendafilov et al. 2002).

Serial Here, delta files are encoded sequentially; most of the reference files are themselves stored as delta files, differenting from some previous version. Thus, the process of recovering a target could involve decoding multiple reference files (Figure 2). Serial differential compression can be implemented by repeated application of a two-file tool. For example, xdelta could recreate a target file from a delta file, then the

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{two-file-compression.png}
\caption{Two-file differential compression.}
\end{figure}

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{serial-compression.png}
\caption{Serial differential compression.}
\end{figure}
recreated target file becomes the reference file for the next delta file.

Some types of Version Control System (VCS), such as SCCS (Rochkind 1975), implement a serial differential compression scheme.

Collection-Based, Single Dependency Collection-based differential compression is applicable to collections comprised of files that are related or have similar content, but are not sequences of versions. When a file is added, it is encoded with respect to another file in the collection, which in turn may be a delta file encoded with respect to some other file in the collection. This approach is very similar to the serial approach described in Section 2, but differs in that branching between delta files can occur (Figure 3).

Nevertheless, each file depends on a single reference. With single dependencies in collection-based differential compression, one can (again) apply a two-file tool repeatedly to build a compressed collection.

The key challenge with type of compression is to identify sufficiently similar pairs of files to differentially compress. Manber (1994) presented a method for finding similar files in a large file system, and suggested that this method could be used for data compression. Following this work, Douglas & Iyengar (2003) showed how to apply hash (or fingerprint) techniques to detect resemblance amongst a collection of files of disparate kinds in order to differentially compress them. Ouyang et al. (2002) clustered files by similarity to optimize branching, which are then compressed using the two-file zdelta utility.

Collection-Based, Multiple Dependencies Finally, a file repository can also be stored as a collection of delta files, each of which has multiple dependencies (Figure 4).

In contrast to encoding with single dependencies, a two-file delta coding tool is not sufficient for this task. Moreover, from multiple dependencies arise significant scalability concerns. These are the central focus of this paper.

3 Scalability Issues

In serial differential compression, when encoding a new delta file, there is no need to decide which existing delta file should be chosen as a reference. As a new version is serially added to the collection, the previous version delta is simply selected as the reference.

The problem with this technique is that decoding a delta file requires decoding all ancestor delta files. In the context of a VCS that employs this scheme, all revisions of a file must be decoded in order to read the latest revision. Consequently, the time needed to decode a file can increase significantly as that file’s history grows. Some VCS systems, such as SVN’s BDB back-end alleviate this problem by instead having the latest version of a file stored as the reference file, with deltas for previous revisions. Generally, it is less likely that a developer requires a revision the older it becomes. The weakness of this approach is that each time a new revision is created, all the previous revisions would have be recalculated.

SVN solves the problem of decoding multiple delta files by introducing skip deltas. By skipping some revisions, the skip delta technique attempts to reduce the number of dependency delta files to be accessed. It determines the predecessor revision by taking the revision number in binary representation, and flips the rightmost bit that has a value of 1. As Figure 5 shows, this scheme provides pathways that require fewer revisions to be decoded: shorter chains. When there are N revisions, this technique limits the maximum number of necessary decodings of a delta file to \( \log_2 N \); there is a space penalty of \( O(\log N) \), but an \( O(N/\log N) \)-time benefit.

Finding the Optimum Reference File The main challenge in single-dependency collection-based differential compression is finding an appropriate (existing) delta file to act as reference file to a newly added delta file. Ideally, one wants to choose a delta file that (in

---

Figure 2: Serial differential compression.

Figure 3: Collection-based differential compression, single dependencies.

Figure 4: Collection-based differential compression, many dependencies.
its decoded form) is the most similar to the target file being added to the collection.

Ouyang et al. (2002) observed that finding the optimal delta encodings between files in a collection, with single dependencies, can be reduced to finding the maximum-weight branching in a (edge-weighted) directed graph. The graph models the situation thus.

1. A node of the graph represents a target file that is to be added to the collection.
2. The weight of an edge from node $r$ to node $t$ is the compression saving obtained when target file $t$ is delta encoded with respect to reference file $r$.
3. There is a null node, which has no edges directed towards it.
4. The weight for the edge originating from the null node to another node $f$ represents the compression benefit obtained if the file $f$ were compressed with respect to itself.

Figure 6 shows an illustration of the possible pairwise compression arrangements.

**Figure 6** Directed weighted graph showing compression savings between pairwise delta encodings (based on an illustration by Ouyang et al. (2002)).

Finding an optimal branching in the directed graph provides the optimal compression possible from a two-file tool. However, as this approach does not scale to large collections of files, Ouyang et al. (2002) experimentally compared a number of file-clustering heuristics. They achieved significant compression improvements compared to concatenating the files in the collection and then applying the gzip tool.

**Recursive Dependencies** An example of a proposed system that has *delta files* with multiple dependencies is Chan & Woo’s (1999) cache-based technique for optimized Web transfers. This system works by having the server encode the target file according to a selected set of reference files to generate a *delta file*; this *delta* is then transmitted to the client, which has copies of the *reference files* in its cache. The system scales well for two reasons: first, the number of *reference files* is bounded by the selection heuristic; second, the *delta file* is only for transfer, and is decoded immediately at the client. The *reference files* are not stored as *delta files*, and there is no need to recursively decode all dependent *delta files* to access the file transferred.

Delta decoding scalability becomes an issue when a repository both stores a collection as *delta files*, and uses these *delta files* as *reference files* for files that are added in encoded form. Consequently, there are long chains of dependencies, which (only) grow as more files arrive in the collection.

**Multiple References** When the differential compression encoding scheme allows for a *delta file* to have multiple *reference files*, the recursive dependency problem becomes infeasible except in cases where one wishes to decode the entire collection of files at once. The multiple files introduce a multiplier effect for each level of encoding recursion. We explore this further in the next section.

**4 Experimental Illustration**

We explore in detail experimentally, and propose solutions to, some of the scalability issues raised in the previous section. We ascertain the scalability problems in multiple-dependency collection-based differential compression. The experiment incrementally builds a differentially compressed collection where all files are stored as *delta files*.

First, a hash-based dictionary is instantiated, where each entry stores a file identifier and offset. As each file is added to the collection, it is assigned a unique file identifier and then scanned and fingerprints are generated using the Karp & Rabin (1987) rolling hashing method. Each hash is then added into the hash table, such that it is associated with its corresponding file identifier and position within the file.

There are two approaches to storing hashes in the dictionary. The first stores all hashes, while the second stores only each sequential non-overlapping hash in the dictionary, disregarding the rest (Bentley & McIlroy 1999). The latter is a simple way to reduce the size of the dictionary, though compromises the detection of smaller strings. This study takes the first approach.

When matches are identified, the matching substrings are replaced with *copy instructions* that specify a file identifier, an offset to begin copying and the number of bytes to copy. All other strings, those not matched, are ‘encoded’ as *add instructions*, which include the unmatched strings verbatim. The algorithm has a recursive decoding function that reads each *delta file* referenced by each *copy instruction*, and
decoding recursively until no more copy instructions are found. Finally, once a target file is successfully added to the collection and stored as a delta file, as a validation step it is once again extracted by converting it into its original expanded form. The experiment records the direct dependencies that each delta file has, that is which reference files it refers to in its copy instructions. The experiment then determines the total number of dependencies, direct or recursive, required to decode the delta file.

Datasets The Enron Corpus dataset has about 500,000 emails from Enron’s senior management\(^1\) (Klimt & Yang 2004). It was released to the public by the United States Federal Energy Regulatory Commission.

The CSDMC dataset was designed for spam classification experiments. Its curators partitioned it into a testing and training set\(^2\).

We also test against development repositories that are composed principally of source code, as well as some technical documentation. The examples are the Linux kernel, Python and Ruby interpreters, and the matplotlib plotting package (Hunter 2007), which incidentally produced the plots in this paper. Table 1 gives an overview of the datasets.

It should be noted that these datasets are treated in the experiments as collections of similar independent files. That is, email headers and file names are not used to infer relationships between files in order to choose reference files. This is because the paper is investigating multiple dependency collection-based repositories for similar files, rather than for example, compressing the datasets as a set of serial or single dependency collection-based repositories.

### 4.1 Results

Figure 7 shows the average number of direct dependencies of delta files that are added into the email repositories. The number of dependencies rapidly increases as the hash table is filled; then, as the hash table is saturated, the average number of dependencies levels off at under 4.5 for all three datasets.

This number of directly connected files appears not to be very high. However, due to the multiplicative effect at every level of recursion, it means that opening a single file requires a very large number of other files be opened. Worse still, this number of total dependencies constantly increases, and seems to be linear in the size of the collection (Figure 8).

This linear relationship is acceptable if one intends to decompress the entire collection in insertion order. However, it negates the advantage that collection-based differential compression has over combine and compress methods. To access a single file, one needs to decompress all, or at least a very large portion, of the collection.

The linear blowout in dependence is not intrinsically due to newer delta files having a large number of files to choose to refer to. It arises, as a consequence of the design of the fingerprint hash table, which favors dependencies to recent files. Figures 9a and 9b demonstrate that, as files are inserted, newer files replace the older ones as reference files.

So far, we have examined only the properties of collections of emails. On the source file datasets, the dependency structure seems far less stable (Figures 10a and 10b). The Ruby dataset is similar to the email collection, though with fewer than two direct dependencies on average per file. On the other hand, at various points, as files are added, the number of dependencies in the Linux kernel collection seems to increase quite suddenly, possibly due to the large variation in file type.

The matplotlib collection appears to have very high direct dependency initially, then tends towards

Table 1: Summary of datasets

<table>
<thead>
<tr>
<th>Dataset</th>
<th># Files</th>
<th>Total Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSDMC Training</td>
<td>4327</td>
<td>27,913,226</td>
</tr>
<tr>
<td>CSDMC Testing</td>
<td>4292</td>
<td>27,265,343</td>
</tr>
<tr>
<td>Enron Corpus</td>
<td>14,357</td>
<td>41,084,874</td>
</tr>
<tr>
<td>Linux Kernel 3.9.2</td>
<td>42,412</td>
<td>479,610,289</td>
</tr>
<tr>
<td>Python 3.3.2</td>
<td>3785</td>
<td>65,582,523</td>
</tr>
<tr>
<td>Ruby 2.0.0-p195</td>
<td>4079</td>
<td>63,919,656</td>
</tr>
<tr>
<td>Matplotlib 1.2.1</td>
<td>4122</td>
<td>62,104,168</td>
</tr>
</tbody>
</table>

\(^1\)www.cs.cmu.edu/~enron/

\(^2\)csmining.org
a count more like the email collections. The instability in the increase of dependencies in the source datasets suggests that more dissimilar datasets are advantageous for multiple dependency collection-based repositories in terms of reducing the large dependency chains, though this would come with a lower overall compression trade-off.

Though we leave it for future study, it would be interesting to investigate how the dependency relationship varies according to the order in which the files are added to the collection. For instance, is a randomized order helpful?

5 Resolving the Scalability Issue

Increasing Hashing Block Size To reduce the number of dependencies, our first strategy is increasing the fingerprinting block size. This increase forces the algorithm to find fewer direct dependencies and larger matches. Figures 11a and 11b show the results of changing from 32 to 64 and 128-byte blocks.

However, this hardly reduces the overall number of dependencies (Figures 12a and 12b). For the CSDMC datasets, it reduces the rate of increase, but the problem still persists. Despite the block size increase, added files still tend to refer to recent files (Figures 13a and 13b).

Increasing Size of Hash Table The next strategy to reduce dependency chains is to increase the size of the hash table itself. A larger hash table is more likely to have older entries: this encourages references to older files, requiring a shorter chain of dependencies. So we repeated the experiment with 32-byte hashes, but this time with a much larger hash table (2,097,143 vs 32,749 bytes).

Unfortunately, there is still a linear increase in the total number of dependencies (Figure 14a). Even though the differential compression algorithm now builds references to much ‘earlier’ files (Figure 14b), it seems to find direct dependencies to more files (compare Figure 14c with Figure 7).

Furthermore, files inserted in close proximity to each other are generally more likely to be similar in content, and this effect cannot be neutralized by larger hash tables.

Hash Table Replacement Policy So far, the hash table policy has been that newer entries displace older entries. This encourages delta files to be encoded against more recently added files, which increases the number of dependencies. Instead, in the next experiment, the differential compression algorithm will favor older hash table entries.

Figure 15a shows the number of direct dependencies for the Enron dataset. Figure 15b shows that, in contrast to previous experiments, the curve
for the total number of dependencies becomes sub-linear. Finally, scalability has improved significantly. Newer files are linking directly to very old files (Figure 15c), thus significantly reducing chain length.

The point at which the curves start flattening depends on dictionary size. In this experiment, the hash table dictionary had 131,071 entries.

Though this policy change appears to resolve the issue of scalability, it does so only by preventing the algorithm from incorporating new information. In some sense, the hash table for the file collection has become stale.

A more sophisticated alternative would cluster the files based on similarity and allocate a separate hash tables per group. This would constrain the number of files each file could refer to, but at the same time ensure that it refers to the most relevant content. It seems more appealing than simply restricting the compression process to older content.

6 Conclusion

By taking advantage of high levels of redundancy that exist between files, and by storing files in their collections as delta files, repositories that incorporate differential compression store data efficiently. There are two key approaches. First, files can be compressed in a pairwise manner using a two-file compression tool. The challenge with this approach is ascertaining a method that can find an appropriately similar file in the collection. This method should ideally take constant or sub-linear time in the number of files in the collection.

The second approach involves a hash-table dictionary to find components of the new file that match components of files already in the collection. Once added, the new file takes the form of a delta file that ‘links’ to multiple reference files. When a delta file has multiple dependencies, decompressing it might require recursively decoding many files. Worse still, the number of dependent delta files increases linearly with the number of files in the collection. The same problem exists with serially encoded differentially compressed collections, such as VCS systems; in this case, however, the decoding recursion is limited to revisions of one file. Similarly, with differentially compressed collections of delta files with single dependencies, as files are encoded against similar reference files, the problem would be constrained to each cluster of files.

For collection-based compression with multiple reference files, the compression algorithm has no constraints, and consequently the number of dependencies increases with the size of the collection.

If one wishes to decompress the entire the whole collection, as is the case with combine and compress methods, then this high level of delta file dependency is not a problem. However, in the context of Web-
based systems needing to access files within compressed collections, except when data is no longer accessed and archived, these high levels of dependencies between delta files are infeasible.

As we have shown, the number of dependencies can be constrained by increasing the size of the dictionary and applying a selection strategy that favors older entries over new. This, however, means that newly added files may not be future reference file candidates.

Collections composed of heterogeneous types of files, when stored in Multiple dependency collection-based repositories, generate smaller chains of file dependencies. In such cases, however, there would also be a reduced compression trade-off.

As future work, we recommend a clustering algorithm that applies a separate hash table dictionary to each cluster, thus constraining the dependency to that cluster of delta files.

References


Figure 13: Oldest direct dependency for larger hash blocks.

Figure 14: Results for larger hash tables.
Figure 15: Results for change in hash-table policy.

(a) Number of direct dependencies.

(b) Total number of dependencies.

(c) Oldest dependency.
Extracting Crime Information from Online Newspaper Articles

Rexy Arulanandam  Bastin Tony Roy Savarimuthu  Maryam A. Purvis

Department of Information Science
University of Otago,
PO Box 56, Dunedin, New Zealand 9016,
Email: (rexy.arulanandam,tony.savarimuthu,maryam.purvis)@otago.ac.nz

Abstract

Information extraction is the task of extracting relevant information from unstructured data. This paper aims to ‘mine’ (or extract) crime information from online newspaper articles and make this information available to the public. Baring few, many countries that possess this information do not make them available to their citizens. So, this paper focuses on automatic extraction of public yet ‘hidden’ information available in newspaper articles and make it available to the general public. In order to demonstrate the feasibility of such an approach, this paper focuses on one type of crime, the theft crime. This work demonstrates how theft-related information can be extracted from newspaper articles from three different countries. The system employs Named Entity Recognition (NER) algorithms to identify locations in sentences. However, not all the locations reported in the article are crime locations. So, it employs Conditional Random Field (CRF), a machine learning approach to classify whether a sentence in an article is a crime location sentence or not. This work compares the performance of four different NERs in the context of identifying locations and their subsequent impact in classifying a sentence as a ‘crime location’ sentence. It investigates whether a CRF-based classifier model that is trained to identify crime locations from a set of articles can be used to identify articles from another newspaper in the same country (New Zealand). Also, it compares the accuracy of identifying crime location sentences using the developed model in newspapers from two other countries (Australia and India).

Keywords: crime mining, information extraction from newspapers, machine learning

1 Introduction

With the advent of the Internet, huge volumes of data (also called ‘big data’) are available online. Electronic newspapers are increasingly being read by users from anywhere, anytime. In New Zealand alone there are about 20 daily newspapers, and many of them make an electronic version available online. Newspapers are a source of (mostly) authentic and timely information. There is a large amount of information available in newspaper articles. For example, newspaper articles contain information about crimes, accidents, politics, cultural events and sports events.

Even though valuable information is available in human-readable form in online newspapers and electronic archives, software systems that can extract relevant information and present these information are scarce and this has been of significant interest to researchers in the field of Information Extraction (Cowie & Lehnert 1996). Even though search engines can be used to query specific information (e.g. cultural events in Auckland), these query results do not provide a historical perspective (i.e. if there are 100 articles on cultural events, the user may have to read all of these in order to gain some insights such as the increase in number of operas in a city). Although, one could manually read through the results and extract valuable information, this process is tedious and error prone. So, this work aims to ‘mine’ information available in online newspaper articles.

In this work, crime information extraction is chosen as the domain for investigation because crime is one of the key variables for people to decide whether to move to a new country (or relocate to a new city) and places to avoid when one travels. For example, a new immigrant may want to compare different cities based on crime rates or compare different neighbourhoods of a particular city to choose a safer one. A traveler may want to know which parts of a particular city to avoid. Currently, this information is not readily available for users, but these can be obtained from newspaper articles of a particular region, as they tend to report the important crimes. To demonstrate the viability of the approach for automatic extraction of crime information, the domain of theft has been chosen in this work. Theft information can be extracted at different levels (street level, suburb level and city level) and this information can be visualized on top of maps (e.g. Google Maps). The system that automatically extracts and presents such information has the potential to be used by the residents of various cities to undertake proactive ‘neighbourhood crime watch’ initiatives to reduce crime. It can also be used by potential immigrants and the visitors of the city to make informed decisions about where to live/stay and also take appropriate precautions when visiting certain areas. Additionally, city councils may use the system to identify crime hot-spots and then employ appropriate monitoring and controlling mechanisms.

In certain countries, crime information is available to the public on top of Google maps. For example UK government has crime map (UK Police 2013) available to the public. However many countries make only the coarse-grained crime information available to the public. For example, crime information in New Zealand (NZ Government 2013) contains coarse-grained information (e.g. total number of thefts in a
district or a province) which is not very useful to an average citizen. Individuals require fine-grained information (e.g., thefts in a particular suburb). Thus, the aim of this research is to extract crime information available in newspaper articles and make the hidden information publicly available. To achieve this goal, this paper discusses a methodology that consists of seven steps. It uses Named Entity Recognition (NER) to identify locations and then employs Conditional Random Field (CRF) to classify whether a sentence containing a location is a crime location or not.

Random Field (CRF) to classify whether a sentence to identify locations and then employs Conditional Random Field (CRF) used in this work. First, it presents a methodology for extracting crime information from newspaper articles. Second, it compares four NER techniques on their ability to identify location information. Third, it evaluates how well the classifier model created for labelling sentences from one English newspaper in New Zealand can be used for identifying theft location information from newspaper articles from three other newspapers, one from New Zealand, one each from Australia and India.

This paper is organised as follows. Section 2 provides an overview of the related work and techniques employed in the area of crime information extraction. Section 3 presents the methodology employed to identify sentences with locations and also to classify whether a sentence is a crime location sentence. Section 4 discusses the various experiments that were conducted using four different newspaper articles from three countries and the results. Section 5 discusses the merits and limitations of the work reported in the paper and also points towards future work. The conclusions are provided in Section 6.

2 Background and Related Work

This section provides an overview of the related work in the domain of crime extraction. It also provides a brief background on the two techniques, Named Entity Recognition (NER) and Conditional Random Fields (CRF) used in this work.

Crime monitoring and prevention is a domain of interest to all countries across the globe in order to make the world a safe place to live. Use of ICT technologies for this purpose has been around since the advent of computers. Currently, with massive amounts of data being available on an individual’s activities (e.g., Twitter and Facebook) and the widespread availability of news articles (e.g., through freely available online newspapers and YouTube), researchers have become interested in combining these information to monitor and prevent crimes. While an individual’s activities can be private, most newspapers are public. In this work we consider such publicly available information. Also the focus of this work is to identify crime locations (in particular theft crime) and make this available to the public. This information can potentially be used in crime prevention.

Researchers working in the area of crime information extraction have used several techniques. In particular, researchers have used techniques such as

A location mentioned in an article does not mean it is a crime location. The objective here is to identify whether the location mentioned in a sentence is a crime location or not.

...Crowd sourcing, data mining and machine learning for this purpose. The Wiki Crimes project (Wiki Crimes 2013, Furthado et al.,) harnesses the power of the crowd, where individuals report crime details online and other users can use this information to make decisions. However, a limitation of this approach is the difficulty of verifying the authenticity of the posted crimes.

Researchers have explored techniques for retrieving relevant information from unstructured documents. The process of extracting information from unstructured documents is difficult because it is written in natural language and the structure of the document is not known ahead of time (when compared to structured files such as databases). However, there has been a lot of work on identifying entities (e.g., person, place, organization) from unstructured documents in the field of natural language processing. Often called as Named Entity Recognition (NER), this technique has been shown employed in many domains (for an overview see (Nadeau & Sekine 2007)). For example, the Coplink project (Chen et al. 2004) of researchers at the University of Arizona aims at identifying information about criminals from police reports. It uses an Entity Extractor system that is based on AI techniques, for detecting identities of criminals automatically and also for analyzing criminal networks using clustering and block modeling.

There are other works that extract relationships between variables available in the form of structured information (e.g., identifying relationship between column variables of a database table) using data mining techniques such as cluster analysis. For example, the work of De Bruin et al. (2006) uses such an approach for analyzing criminal careers. Based on the analysis, they have identified four important factors (crime nature, frequency, duration and severity). By using these factors, they created criminals’ profiles and compared each criminal with all the other criminals using a new distance measure and also clustered similar criminals. They obtained data from the Dutch National Criminal Record Database for their study. Chandra et al. (2007) have employed clustering to identify crime hot-spots based on Indian crime records. These works are mainly based on structured data. In contrast to these works, the work reported in this paper aims to extract information from unstructured newspaper articles.

Researchers have used Conditional Random Fields (CRFs), a statistical modelling technique for machine learning. Conditional Random Fields (CRFs) are used to build probabilistic models that can be used to label data. It is a discriminative probabilistic model and it learns weights between features from the training dataset and outputs a model that can be used to assign labels for test data (for an overview see (Laferty et al. 2001)). They have been shown to offer several advantages over Hidden Markov Models (HMMs). Also, they avoid the labour bias problem suffered by the Maximum Entropy Markov Model (MEMM). CRFs have been used in many domains. For example, Angrosh et al. (2010) have used CRFs for classifying sentences in a research article into different categories (e.g., background sentence, related work sentence, and shortcoming sentence). Peng & McCallum (2006) have also used CRFs to extract information from research papers. The work reported in this paper employs NER algorithms to identify locations and a CRF algorithm to train a model based on a set of features defined by the authors which is subsequently used to assign labels to sentences. Ku et al. (2008) aim to extract crime information from a variety of sources such as police reports, victim state-
ments and newspaper articles. Among other types of crime information, they identify locations. Their focus is to identify just the locations and not whether the location is indeed a crime location (i.e. there could be other types of locations such as victim’s or offender’s hometown which may not be the crime location).

Our work is inspired by the approach used by Angrosh et al. (2010). The domains of interest are different in both works, hence, the features identified and labels used are distinct. Our work employs NERs for location identification, which was not required in the cited work because their domain of interest was in the area of labelling sentences in research articles based on their purpose (e.g. background sentence and shortcoming sentence) and does not involve locations details. Also, to the best of our knowledge our contribution is unique to the domain of crime information extraction.

3 Methodology

The objective of this work is to identify the theft location from a corpus and categorize each sentence in an article into Crime Location Sentence (CLS) and Not a Crime Location Sentence (NO-CLS).

This section describes the methodology used for classifying sentences in newspaper articles into CLS and NO-CLS sentences. Figure 1 shows the steps used in the methodology followed in this work.

1. Corpus building - The first step is to build a corpus of relevant newspaper articles for our study. We used Mozenda Web Screen Scrapper tool (Mozenda 2013) for this purpose. Mozenda is a web scrapping tool to extract the specific information from websites. If we train the tool by pointing and clicking at details that need to be extracted, the tool can extract the same set of information automatically from a given pool of documents. It also saves the extracted information in different formats for latter use. We built a corpus of theft-related articles.

2. Sentence tokenization - Upon extracting relevant newspaper articles, the individual sentences need to be extracted (i.e. an article needs to be split into individual sentences). We used a PunktTokenizer (Kiss & Strunk 2006) from the NLTK toolkit (Bird et al. 2009) for this purpose. The tokenizer divides a given article into a list of sentences.

3. Location identification - Upon extracting individual sentences, the locations in each of the sentences have to be identified. Locations are identified using Named Entity Recognition algorithms (Nadeau & Sekine 2007). These algorithms are discussed in Section 4.1.

4. Feature identification - The fourth step is to define a set of features that can then be used to assign labels to sentences. The sentences will be labelled as crime location sentence (CLS) and not a crime location sentence (NO-CLS). The features include sentHasCrimeTerm which means that a sentence has a crime term, sentHasCity-Loc which means that a sentence has city location etc. A list of terms used in features and their descriptions are provided in Table 2. Also, Table 2 provides examples of sample terms (or phrases) that are used to represent a feature and the number of such terms identified from a corpus of 70 articles from Otago Daily Times (discussed in Section 4). For example, the first row shows that the theft related crime terms include theft, burglary, and stolen. There were a total of 336 instances that were identified in the corpus. There were 55 unique terms defined for this purpose (in brackets)2.

5. Label assignment - Once features have been identified, the labels are manually assigned for each sentence in an article. The assigned labels are a) CLS - Crime Location Sentence and b) NO-CLS - Not a Crime Location Sentence. Figure 2 shows the sample Crime Location Sentence (CLS) and Not a Crime Location Sentence (NO-CLS). The first sentence has details about the theft (i.e. a car was stolen), the address of the theft (i.e. Norfolk St) and the city (i.e. Dunedin). The second sentence only has a street location. A human reading these sentences can classify the first sentence as a sentence that contains a location that is the crime location. However, the second sentence does not provide a clue whether the sentence is a crime location. In order for a system to classify a sentence into a crime location sentence, we need to assign features. It can be observed from Figure 3, three features have been defined for sentence one and one feature has been defined for sentence two. The feature extracting is automated by a python program using regular expressions which assigns features for each sentences. Once the features are assigned for the training data, each sentences has to be manually labelled2. The first sentence with the three features will be labeled as CLS and the second sentence will be labelled as NO-CLS. Some sample features of sentences and their labels are shown in Table 3.

6. Training CRF - From a dataset of articles that have been annotated with features and labels, certain percentage (e.g. 70%) is chosen as the training data. The CRF algorithm learns weights between features from training dataset and creates a model. This model, when given a new set of data (e.g. a new article with features),

\[2\]The values in brackets are absent in many because in those cases unique terms are not defined manually. These are location information (city, suburb etc.). They are obtained through regular expressions on sentences that are annotated with results from NERs.

\[3\]In supervised learning, a dataset is divided into two parts, training data and test data.
<table>
<thead>
<tr>
<th>No.</th>
<th>Features</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>sentHasCrimeTerm</td>
<td>Sentence has crime term</td>
</tr>
<tr>
<td>2.</td>
<td>prevsentHasCrimeTerm</td>
<td>Previous sentence has a crime term</td>
</tr>
<tr>
<td>3.</td>
<td>sentHasRegionLoc</td>
<td>Sentence has region location</td>
</tr>
<tr>
<td>4.</td>
<td>sentHasCityLoc</td>
<td>Sentence has city location</td>
</tr>
<tr>
<td>5.</td>
<td>sentHasSuburbLoc</td>
<td>Sentence has suburb location</td>
</tr>
<tr>
<td>6.</td>
<td>sentHasStreetLoc</td>
<td>Sentence has street location</td>
</tr>
<tr>
<td>7.</td>
<td>sentHasPersonLoc</td>
<td>Sentence has person location</td>
</tr>
<tr>
<td>8.</td>
<td>sentHasPoliceLoc</td>
<td>Sentence has police location</td>
</tr>
<tr>
<td>9.</td>
<td>sentHasCourtLoc</td>
<td>Sentence has court location</td>
</tr>
<tr>
<td>10.</td>
<td>sentHasLocation</td>
<td>Sentence has other country names</td>
</tr>
</tbody>
</table>

Table 1: Features defined and their descriptions

Figure 2: Sample CLS and NO-CLS sentences

automatically assigns labels to each of the sentences in the article. The model produced is used in the next step.

7. **Sentence classification** - Once the model is created, it is used to label the sentences (i.e. automatic label assignment as opposed to manual label assignment in Step 5) in the remaining articles (i.e. the test data). The labels obtained through the model are then compared with the labels assigned by the humans. We compute the precision, recall, f-score and accuracy for the results obtained. The formulae to compute these are given below where TP, FP, TN and FN are the number of true positives, false positives, true negatives and false negatives respectively.

\[
\text{Precision}(P) = \frac{TP}{TP + FP} \quad (1) \\
\text{Recall}(R) = \frac{TP}{TP + FN} \quad (2) \\
F - \text{score} = \frac{2PR}{P + R} \quad (3) \\
\text{Accuracy}(A) = \frac{TP+TN}{TP+TN+FP+FN} \quad (4)
\]

The steps listed in the methodology are at a high-level of abstraction. We describe them in more detail in the context of the experiments conducted in the next section.

4 Experiments and Results

We conducted four experiments to demonstrate the efficiency of the system designed to identify crime location sentences. In the first experiment, we studied the impact of four types of NER algorithms which identify locations on the classification obtained in a regional newspaper in New Zealand. Second, based on the locations identified by the best model, we evaluated the performance of our system on the accuracy of identifying crime location sentences from the articles in a regional newspaper (Otago Daily Times\(^4\)). Third, using the model created from Otago Daily Times articles, we labelled articles from another newspaper in New Zealand (New Zealand Herald\(^5\)) and investigate the accuracy of the developed model. Fourth, using the same model, we classified (i.e. labeled) sentences from newspaper articles from two countries, Sydney Morning Herald\(^6\) from Australia and The Hindu\(^7\) from India. We compared the accuracy of the results obtained. These experiments and the results are presented in the following subsections.

4.1 Comparing Efficiencies of Four Types of NER Algorithms on a Regional Newspaper

**Experimental set up** - We collected 70 articles from Otago Daily Times that contained “theft” as a search term using Mozenda. We tokenized these articles into sentences. We then investigated four different NER algorithms to find the one that yields the best results in identifying the locations correctly (step 3 of Figure 1).

The details of the four NER algorithms compared are given below.

1. **NLTK pre-trained named entity chunker** - The nltk named entity chunker (Bird et al. 2009) uses \texttt{nechunk} method from \texttt{nlkchunk} module to identify the named entities such as person, organization, geo-political entities (e.g. city, region and country).

2. **Stanford NER** - Stanford Named Entity Recognizer is a Java-based tool (Finkel et al. 2005). A

\(^4\)www.odt.co.nz 
\(^5\)www.nzherald.co.nz 
\(^6\)www.smh.com.au 
\(^7\)www.thehindu.com
1. Crime Terms
   Words or phrase that are related to theft crime
   Sample keywords: theft, burglary, stolen, failed to pay
   Number of terms identified: 336 (55)

2. Police Locations
   The word Police occurs after a specific location
   Sample keywords: Dunedin police, Timaru police
   Number of terms identified: 33

3. Region Locations
   The location identifies as a region
   Sample keywords: Otago, Canterbury
   Number of terms identified: 36

4. City Locations
   The location identifies as a city
   Sample keywords: Dunedin, Queenstown
   Number of terms identified: 122

5. Suburb Locations
   The location identifies as a suburb
   Sample keywords: Mary hill, Roslyn
   Number of terms identified: 88

6. Street Locations
   The location identifies as a street
   Sample keywords: Princes Street, Easther Cres
   Number of terms identified: 55

7. Court Locations
   The word Court occurs after a specific location
   Sample keywords: Dunedin District court
   Number of terms identified: 28

8. Person Locations
   Terms that describe a person occurs after a specific location
   Sample keywords: Dunedin man, Mosgiel woman, Wanaka pair, Auckland teenager
   Number of terms identified: 53 (19)

Table 2: Features defined and sample keywords

screenshot of the results obtained from this tool is given in Figure 4. It identifies four types of entities: location, organization, person and miscellaneous. We used the location information in this work. The algorithm uses the CRF classifier to train the model for identifying named entities.

3. NLTK chunker class using Gazetteer - We used LocationChunker class from NLTK cookbook (Perkins 2010). It uses the gazetteers corpus to identify location words. Gazetteer corpus is a large database containing the locations from all around the globe. However, the level of details available for each country varies. Users can upload their own (better) data to this corpus in order to obtain better results (i.e. reduce the number of false positives).

4. LBJ Tagger - LBJ NER Tagger (Ratinov & Roth 2009) is one of the models of the Named Entity Recognition system developed at the University of Illinois. This model is based on regularized average perceptron. It uses gazetteers extracted from Wikipedia, where the models for word class are derived from unlabelled text and expressive non-local features. We used the classic 4-label type model to identify the locations and organizations.

The results of these four algorithms to identify the locations correctly on the same set of 50 articles from Otago Daily Times is given in Table 4. Out of these four algorithms, LBJTagger has the highest accuracy (94%) followed by the NLTK Chunkparser (81%).

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Precision</th>
<th>Recall</th>
<th>F-Score</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>NLTK pre-trained named entity chunker</td>
<td>0.93</td>
<td>0.78</td>
<td>0.85</td>
<td>0.74</td>
</tr>
<tr>
<td>Stanford NER</td>
<td>0.90</td>
<td>0.86</td>
<td>0.90</td>
<td>0.86</td>
</tr>
<tr>
<td>NLTK Chunkparser using Gazetteer</td>
<td>0.95</td>
<td>0.91</td>
<td>0.93</td>
<td>0.81</td>
</tr>
<tr>
<td>LBJ Tagger</td>
<td>0.98</td>
<td>0.96</td>
<td>0.97</td>
<td>0.94</td>
</tr>
</tbody>
</table>

Table 4: Comparisons of four different NER algorithms based on location identification

The data about organizations are used to in conjunction with the data about locations to in order to improve the accuracy of locations in our work.

4.2 Accuracy of Crime Sentence Labelling in a Regional Newspaper

We used the best NER algorithm, the LBJTagger to identify locations in the 70 Otago Daily Times (ODT) articles. Then, the features in all these articles were identified using a Python program which employed regular expressions (step 4 in Figure 1). The labels were assigned manually for the training set (step 5). We used Mallet, a Java-based tool (McCallum 2002), to train the CRF. Mallet uses SimpleTagger class for training and testing datasets. We evaluated our results using 10-fold cross validation by splitting the dataset into 10 sets of training and test data sets. One set contained 63 training articles and 7 test articles (training to test ratio of 9:1 following the work of Angrosh et al. (2010)). Table 5 shows the precision, recall, f-score and accuracy for the test dataset. We achieved overall accuracy of (84% overall, with individual accuracies of CLS
Table 5: Result of 10-fold cross validation (ODT articles using LBJ tagger)

<table>
<thead>
<tr>
<th>Label</th>
<th>First and Zero order</th>
<th>Precision</th>
<th>Recall</th>
<th>F-score</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLS</td>
<td>Precision</td>
<td>0.87</td>
<td>0.88</td>
<td>0.87</td>
<td>0.77</td>
</tr>
<tr>
<td>NO-CLS</td>
<td>Recall</td>
<td>0.93</td>
<td>0.93</td>
<td>0.93</td>
<td>0.88</td>
</tr>
</tbody>
</table>

Table 6: Results of comparisons of three newspapers (First and Zero order)

<table>
<thead>
<tr>
<th>Label</th>
<th>NZ Herald</th>
<th>Precision</th>
<th>Recall</th>
<th>F-score</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLS</td>
<td>Precision</td>
<td>0.90</td>
<td>0.94</td>
<td>0.92</td>
<td>0.85</td>
</tr>
<tr>
<td>NO-CLS</td>
<td>Recall</td>
<td>0.96</td>
<td>0.96</td>
<td>0.92</td>
<td>0.92</td>
</tr>
</tbody>
</table>

No-CLS being 77% and 88% respectively) using first and zero order CRF. The confusion matrix for the results obtained for the 70 articles (with a total of 523 sentences) is given in Figure 5. 155 sentences marked as CLS sentences by a human were also identified as CLS articles by the classifier model that has been developed in this work. 322 NO-CLS articles have also been identified correctly.

![Figure 5: Confusion matrix for 70 articles from ODT for 10-fold cross validation](image)

4.3 Comparison of Crime Location Extraction from Two Newspapers in New Zealand

Our hypothesis was that English used within New Zealand will be similar. Hence, the model that was created from training samples in Otago Daily Times (based in Dunedin) must be applicable for labelling articles of New Zealand Herald (based in Auckland). To test this hypothesis, we chose 50 theft-related articles from New Zealand Herald. The results obtained are presented in Table 6. It can be seen that we achieved a high accuracy (overall accuracy of 90% with individual accuracies for CLS and NO-CLS sentences being 85% and 92% respectively). So, the results obtained for New Zealand Herald is in support of our hypothesis that the model trained for ODT is applicable to New Zealand Herald. However, we need to conduct a large study involving more articles. Also, this might not be true to all countries. The use of English (e.g. written style) can vary from one part of the country to another.

4.4 Comparison of Crime Location Extraction Across Countries

We further hypothesized that the model developed for labelling newspaper articles in New Zealand might be adequate for classifying newspaper articles in Australia. In order to examine our hypothesis, we chose 50 theft-related articles from Sydney Morning Herald. The model reported in the previous step (i.e. model developed by training the 70 articles obtained from Otago Daily Times was our training data) was used to test the data from the 50 theft-related newspaper articles from Sydney Morning Herald. The results are presented in Table 7. It can be observed that the accuracy of the results was lower when compared to the articles from New Zealand newspapers (overall accuracy of 75% with individual accuracies for CLS and NO-CLS being 64% and 81% respectively). We also conducted a similar study on 50 articles from The Hindu. The accuracy was low (overall accuracy of 73% with individual accuracies for CLS and NO-CLS being 59% and 79% respectively).

We investigated the reasons for difference in the accuracy. There were two main reasons. First, the efficiency of the LBJTagger on these articles was lower than the articles from New Zealand (i.e. locations were not identified correctly to start with which impacted the final results). Second, there were more instances of crime locations occurring in sentences that were apart (i.e. first sentence of the article talks about the crime and the fifth sentence specifies the crime location). Such instances were relatively rare in the articles obtained from New Zealand newspapers\(^2\). This primarily is an issue of writing style. Currently our work does not handle relationship between sentences. For example, it does not tie information from the first sentence (the crime information) and the fifth sentences (the location information) together. This is one of the areas for further work.

5 Discussion

We have have demonstrated the best results on sentence classification are obtained when LBJTagger is used. The accuracy of location identification is crucial for our approach, because the quality of this step (step 3) affects the subsequent steps. We have demonstrated that our approach works well for labelling sentences in Otago Daily Times articles into crime location sentences or not (accuracy of 84%). Also, we have demonstrated that re-usability of this model in the context of another newspaper (i.e. the accuracy for the articles in NZ Herald was 90%). However, the accuracies obtained by employing the same model for newspapers from other countries are slightly lower (75% and 73% for articles from Australia and India). We have discussed the underlying reasons and what needs to be done in the future. There are a couple of approaches in building models for crime sentence identification. The first one is to create individual models (one for each English speaking country). The second one is to create a global model which can be used for this purpose. The second one can be built from the datasets of the first one (i.e. global model can be built as an agglomeration of the local models). We currently have extracted fine grained information from the theft related articles in Otago Daily Times and New Zealand Herald. These information include city location, suburb location. We used Google Fusion Tables (Halevy & Shapley 2009) to dis-

\(^2\)This might not be the same for other crimes.
play this information. By clicking on a balloon on a map, the article related to that particular crime location can be viewed by the user (see snapshot shown in Figure 6). Currently, we provide details such as the title of the article, the URL, crime terms and the crime location. We are planning to modify this set up with a set of information that might be beneficial to a variety of stakeholders. For example, we plan to make the following pieces of information available to the user.

1. Offender’s place of origin
2. Victim’s place of origin
3. Police involvement details
4. Court involvement details
5. Involvement of organizations

There are a few limitations of the research work. First, the relationship between different sentences have not been explored. For example, the crime details may be in the first sentence and the location details can be in the fifth sentence of the same paragraph or even in the subsequent paragraph. This has not been modeled in this work. Using an appropriate relationship identification algorithm (e.g. Sutton & McCallum (2007)) for this purpose is the next step of this research. Second, we have not considered eliminating the duplicate articles reporting the same crime within a newspaper (e.g. elaborate news may follow brief news items) and across newspaper articles since our study was a feasibility study to demonstrate that our approach works. We plan to consider this in the future. Third, the approach uses a small sample size (70 articles in one newspaper) for the training data set. We believe, we will be able to improve the results by increasing the number of articles considered. Despite these limitations, we believe, the research reported in this work can be used to create a system which will be beneficial for visitors, immigrants to a city to make right decisions about where to stay/live and which areas to avoid. Also, the system will be useful for neighbourhood watch groups and city councils to monitor and prevent crimes.

A further extension of this study is to consider the full range of crimes as categorized in law (Australian Government 2013) and also extend this to other domains such as extracting historical record of cities on their cultural events, sports events, etc. Historical newspapers can be obtained from Archives New Zealand (New Zealand Government Archives 2013) that contain valuable historical events which can then be mined and visualized using a system like ours. For example, 19th century Dunedin can be visualized on top of the map based on the type of activities that were reported in newspaper articles between 1861 to 1900.

6 Conclusions

This paper presents a methodology for extracting crime location sentences (particularly ‘theft’ crime information) from online newspaper articles. It employs named entity recognition (NER) algorithms to identify locations in sentences and uses Conditional Random Field (CRF) to classify sentences into crime location sentences. The proposed system is evaluated on four newspaper articles from three different countries. It demonstrates that the accuracy of the results obtained for New Zealand articles varies from 84% to 90%. For articles from the two other countries (India and Australia) it varies from 73% to 75%. We have also discussed the limitations of our work and the areas for future improvements.

7 Acknowledgments

We would like to thank the Information Science department for the internal grant to support the first author to pursue this research. Also, special thanks to Angrosh Mandya for his valuable help during the early stages of this work.

References


Bird, S., Klein, E. & Loper, E. (2009), Natural language processing with Python, O’reilly.


Perkins, J. (2010), Python Text Processing with NLTK 2.0 Cookbook, Packt Publishing Ltd.


The Efficiency of Corpus-based Distributional Models for Literature-based Discovery on Large Data Sets

Michael Symonds¹  Peter Bruza¹  Laurianne Sitbon²

¹ School of Information Systems, Queensland University of Technology, Brisbane, Australia
² Computer Science Department, Queensland University of Technology, Brisbane, Australia

Email: michael.symonds@qut.edu.au, p.bruza@qut.edu.au, laurianne.sitbon@qut.edu.au

Abstract

This paper evaluates the efficiency of a number of popular corpus-based distributional models in performing discovery on very large document sets, including online collections. Literature-based discovery is the process of identifying previously unknown connections from text, often published literature, that could lead to the development of new techniques or technologies. Literature-based discovery has attracted growing research interest ever since Swanson’s serendipitous discovery of the therapeutic effects of fish oil on Raynaud’s disease in 1986. The successful application of distributional models in automating the identification of indirect associations underpinning literature-based discovery has been heavily demonstrated in the medical domain. However, we wish to investigate the computational complexity of distributional models for literature-based discovery on much larger document collections, as they may provide computationally tractable solutions to tasks including, predicting future disruptive innovations.

In this paper we perform a computational complexity analysis on four successful corpus-based distributional models to evaluate their fit for such tasks. Our results indicate that corpus-based distributional models that store their representations in fixed dimensions provide superior efficiency on literature-based discovery tasks.

Keywords: Efficiency, literature-based discovery, corpus-based distributional models

1 Introduction

This paper examines, the often overlooked, impact of a model’s computational complexity on the successful application to the task of literature-based discovery (LBD) on very large data sets. LBD relies on the identification of undiscovered connections between concepts in literature (including online document collections). These concepts are often linked indirectly via other concepts, as illustrated by the medical discovery process linking illness A and drug C depicted in Figure 1. This approach was first popularised by Swanson (1986) in discovering the link between fish oil (Drug C) and Raynaud’s disease (Illness A).

The need for a complexity analysis of corpus-based models on the task of discovery, stems from: (i) the growing size of literature collections (especially those found online), and (ii) the low cost and flexibility of corpus-based models that do not rely on hand-crafted semantic resources, such as ontologies and thesauri. Over the past three decades cognitive science researchers have developed a class of corpus-based models, known as corpus-based distributional models, that automatically build representations of words from their occurrence patterns (distributions) in streams of natural language, hence their name distributional models. The most well-known of these is the latent semantic analysis (LSA) model (Landauer & Dumais 1997). Corpus-based distributional models have seen growing interest due to their low cost and proven effectiveness on a wide range of applications, including information retrieval (Turney & Panpel 2010, Symonds, Bruza, Zuccon, Koopman, Sitbon & Turner 2013).

The ability to efficiently and effectively perform discovery across multiple domains of knowledge (as represented by the document collections found on the web) is especially relevant in novel research areas, such as the detection of future disruptive innovations (Christensen 2006). Disruptive innovations are those which initially offer a lower performance according to the mainstream, however, offer some new performance attributes that make them prosper in a different market, during which time their performance in traditional markets improves to the point that they displace the former technology. The mobile phone is an example of a disruptive innovation, as it initially offered poorer sound quality and were expensive. However, their advantage was their portability. As the sound quality improved and price dropped, they replaced the analogue phone.

The ability to identify future disruptive innovations may be possible through accessing many very large data sources, including patent information and online document collections (Daim et al. 2006). These data sources are likely to be very large, multi-lingual and given their nature (i.e., documenting new concepts) unlikely to have available hand-crafted semantic resources (such as dictionaries, thesauri and other hand-crafted ontologies). Therefore, LBD techniques that are underpinned by corpus-based distributional models are likely to be well suited to these emerging areas of research. Which corpus-based model would perform the best on discovery? This depends on the efficiency and effectiveness of a model. This research focuses on evaluating the efficiency of a number of successful, corpus-based distributional models for the task of discovery.

This paper is structured in the following way: (i) A review of current LBD practices, including the iden-
tification of relevant weaknesses beyond the growing complexity issues targeted in this work, (ii) A review of the complexity of four popular corpus-based distributional models and how they are positioned with respect to the existing weaknesses of LBD approaches, and (iii) A discussion of findings from the review and complexity analysis that can help researchers select corpus-based distributional models that are most likely to provide superior efficiency and effectiveness on the task of LBD.

2 Related Work

The two areas of work that underpin this research include: (i) literature-based discovery (LBD), and (ii) the use of corpus-based distributional models to identify potentially useful, undiscovered connections.

2.1 Literature-based Discovery

Since the serendipitous discovery of the therapeutic effects of fish oil on Raynaud’s disease by scientist Don Swanson in 1986 (Swanson 1986), the field of literature-based discovery has seen strong interest, as evidenced by the increasing papers, conferences, workshops, books and reviews of LBD research (Weeber et al. 2005, Bruza & Weeber 2008). LBD aims to identify possible useful undiscovered connections between concepts in literature. This LBD process relies on being able to identify intermediary concepts, i.e., concepts that link two other concepts that are not directly linked to each other within the literature. For example, concept A may be known to have an association with concept B (as demonstrated by their co-occurrence in one or many documents), while concept C may also be known to have an association with concept B. However, if A and C do not co-occur in any documents together their indirect association through B (the intermediary or bridging concept) may be unknown. If yet undiscovered, this link through B may provide researchers with valuable insights that could potentially lead to advances in technology or identification of disruptive innovations.

The identification of B within the LBD process allows for two modes of discovery, termed open and closed. Open discovery involves two steps: (i) starting with a known concept A, identify a limited number of possible intermediary concepts (i.e., B concepts that co-occur with A), and (ii) exploring the literature containing B concepts to identify potentially useful C concepts. In closed discovery, the process starts with a hypothesized connection between A and C, and an explanation for this observation is sought by finding an appropriate B that co-occurs with both A and C.

Both of these discovery methods can lead to very large numbers of possibly useful undiscovered connections, especially within very large, modern corpora (e.g., medical or patent collections). To illustrate, consider that each concept in the corpus (referred to as A in the example above) may co-occur with thousands of other concepts (referred to as B). Given that B may also co-occur with thousands of other concepts (referred to as C) not seen with A, the number of possible useful connections may be in the order of hundreds of thousands. Therefore, the next critical step is to narrow the list of possibilities.

Early LBD researchers relied on human experts to reduce the list. However, this is a very costly process, and intractable in large modern data sets. Therefore, modern LBD researchers have relied on a number of scalable methods, based on distributional statistics, to limit the list of useful connections. Many of these rely on co-occurrence information alone (Gordon & Lindsay 1996, Gordon & Dumais 1998). However, within the medical domain researchers have argued that co-occurrence information on its own does not take advantage of higher order semantic relationships that exist between concepts (Cohen et al. 2012, Hristovski et al. 2006). One approach to extending distributional models has been to incorporate natural language tools (including ontological information) to produce a predictive LBD method that aims to identify discovery patterns (Hristovski et al. 2006). Other LBD researchers have taken these ideas and combined them with more efficient distributional approaches aimed at overcoming the computational complexity issues that arise with earlier distributional models (Cohen et al. 2012).

Two of the strongest criticisms of current LBD approaches is argued to stem from (i) the bias of distributional models toward high frequency concepts (Kostoff 2008) and (ii) the need for hand-crafted semantic resources, which may exist for medical LBD, but are not available for many other domains, and would be very expensive to create. With regard to the first criticism, Kostoff (2008) argues that high frequency concepts are less likely to be undiscovered, therefore, any system biased toward them is less likely to be effective at discovery. The solution proposed by Kostoff (2008) is to include more human intervention (primarily by authors in the fields in which the discovery process is undertaken). However, such human-dependent approaches, known as literature related discoveries (LRD (Kostoff 2008)), will naturally be more costly, especially as the literature for a field of knowledge naturally grows. Therefore, identifying best practice in LBD and extending these approaches to account for the weaknesses identified by researchers is required (i.e., not relying on hand-crafted semantic resources, and reducing the bias toward high frequency concepts). We argue that this is likely to be achieved using corpus-based distributional models.

2.2 Corpus-based Distributional Models

Corpus-based distributional models commonly use word order and co-occurrence information found in streams of natural language to build geometric or probabilistic representations of concepts. The premise of these models stems from the distributional hypothesis, which states that words with similar meaning will tend to co-occur with similar words (Harris 1954). Within distributional models,
the semantic associations that underpin the meaning of words can be modelled using measures of similarity, specific to the mathematical framework in which they are set. For example, in a geometric setting the strength of semantic associations can be measured from the distance between concepts in the space, hence their popular name semantic space models. In the past, corpus-based distributional models have been successfully applied to applications that involve relatively large data sets, including synonym judgement (Landauer & Dumais 1997, Symonds et al. 2011).

The advantage of corpus-based distributional models over those using hand-crafted semantic resources, is their reduced cost and flexibility in modelling concepts based on the context they are seen within the training corpora, as well as the ability for the model to be applied to document collections of any language. An LBD approach that relies solely on a corpus-based distributional model begins to address the concerns raised by Kostoff (2008) as long as it can be shown to have reduced bias to high frequency concepts. This point will be explicitly addressed in addition to the computational complexity calculations for each of the models reviewed in this work.

3 Computational Complexity

The implementation of the open discovery process using corpus-based distributional models can be broken down into the following computational steps:

1. Pre-processing of the documents (i.e., stemming, stopping, etc)
2. Building representations for vocabulary terms.
3. Retrieving terms based on the similarity of representations.

Within this work, we assume all models evaluated use the same methods to achieve step 1 (pre-processing). Therefore, our complexity analysis focuses on the costs of achieving steps 2 (building the representations) and 3 (computing similarity) for four successful corpus-based distributional models. The four models include (i) LSA (Latent Semantic Analysis (Landauer & Dumais 1997)), (ii) HAL (Hyper-space Analogue to Language (Burgess et al. 1998)), RI (Random Indexing (Kanerva et al. 2000, Karlgren & Sahlgren 2001)), and the TE model (Tensor Encoding (Symonds et al. 2011)). All of these approaches build representations for each term in the vocabulary of the document collection. Topic models, such as latent dirichlet allocation (LDA) (Blei et al. 2003) and probabilistic latent semantic analysis (pLSA) (Hofmann 1999), were not included in this research as we were unable to find any previous examples using topic models for open discovery. This may be due to the reduced chance of discovery when using a limited set of latent topics in the discovery process. Increasing the number of topics is likely to have a significant impact on efficiency, in the case of LDA the complexity becomes NP-hard (Sontag & Roy 2011). A detailed investigation into the possible use of topic models for open discovery, and their efficiency is left for future work.

The complexity analysis in this work assumes that the document collection is stored on disk and that all steps of the LBD process can be achieved in main memory. For large data sets, the representations may not entirely fit within main memory. In this case it is assumed that a large main memory will cache a sufficiently large proportion of working data such that retrieval time is not affected. From an implementation point of view, however, the question of how to efficiently retrieve out of core representations is a question for future research. Storage complexity will be measured in terms of a number representation with 32 bits of precision, such as a 32 bit float or integer.

The analysis will also provide a computational estimate of performing LBD using each model on the MAREC patent document collection1 (Table 1). MAREC is a static collection of over 19 million patent applications and granted patents from a number of international sources, spanning a range from 1976 to June 2008. As the MAREC dataset has a vocabulary size in the order of tens of millions, it is considered to be an example of a very large collection. It is also similar to collections found on the web as it covers multiple domains of knowledge in more than one language, and provides a reasonable chance of containing previously undiscovered links between concepts that could be used for discovering future disruptive innovations.

The complexity analysis considers storage complexity, denoted as \( M(n) \) representing the memory requirements for a given input size \( n \), and time complexity, denoted as \( T(n) \) representing the worst case time complexity for a given input size \( n \). The analysis begins by considering the complexity of LSA.

3.1 Latent Semantic Analysis (LSA)

LSA is probably the best known corpus-based distributional model. LSA builds latent representations of vocabulary terms from a full term-document matrix created from the training corpus (Landauer & Dumais 1997). Even though LSA uses a reduced matrix form to calculate the semantic similarity of vocabulary terms, the full term-document matrix needs to initially be constructed. Each term’s vector representation is a row in the matrix whose elements are the frequency of the term in each document. For example, on the MAREC patent document collection (Table 1) the full term-document matrix would be a 75,000,000 × 19,000,000 matrix. The full term-document matrix can be obtained by building an index of the collection. LSA then applies a technique from linear algebra, known as singular value decomposition (SVD), to reduce the matrix to the \( k \) most significant latent terms (where \( k \) is the number of singular values to be used in computations). SVD is an expensive process, however, once it has been performed, only the reduced matrix needs to be used to perform similarity calculations between vocabulary terms, which is ultimately required when performing open discovery.

\[\begin{array}{c|c|c|c}
\text{Collection} & |D| & |V| & |C| \\
\hline
\text{MAREC} & 19,386,697 & 44,547,422 & 63,611,683,654 \\
\end{array}\]

Table 1: Details of the MAREC document collection used as an example in the computational complexity analysis of each model. \( |D| \) is the number of documents in the collection, \( |V| \) represents the size of the vocabulary and \( |C| \) represents the total number of terms in the collection.

1http://www.ir-facility.org/prototypes/marec
3.1.1 Building Representations

From a storage complexity perspective, LSA still requires the full term-document matrix to be created, so initially it has a storage complexity of \( M(n) = O(|V| \times |D|) \), where \(|V|\) is the size of the vocabulary and \(|D|\) is the number of documents in the training collection. In the case of the MAREC data set, \( M(n) = (75 \times 10^3)(19 \times 10^6) = 1.425 \times 10^{15} \).

From a time complexity perspective, the SVD process is the most costly, and has a time complexity of \( T(n) = O(|V|^2|D| + |D|^3) \). Therefore, for the MAREC data set, \( T(n) = (75 \times 10^6)^2(19 \times 10^6) + (19 \times 10^6)^3 = 1.14 \times 10^{24} \).

3.1.2 Computing Similarity between Terms

Similarity calculations between vocabulary terms are often achieved using geometric measures such as the cosine metric or a Minkowski norm (i.e., city block or Euclidean distance) to compare the vector representations in the latent concept space. The cosine measure has demonstrated robust effectiveness on a number of tasks, including similarity judgement (Landauer & Dumais, 1997).

For computing step 3 of the open discovery mode (i.e., retrieving terms based on their similarity) in LSA the B concepts in the vocabulary can be found by (i) listing the nearest neighbours to the A concept (i.e., performing a cosine measure across the vocabulary with A), and then (ii) performing a cosine similarity between each potential B concept and those found in a reduced vocabulary created by removing all terms that appeared in any of the documents A had appeared in. The time complexity of step (i) would be \( T(n) = O(|V|(|d_i|)) \), where \(|d_i|\) is the dimensionality of the reduced vectors. Assuming there are \( b \) intermediary concepts (i.e., highest ranked \( B \) concepts), then the overall time complexity of performing open discovery with the LSA model is \( T(n) = O(|V|(|d_i|) + b|V_c|(|d_i|)) \), where \(|V_c|\) is the number of vocabulary terms that did not co-occur with \( A \).

Setting \( d_i = 300 \), \( b = 1,000 \), and \(|V_c| = 0.8 \times |V| \), the worst case time complexity of computing \( C \) concepts for open discovery in the MAREC dataset is \( T(n) = (7.5 \times 10^3)(300) + (1 \times 10^3)(0.8 \times 7.5 \times 10^7)(300) = 1.8 \times 10^{12} \).

It is worth noting that the second step in the retrieval process may require a further SVD operation (for each \( A \) concept of interest) to be performed on the term-document matrix containing all possible \( C \) concepts (i.e., that did not co-occur in documents containing concept \( A \)). This would further increase the time complexity of building the semantic space for the LSA model. LSA has demonstrated effective performance on tasks, such as synonym judgement (Landauer & Dumais, 1997), that evaluate associations between low frequency concepts, and hence LSA is likely to address the criticism raised by Kostoff (2008) that for open discovery effective distributional models should not be biased toward high frequency terms.

3.2 Hyperspace Analogue to Language (HAL)

The HAL model creates a term co-occurrence matrix by moving a sliding context window across the training corpus and collecting co-occurrence frequencies between terms (Burgess et al. 1998). To illustrate, consider the HAL matrix shown in Table 2, which was created for the toy sentence “A dog bit the mailman,” using a sliding context window of length 5 (i.e., 2 words either side of the focus word). The co-occurrence information preceding and following each word are recorded separately by the row and column vectors. The values assigned to each co-occurrence are scaled by their distance from the focus word, with words next to the focus word given a value of 2 (when a context window length of 5 is used), and those at the edge of the window scaled by 1.

3.2.1 Building Representations

This \(|V| \times |V|\) matrix means that the storage complexity of the HAL model is \( M(n) = O(|V| \times |V|) = O(|V|^2) \). For the MAREC corpora, \( M(n) = (7.5 \times 10^3)^2 = 5.6 \times 10^{15} \). The time to build the vocabulary representation within the HAL model involves incrementing the vector elements as the context window is moved across the documents, and has a worst case time complexity equal to \( T(n) = O(|C| \times s) \), where \(|C|\) is the total number of terms in the collection and \( s \) is the size of the context window. For the MAREC data set where \(|C| = 6.6 \times 10^{10}\), we set \( s = 5, T(n) = (6.6 \times 10^{10}) \times 5 = 3.3 \times 10^{11} \).

3.2.2 Computing Similarity between Terms

Computing step 3 of the open discovery process (retrieving terms based on similarity) in HAL involves, (i) identifying the \( B \) concepts in the vocabulary by listing the nearest neighbours to the \( A \) concept (i.e., performing a cosine measure across the vocabulary with \( A \)), and then (ii) performing a cosine similarity between each potential \( B \) concept and those found in a reduced vocabulary created by removing all concepts that appeared in any document that contained concept \( A \). For the HAL model, the time complexity of step (i) would be \( T(n) = O(|V| |V|) = O(|V|^2) \). Assuming there are \( b \) intermediary concepts (i.e., highest ranked \( B \) concepts), then the overall time complexity of performing open discovery with the HAL model is \( T(n) = O(|V|^2 + b|V_c| |V|) \), where \(|V_c|\) is the number of vocabulary terms that did not co-occur with \( A \). Setting \( b = 1,000 \) and \(|V_c| = 0.8 \times |V| \), the worst case time complexity of computing \( C \) concepts for open discovery in the MAREC dataset is \( T(n) = (7.5 \times 10^3)^2 + (1 \times 10^3)(0.8 \times 7.5 \times 10^7)(7.5 \times 10^7) = 4.5 \times 10^{18} \).

To reduce the computational complexity of the HAL model, researchers have previously only retained the dimensions of the \( k \) most frequent terms in the vocabulary, where \( k \) is often around 100,000 (Bullinaria & Levy 2007). In this way, the storage complexity becomes, \( M(n) = O(|V| \times k) \). However, given low frequency concepts are most likely to produce useful discoveries (Kostoff 2008), ignoring them is unlikely to provide the most effective LBD outcomes. Therefore, the HAL complexity associated with the full term-term matrix will be used for the comparative analysis in this work.
3.3 Random Indexing (RI)

RI is a more recent semantic space approach that creates fixed dimension vector representations, the size of which are independent of the number of terms in the vocabulary. These representations are created from an approximately orthogonal basis formed by assigning each term a random environment vector of dimensionality \(d_c\), where \(d_c \ll |V|\). The final representations for vocabulary terms are created by summing the environment vectors of terms that co-occur within a sliding context window that is moved across the corpus.

3.3.1 Building Representations

Fixing the dimensions of the representations reduces the storage complexity of the model to \(M(n) = O(2|V|(d_c))\), where \(d_c\) is the dimensionality of the context vectors (and environment vectors). Recent LBD research (Cohen et al. 2012), using an enhanced RI model based on bit vectors and incorporating a NLP resource for our complexity analysis. The resulting storage complexity of RI for the MAREC dataset would be \(M(n) = 2 \times (7.5 \times 10^5) (1 \times 10^9) = 1.5 \times 10^{11}\). Which is almost 30,000 times less than the memory footprint for HAL.

The time complexity of building the RI semantic space is similar to HAL, except that all elements of the vectors must be summed (c.f., as opposed to incrementing a single element value), as the RI model being considered is based on dense distributed representations in which the dimensions of the vectors do not relate to a term-id or document-id. The worst case time complexity of building the representations within the RI model would be \(T(n) = O(|C|(|s|d_c))\), which is \(d_c\) times greater than the HAL model. Setting \(d_c = 32,000\) (as each bit needs to be considered in building and comparing vectors) and \(s = 5\), the time complexity of the RI model to build the vocabulary for the MAREC training collection becomes, \(T(n) = (6.6 \times 10^{10}) \times 5 \times (3.2 \times 10^3) = 1.1 \times 10^{17}\).

3.3.2 Computing Similarity between Terms

When performing similarity within the RI model, a geometric measure such as the cosine metric or Minkowski measure is often used, as done in past research applying RI to LBD (Cohen et al. 2012). For the RI model using bit-vectors, as used in Cohen et al. (2012), the cosine similarity measure is actually the Hamming distance, and the process for computing step 3 of the open discovery process using RI involves (i) comparing all vocabulary terms to the representation of \(A\) (i.e., \(T(n) = O(|V|d_c)\)), and (ii) computing the similarity of each \(B\) concept with all vocabulary terms that did not co-occur in documents that contained \(A\) (i.e., \(T(n) = (b)|V_c|d_c\), assuming there are \(b\) intermediary concepts; i.e., highest ranked \(B\) concepts). Therefore, the time complexity of performing the retrieval process in open discovery with the RI model is \(T(n) = O(|V|d_c + (b)|V_c|d_c)\). Setting \(d_c = 32,000\), \(b = 1,000\) and \(|V_c| = 0.8 \times |V|\), the worst case time complexity of computing \(C\) concepts for open discovery in the MAREC dataset is \(T(n) = (7.5 \times 10^{10})(3.2 \times 10^5) + (1 \times 10^9)(0.8 \times 7.5 \times 10^7)(3.2 \times 10^5) = 1.92 \times 10^{16}\).

RI models used in semantic space research often require a form of frequency cut-off to be applied to achieve superior task effectiveness (Sahlgren et al. 2008, Karlsgren & Sahlgren 2001, Cohen et al. 2012). Frequency cut-offs are often used to remove very high frequency terms, however, they can also be used to remove low frequency terms (Karlosgren & Sahlgren 2001). Therefore, RI may have difficulty addressing the concern raised by Kostoff (2008) relating to the bias toward high frequency terms argued to exist in current LBD approaches using distributional models.

3.4 The Tensor Encoding (TE) model

The TE model is a recent model of word meaning that has demonstrated superior effectiveness over a state-of-the-art HAL-based model on a number of semantic tasks, including synonym judgement and the similarity judgement of medical concepts (Symonds et al. 2011, 2012).

3.4.1 Building Representations

The TE model builds tensor representations for vocabulary terms through a unique binding process. These sparse tensor representations are stored in low-dimensional storage vectors, whose dimensionality is independent of the vocabulary size.

To demonstrate, consider the construction of a vocabulary term \(bit\) for the following example sentence, \emph{a dog bit the mailman}, and the resulting vocabulary terms and environment vectors in Table 3.

<table>
<thead>
<tr>
<th>Term Id</th>
<th>Term</th>
<th>Environment vector</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>dog</td>
<td>(e_{dog} = (1\ 0\ 0)^T)</td>
</tr>
<tr>
<td>2</td>
<td>bit</td>
<td>(e_{bit} = (0\ 1\ 0)^T)</td>
</tr>
<tr>
<td>3</td>
<td>mailman</td>
<td>(e_{mailman} = (0\ 0\ 1)^T)</td>
</tr>
</tbody>
</table>

Table 3: Example vocabulary for the sentence: \emph{A dog bit the mailman}

For the second-order TE model the representations are constructed by summing the proximity-scaled outer products of the environment vectors found within a sliding context window moved over the text. To demonstrate, consider the memory matrices created by the TE model’s second order binding process for vocabulary term \emph{2 (bit)} where a sliding context window of radius \(2\) is chosen:

\[
A = \begin{bmatrix}
dog \\
[\text{bit}]
\end{bmatrix} \text{the} \\
mailman
\]

\[
M_{bit} = 2 \times e_{dog} \otimes e_{bit}^T + e_{bit} \otimes e_{mailman}^T
\]

\[
= 2 \times \begin{bmatrix}
1 \\
0
\end{bmatrix} \begin{bmatrix}
0 & 1 & 0 \\
0 & 0 & 1
\end{bmatrix} + \begin{bmatrix}
0 & 1 & 0 \\
0 & 0 & 1
\end{bmatrix}
\]

\[
= \begin{bmatrix}
0 & 2 & 0 \\
0 & 0 & 1
\end{bmatrix}
\]

(1)

The matrix representations for \emph{bit} can be stored efficiently in the following storage vector (SV):

\[
SV_{bit} = (-1\ 2\ 3\ 1),
\]

(2)
where parenthesis have been added to illustrate implicit grouping of \((T \ CF)\) pairs, where \(T\) is the term id of the co-occurring term with term \(w\) and \(CF\) is the cumulative, proximity-scaled, co-occurrence frequency of \(T\) with \(w\), and \(w = 2\) in this example, as bit is the second term in the vocabulary). The sign of \(T\) (term-id) indicates the word order of \(T\) with \(w\). The information in this vector can be used to reconstruct the memory matrix using the following process:

1. If the term id \((T)\) is positive, the \(CF\) value is located at row \(w\), column \(T\) in the memory tensor. Otherwise, the \(CF\) value is located at row \(T\), column \(w\).

At an implementation level, the construction of the second-order representations can be efficiently achieved using fixed dimension storage vectors and the following process:

1. For each co-occurrence with target term \(w\), search the storage vector \((SV_w)\) for a matching \(T\) value and its sign to ensure it occurs in the same word order with \(w\).
2. If a match is found then, the \(CF\) element of the pair is increased by the scaled, co-occurrence frequency of \(w\) with \(T\) within the current context window. End process.
3. If no match is found then, check if the storage vector is full
4. If the storage vector is full then, the first low information pair in the storage vector should be removed and the new pair added to the end of the storage vector.
5. If the vector is not full then add the new pair to the end of the storage vector.

The removal of the first, low information pair in the storage vector, when the vector is full, applies a compression to the model. This compression has been argued to reduce the noise in the representations, known as syntagmatic and paradigmatic associations.

The paradigmatic measure proposed by the TE research (Symonds et al. 2012) shows how the indirect associations exist between concepts that are able to acceptability of the sentence (e.g., synonyms, or related verbs like eat-drink). It is the paradigmatic associations which have the greatest function for the task of LBD, because, in effect concepts \(A\) and \(C\) have a paradigmatic association via their common neighbour \(B\). The syntagmatic association for \(A\) and \(C\) in LBD should be zero (i.e., never seen together in a document). This makes the TE model well adapted to the task of performing LBD, as both steps in the open discovery mode, discussed in Section 3.1 can be completed in one formalism. The TE model’s formalism can be expressed as a conditional probability of concept \(C\) being suggested as a useful connection for a given \(A\) concept:

\[
P(C|A) = \gamma S_{par}(A, C) + (1 - \gamma) S_{syn}(A, C),
\]

where \(\gamma\) is a mixing parameter that combines the measures of paradigmatic and syntagmatic associations. The paradigmatic measure proposed by the TE research (Symonds et al. 2012) shows how the indirect relationship between concept \(A\) and \(C\) can be modelled via \(B\) concepts:

\[
s_{para}(A, B) = \sum_{i \in V} \max(f_{TCA} - f_{TCA}, f_{TCA}, f_{TCA})^2,
\]

where \(f_{TCA}\) is the unordered co-occurrence frequency of concepts \(B\) and \(A\), and the term \(f_{TCA}\) in the denominator penalises the paradigmatic score if \(A\) and \(C\) have a strong syntagmatic association.

For the TE model, the time complexity of the retrieval process would be the time complexity of the syntagmatic measure and paradigmatic measures combined. Using the syntagmatic and paradigmatic measures outlined in previous TE research (Symonds et al. 2012), the time complexity to perform the retrieval process would be \(T(n) = O(d_{sv} + d_{sv}^2)\). For the MAREC data set, the worst case time complexity of computing the \(C\) concepts would be \(T(n) = (1.25 \times 10^8) + (1.25 \times 10^8)^2 = 1.25 \times 10^8\). Two orders faster than HAL and RI. The substantially reduced time complexity in calculating the similarity of terms within the TE model stems from the fact that the time complexity of the TE model is independent on the vocabulary size. It is the only model in our investigation with this property and whose time complexity advantage over other models would increase as the vocabulary size of the collection increases. This result is achieved by the TE model because (i) the two steps of the retrieval process can be computed in one step within the TE model, and (ii) only terms in a small set of storage vectors need be considered in the calculations as terms not in the storage vector of \(A\) are considered to have no similarity with \(A\), and all terms not in the storage vectors of terms syntagmatically related to \(A\) are considered to have no paradigmatic associations with \(A\).
Past research using the TE model to perform synonym judgement has shown that low frequency concepts are not discriminated against (Symonds et al. 2012). This indicates that the compression within the TE model and the similarity measures appear to effectively manage frequency bias, and hence the concern raised by Kostoff (2008) for the task of LBD.

4 Discussion

The computational complexity of each model for the task of LBD is shown in Table 4, along with its efficiency on the MAREC document collection. The important finding from Table 4 is that the time complexity of computing similarity within the TE model is independent of the vocabulary size ($|V|$). This means that as the vocabulary size increases, the time complexity for computing similarities within the TE model does not. As computing similarities is likely to be performed many more times than building the vocabulary (which only occurs once), this property of the TE model becomes more important.

The superior overall efficiency of the TE model can more easily be seen when the time complexities for building the vocabulary representations and computing the open discovery process for each of the four models are graphed (Figure 2).

![Figure 2: Efficiency comparison of LSA, HAL, RI and TE on the MAREC dataset when considering the time complexity of building the vocabulary representations (Build vocab.) and computing the similarity of terms involved in the open discovery process (Retrieval).](image)

In the case where more documents are added to the collection dynamically, all models are reported to be able to update the representations with little additional overhead. It is also worth noting that the vocabulary size of the MAREC dataset was computed assuming only representations for single word terms were required. However, research indicates that including multi-word concepts in any semantic model is likely to be of value due to the compositional nature of meaning (Grefenstette & Sadrzadeh 2011). Including multi-word terms to the vocabulary and building representations for each will impact the complexity values above, through an increased in vocabulary size ($|V|$). This again highlights the benefits of using a corpus-based model whose complexity in computing similarity between vocabulary terms is independent on $|V|$.

4.1 The TE Model’s Paradigmatic Measure

An initial investigation into the effectiveness of the TE model’s paradigmatic measure, with minimal impact that do not display syntagmatic associations was tested on a data set based on the TREC 2011 MedTrack collection which consisted of clinical patient records. Following the procedure outlined by Koopman et al. (2012) the original textual documents were translated into UMLS medical concept identifiers using MetaMap, a biomedical concept identification system (Aronson & Lang 2010). After processing, the individual documents contained only UMLS concept ids. For example, the phrase congestive heart failure in the original document will be replaced with C0018802 in the new document.

The TE model was then run to build representations, and then a number of sample terms, reported in past LBD research, were investigated. These included Raynaud’s disease (C0034734) and Migraine (C0149331). The investigation demonstrated that of the top 800 concepts suggested by the TE model’s paradigmatic measure (Equation (4)) for the concept (C0034734) representing the term Raynaud’s disease, only 72 words (i.e., 9%) displayed any syntagmatic association with C0034734. The paradigmatic measure could be easily modified, with minimal impact on efficiency, to ensure any terms displaying syntagmatic association (i.e., exist in the storage vector of the target term) receive a paradigmatic score of zero.

![Table 5: Details of the reduced medline document collection, based on the TREC’11 MedTrack task, used to evaluate the initial effectiveness of the TE model in performing LBD.](image)

| Collection | $|D|$ | $|V|$ | $|C|$ |
|------------|------|------|------|
| Medline Concept | 17,198 | 64,946 | 94082894 |

This prototype investigation into the use of the TE model for LBD also found magnesium was returned for a target term of migraine. This demonstrates early support for the potential effectiveness of the TE model on the task of open discovery.

4.2 Heterogenous LBD

There is a growing trend for automated LBD models to be used in conjunction with other manual discovery processes across multiple literature sources (Kostoff 2008). The ability for automated tools, like distributional models to perform LBD across different knowledge sources, known as heterogenous LBD, may allow even faster progress in these areas. Heterogenous LBD would entail even larger combined data sets and increase the importance of automated tools being efficient. An emerging form of heterogeneous LBD is literature related discovery and innovation (LRDI). LRDI integrates LBD with innovation e.g., re-invigorating prior art (Kostoff 2012).
Complexity

Table 4: Complexity of the Latent Semantic Analysis (LSA), Hyperspace Analogue to Language (HAL), Random Indexing (RI), and the Tensor Encoding (TE) model for performing the building and retrieval steps for open discovery. Where $|V|$ is the size of the vocabulary ($|V| \approx 7.5 \times 10^6$ for MAREC), $|D|$ is the number of documents in the collection (|D| = 6.6 \times 10^{10}$ for MAREC), $d_l$ is the number of singular values used by LSA ($d_l = 300$), $s$ is the size of the context window ($s = 5$), $d_c$ is the dimensionality of the RI context vectors ($d_c$ is equal to 1,000 and 32,000 for storage and time complexity, respectively), and $d_{sv}$ is the dimensionality of the TE storage vectors ($d_{sv} = 1,000$).

<table>
<thead>
<tr>
<th>Model</th>
<th>Complexity</th>
<th>For MAREC collection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Storage Complexity</td>
<td>$M(n) = O(</td>
<td>V</td>
</tr>
<tr>
<td>LSA</td>
<td>$M(n) = O(</td>
<td>V</td>
</tr>
<tr>
<td>HAL</td>
<td>$M(n) = O(2</td>
<td>V</td>
</tr>
<tr>
<td>RI</td>
<td>$M(n) = O(</td>
<td>C(s)</td>
</tr>
<tr>
<td>TE</td>
<td>$M(n) = O(</td>
<td>C(s)</td>
</tr>
</tbody>
</table>

| Time complexity for building the vocabulary | $T_b(n) = O(|V|)$                                                             | $T_b(n) = 1.8 \times 10^{18}$ |
| LSA       | $T_b(n) = O(|C(s)|)$                                                        | $T_b(n) = 4.5 \times 10^{18}$ |
| HAL       | $T_b(n) = O(|C(s)|)$                                                        | $T_b(n) = 1.9 \times 10^{16}$ |
| RI        | $T_b(n) = O(|C(s)|)$                                                        | $T_b(n) = 1.25 \times 10^{8}$ |
| TE        | $T_b(n) = O(|C(s)|)$                                                        | $T_b(n) = 1.25 \times 10^{8}$ |

5 Conclusion

This paper has provided a computational complexity analysis of four successful corpus-based distributional models on the task of open discovery. These models provide a cost-effective method of identifying potentially novel, undiscovered connections between concepts within large document collections, such as those found online, and used within tasks such as literature-based discovery (LBD). These discoveries may ultimately lead to technological breakthroughs, or the ability to identify future disruptive innovations.

Our analysis finds that distributional approaches that store representations in fixed dimensions have a smaller memory footprint, and can allow faster computation of associations between vocabulary terms. Of particular significance is the finding that the TE model is well adapted to the task of open discovery in LBD due to its efficient method of storing representations and computing similarities from these representations. These features allow the process of open discovery to be computed with an efficiency that is independent of the vocabulary size. The findings of this work motivate a future evaluation of the TE model performing LBD based tasks, such as the discovery component of the emerging field of literature related discovery and innovation (LRDI).

The computational analysis carried out in this work contributes to the field of LBD, and more broadly information retrieval, by providing insights into the effectiveness of corpus-based models, which allows a more complete consideration of model’s performance to be achieved.

6 Acknowledgements

I would like to thank the QUT HPC team for the use of the computational resources to perform the initial experiments. Thanks also to Lance De Vine for feedback on this paper.

References


A Systematic Approach to Measuring Advertising Transparency Online: An Australian Case Study

Paul A. Watters
School of Engineering and Advanced Technology
Massey University, New Zealand

paul.watters@cantab.net

Abstract

Illicit sharing of infringing content, such as movies and TV, remains a persistent and ongoing threat to the viability of Australia’s creative industries. The revenue model that underpins torrent indexing and file locker sites which enable this sharing – like much of the World Wide Web – is based on advertising. Recent research has suggested that there has been a shift from mainstream to High-Risk advertising on these sites. In this study, advertising targeting Australians was analysed from the Top 500 Google-upheld DMCA complaints for movies and TV distributed by major Hollywood studios, with 10 sites from each complaint sampled for all ads displayed. It was found that only one site carried mainstream advertising with ads targeting Australian consumers. The policy implications of this result and future research directions, including methodology enhancements, are discussed.

Keywords: Infringing content, internet advertising, DMCA, internet safety.

1 Introduction

Online advertising has a 20 year long history (Medoff, 2000), progressing from simple ad banners displayed on a fixed rotation schedule, through to personalised, behavioural advertising networks, which use profiles of individual users to present the most “relevant” advertisements (McStay, 2011). Such technologies make extensive use of “tracking cookies” (Watters, 2012) and the linkages between advertising networks and cookies have recently been monitored and explored for the most popular websites in Australia (Herps et al, submitted). The most interesting result from this study was that the number of cookies stored on a user’s computer from any of the Top 50 most-visited sites for Australians ranged between 0 and 86. The sophistication and the extent to which user behaviour is tracked and experiences customised is only going to increase over time, as is the overall volume of advertising. Indeed, in 2012, online advertising spending in the US reached US$39.6b, exceeding the amount spent on traditional print advertising for the first time (eMarketer, 2012).

Furthermore, some companies are in a unique position to know “everything” about their customers. Google, for example, has the capacity to monitor almost all of the world’s information, including personal emails, YouTube movies, Android phones, news services, images, shopping, blogs and so on (Cleland, 2013). Through its acquisition of Doubleclick, Google controlled an estimated 69% of the online advertising market (Browser Media, 2008), however, the rise of social media advertising (especially through Facebook) has seen this reduce to 56% (Womack, 2013). Clearly, there is a potential confluence of capability and opportunity to maximise the number of “eyeballs” exposed to online ads.

What are the implications of this massive rise in advertising expenditure, which coincides with an increased ability for online advertising networks to be able to best “place” ads to suit specific customers? One particular type of website – those associated with file sharing of infringing content – appears to have wholeheartedly embraced advertising. Indeed, advertising revenues provide the commercial motivation for criminal syndicates to operate such ‘rogue’ web sites. While the connection between film piracy and organised crime has been explored elsewhere, in terms of direct revenues (Treverton et al, 2009), there has been far less publicity about the advertising revenues generated from sites that appear to offer infringing content for free, or at least, offer torrents that enable users to download such material. Certainly, the links between the underground economy and the internet have been criticised for facilitating sexual exploitation and human trafficking through organised crime – in the classic paper in this field, Hughes (2000) highlighted how global advertising and marketing of prostitution have led to increases in volume globally. Furthermore, Hughes identified that a lack of regulation of internet advertising was the key policy failure in preventing harm to women and children.

The Pirate Bay is one of the most popular sites for providing torrents to infringing content, and has been the subject of criminal proceedings against its operators in Sweden. In the 2009 trial of its operators, their expenses were estimated to be US$110,000 (Olsson, 2006; Kuprianko, 2009), with advertising revenues in the order of US$1.4m (Sundberg, 2009) – in other words, an extremely profitable business with gross margins of 1272%! A recent study (Detica, 2012) indicated that there are six different business models operating within the pirate site marketplace, ranging from advertisement and...
donation funding, through to subscriptions and freemium sites, where subscribers can gain faster access to illicit content by paying a subscription fee. 83% of the sites in that study operated using a central website. Selling advertising on file locker and torrent search sites is the major source of revenue for such sites. The Pirate Bay, for example, regularly features in the Top 50 sites accessed by Australians (as computed by alexa.com), and so it is a potentially attractive space for advertisers and ad networks, since the number of potential “eyeballs” is very high. Maximising “eyeballs” leads to clicking, which drives revenue for the ad networks (if they operate a Pay Per Click revenue model), and sales for the advertisers. A key question for advertisers and ad networks is the extent to which they wish to be associated with this type of activity; indeed, due to the complex algorithms which decide which ads to display to which users, advertisers may not be aware of every site that their ads are being displayed on.

Being able to quantify the scale of advertising on these sites is important, since informing and making advertisers aware of the integrity of the sites on which their ads are being displayed can then be undertaken. Advertisers will thus be able to make more informed choices about their use of online advertising networks (the companies who provide aggregation of space on web sites) who are supporting piracy by selling ad space on torrent and file locker sites. A recent set of best practice guidelines for ad networks to address piracy and counterfeiting have recently been released1, and early indications are that most of the world’s major web companies will participate.

There have been few systematic studies investigating the relationship between piracy and advertising, and most have been concerned with the impact of interventions to reduce piracy. For example, Sheehan et al (submitted) identified that increasing the perception of legal risk for downloading each page from the “top 500” complaints as assessed by a third-party (Google), the result could be that the January report achieved its goal of sensitising advertising networks, and that Google subsequently withdrew from placing ads on those sites. Alternatively, the variation could be due to biases inherent in studies using an observational methodology, including:

- Selection bias, in the way that infringing sites are selected. The study uses a single source (the Google Transparency Report of domains with the most DMCA takedown requests), rather than using a consensus technique which combines the ranks of several different data sources to provide the most accurate ranking. This type of triangulation is commonly used in observational studies as a form of triangulation;

- Information bias, since only one technique for collecting data is used (HTML and JavaScript code scraping), where other techniques may be more accurate or representative of advertising behaviour. For example, persistent cookies have been strongly associated with behavioural advertising, and the frequency of tracking cookies being stored by ad networks could provide an alternative measure of presence of significance. Yet the USC report does not analyse cookies at all; and

- Recall bias, since the data analysed was only from English-language websites and advertising networks which may potentially have a higher level of visibility than networks which operate in other geographic zones, languages, encoding types etc.

Also, the lack of detail in how measures like the “top 500” sites prevent the study results from being directly replicated, which would be the standard required for peer review by other researchers. By not providing this level of detail, the credibility of the USC report may be called into question by the very vocal critics of any research in the anti-piracy field.

In this paper, we present a more rigorous and fully replicable methodology which should provide a much clearer view of advertising network behaviour in different countries, jurisdictions, languages etc. In this initial study, we specifically target Australian users content produced and distributed by major Hollywood Studios; the methodology itself is sufficiently general that it could be applied to any country and any category, including music, computer games, e-books etc.

2 Methods

The main goal of the methodology is to identify the advertising networks and advertisers from a sample of DMCA complaints, which have been ranked in terms of the number of complaints upheld by Google (through their Transparency Report). These complaints typically relate to the availability of search results for a wide range of potentially infringing content; by only selecting the most complained about and subsequently upheld complaints as assessed by a third-party (Google), the results should be robust against criticisms that there is no proof that the sites in question were hosting torrents of infringing content or infringing content directly, in the case of a file locker site. The methodology operates by downloading each page from the “top 500” complaints


submitted to Google within the previous month, ordered by the number of upheld complaints. Since each DMCA notice can contain many thousands of individual URLs, a sampling procedure can be used to identify a representative subset of URLs, and the advertisements on each page can be downloaded along with their metadata. In the case of simple banner ads, it is then relatively easy to identify the advertisers concerned; in the case of each distinct advertisement, a rule can be generated using SQL or similar to identify all advertisements with the same metadata. However, some advertising networks use JavaScript obfuscation and a series of redirects to obscure the ultimate destination for the advertising banner; in this case, manual inspection must be performed, in the absence of a general purpose image/logo recognition system. The overall prevalence of a particular advertiser on each network can be then be computed and ordered by frequency.

Furthermore, it may be of interest to separate out “mainstream” advertisements as opposed to “High-Risk” advertising, since the Annenberg reports indicate a flight by mainstream advertising this year from sites that host infringing content. Advertisers who may otherwise be unable to place their ads on a mainstream site can then take advantage of increasing “eyeballs” by occupying display space. Results are thus reported for the High-Risk and mainstream categories, with the former including categories such as:

- Sex Industry, which includes adverts for:
  - o Penis length extension medication
  - o Fake personal/dating sites
  - o Pornography of various kinds
  - o Dating and “foreign bride” sites
- Online Gambling
- Malware, including
  - o Fake software incorporating Trojan horse malware (numerous alerts were raised by anti-virus software during the data collection process due to “drive by downloads” of malware)
  - o Fake anti-virus or anti-scamware
  - o Suspicious software such as fake video codecs or video players that replicate existing functions within Microsoft Windows. The purpose of such downloads is unclear, although it is possible that they could host Trojans or provide backdoor access to systems.
- Scams, as defined by Stabek et al (2010), such as:
  - o Premium rate SMS scams
  - o Fake competitions where no prizes are offered
  - o Investment scams
  - o Employment scams

The algorithm works as follows:

1. A data collection system is installed physically or logically to attract advertising for a specific geographical/country segment. For this study, Australia was selected.
2. The current Google Transparency Report
3 is downloaded, which lists all of the DMCA requests for the previous month. This list provides one means of identifying sites involved in sharing pirated material.
3. The dataset is sorted by the number of URLs removed, retaining the “top 500” DMCA requests (the request list) by complaint category. For this study, the complaint category was movies and TV shows; other complaint categories such as pirated software, adult material, music etc were excluded.
4. For each report in the request list first 10 URLs are extracted as a representative sample of all of the URLs contained within the report. This gives a total of 5,000 web pages to be downloaded (the sample).
5. Each of the 5,000 web pages in the sample is downloaded, and a screenshot is taken, showing the ads being served. Note that pop-up ads are not captured.
6. For each web page in the sample, the code blocks that contain advertising are parsed and extracted. This can be achieved by matching against the Easy List
4 (used by Adblock Plus for filtering), for known URL patterns and hostnames of advertisers. Some pages in the sample will have no ads, while others will have multiple ads.
7. For each advertising code block, the domain of the advertising network being used is identified, by stripping extraneous code and links from the code block, and counting the frequency of appearance of each ad network domain.
8. For each identified advertisement, an attempt is made to identify the actual advertiser, by analysing metadata, following the link and extracting the domain of the actual advertiser, or through visual inspection. A list of all identified advertisers is then generated.
9. For all “mainstream” advertising networks identified as present on web page, a further 100 samples of advertising are downloaded and added to any unseen advertisers to the identified list.

2.1 Definitions

- Internet Advertising. Ads are typically placed as “banners” on a website, which direct a user to another site when clicked. The contents of the ad are similar to a highway billboard, except that the can incorporate interactive elements such as animation. Ads on the same page are often rotated through a predetermined or random sequence, depending on the advertising plan that an advertiser has subscribed to. While some sites host and manage their own banners, most often, these are managed by a third-party advertising network. These ad networks act as an intermediary between an advertiser and many hundreds, thousands or millions of sites, allowing an advertiser to increase their reach to potential consumers while only dealing with a

https://www.google.com/transparencyreport/removals/copyright/data/
4 http://easylist.adblockplus.org/en/
single agency. Advertisers typically operate either a "pay per impression" or "pay per click" model, billing an advertiser every time a user views or clicks on a banner ad respectively.

- **Mainstream Advertising.** Mainstream ads are those placed by legitimate businesses that operate within the formal economy. Such businesses operate through a corporate structure and offer goods or services which fall outside the black market, grey market or underground economy.

- **High-Risk Advertising.** High-Risk ads are those promoting goods or services which fall outside the legitimate economy or white market, may be illegal or restricted within certain jurisdictions but not others, or may be fake or counterfeit. Examples include the sex industry, gambling and suspicious software/malware, such as anti-virus software which actually installs a Trojan Horse on a user’s system. Many of the ads are likely to fall into scam categories described by Stabek et al (2009).

- **Advertising Network.** Ad networks facilitate the placement of an advertiser’s ads on numerous websites according to a specific revenue model. Ad networks specialise in anticipating consumer’s needs and wants by building up profiles of users who click most frequently on certain ad categories on certain page themes, which can lead to more targeted, personalised, and relevant advertising. For the purposes of this paper, sites that host advertising on behalf of external / third-party advertisers are also grouped under this category, even if they only provide banners on sites within their own domain. For example, isohunt.com provides their own ad network exclusively for their own site, and not to other sites; they also host banners from other ad networks.

- **Internet Advertiser.** A business, government, association or individual that desires to sell goods or services, or provide information to, a target group of consumers. Internet advertising competes with traditional advertising for marketing budgets. Australia’s online advertising market is currently valued at $17.1b (Cameron, 2013).

- **Rogue Site.** A website which provides an index and search capability for torrents of infringing content, a “file locker” site which provides hosting for such material, or a “link site” which provides direct links to content on third party sites. The primary motivation for users visiting these websites is to access infringing content. These sites can all use advertising as either primary or secondary sources of income.

3 Results

After analysing the TV and movie DMCA reports from major Hollywood studios such as Fox, Warner Bros etc, only one site from the sample was found to be hosting mainstream advertising; all other sites were only hosting High-Risk advertising. From the 5,000 pages analysed in Step 4, a total of 12,638 distinct advertising blocks were identified in Step 6, giving an average 2.5276 ads per page. Postprocessing of the identified domains were performed to ensure that all ad blocks were correctly identified, for example, by removing port numbers that were included as part of a URL. 351 unique domains for advertising networks were identified, indicating an average 36.01 ads per network in the sample (keeping in the mind that the distribution – shown in Table 1’s Top 10 advertising networks - is non-uniform). Note that no merging of distinct services was performed, eg, the several domains of The Pirate Bay were not aggregated. Also, where a domain appears within an ad block, this is a technical definition as per the methodology in Steps 6 and 7, ie, if the site or known ad URL appears in the block, then it will be counted. This could include Facebook social plugins, for example, rather than Facebook ads.

<table>
<thead>
<tr>
<th>Advertising Network</th>
<th>Frequency</th>
<th>% of Ads</th>
</tr>
</thead>
<tbody>
<tr>
<td>propellerads.com</td>
<td>1,565</td>
<td>12%</td>
</tr>
<tr>
<td>adexprt.com</td>
<td>1,058</td>
<td>8%</td>
</tr>
<tr>
<td>fhserve.com</td>
<td>862</td>
<td>7%</td>
</tr>
<tr>
<td>isohunt.com</td>
<td>690</td>
<td>5%</td>
</tr>
<tr>
<td>filestube.com</td>
<td>597</td>
<td>5%</td>
</tr>
<tr>
<td>sumotorrent.com</td>
<td>583</td>
<td>5%</td>
</tr>
<tr>
<td>adcash.com</td>
<td>357</td>
<td>3%</td>
</tr>
<tr>
<td>friendlyDuck.com</td>
<td>332</td>
<td>3%</td>
</tr>
<tr>
<td>torrentco.com</td>
<td>327</td>
<td>3%</td>
</tr>
<tr>
<td>rtbpop.com</td>
<td>210</td>
<td>2%</td>
</tr>
</tbody>
</table>

3.1 High-Risk Advertising- Top 10 Ad Networks

The results for the breadth-first search (step 8) confirm that there are still mainstream advertisers prepared to support the distribution of infringing content. One exception was noted - while many of the file locker sites visited had no advertising slots at all, they were offering subscription packages of up to two years or pay-per-view packages for single titles. Others appeared to rely on both advertising and membership; eg, isohunt.com charges $1 per month for premium membership, as well as hosting ads.

Where advertising was hosted on torrent and file locker sites, it sometimes fell squarely into what can only

---

5 Note that some ad networks like isohunt.com and sumotorrent.com do not display their ads outside their own domain; they are ranked highly because of the high number of DMCA complaints against their site.
be described into High-Risk and often “scam” categories, typically advertising fake or harmful goods or services.

Table 2 contains a summary of the results from the Top 10 ad networks. There were 5,598 advertisements in this sub-sample of which 169 were distinct. Each of these advertisements was downloaded, visually inspected and categorised. The results indicate that the sex industry, malware, downloading sites, gambling or scams (including employment, investment and SMS premium rate) were the most popular distinct advertising types. The categories are summarised in Figure 1.

Table 2: High-Risk ad type frequencies by network

<table>
<thead>
<tr>
<th>Ad Network</th>
<th>Sex Industry</th>
<th>Malware</th>
<th>Downloading</th>
<th>Gaming/Gambling</th>
<th>Scams</th>
</tr>
</thead>
<tbody>
<tr>
<td>propellerds.com</td>
<td>5</td>
<td>29</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>adsexptr.com</td>
<td>7</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>fileserve.co.m</td>
<td>0</td>
<td>8</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>isohunt.co.m</td>
<td>0</td>
<td>2</td>
<td>9</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>filestube.com</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>sumotorrent.com</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>adcash.co.m</td>
<td>8</td>
<td>10</td>
<td>0</td>
<td>1</td>
<td>12</td>
</tr>
<tr>
<td>friendlyDuick.com</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>torrentco.com</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>rtbtops.co.m</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>TOTAL</td>
<td>23</td>
<td>49</td>
<td>18</td>
<td>3</td>
<td>15</td>
</tr>
</tbody>
</table>

An example of malware downloaded is provided by the advertising link http://isohunt.com/a/adclick.php?bannerid=493&zoneid= &source=btDetails &banner&dest=http://3A%2F%2Fp.ncdownloader.com%2 Fexact%2F%3F3DCannonball+Run+II+1984. Upon visiting this page, a download is initiated to the user’s computer containing the file Cannonball Run II. 1984.exe which is only 292K in size – much smaller than a typical video file of at least 700M. Running this file through the online scanner virscan.org – which analyses suspicious files using 36 different products – the file is verified as ADWARE/Adware.Gen (http://v.virscan.org/ADWARE/Adware.Gen.html) by AntiVir 8.2.10.202 and as Adware.Downware.1166 by ClamAV (http://v.virscan.org/Adware.Downware.1166.html). A review of the other known filenames associated with this malware indicates a typical strategy of associating a desirable filename with the malicious code, ie, using a filename that users desiring to download infringing content will click on, including Mortal Kombat - Complete Edition Crack (2013) Downloader.exe and Transformers 3 - Dark of the Moon (2011) [1080p].exe

3.2 Mainstream Advertising- All Sites

Table 3 contains the results of the step 8 results obtained by visually inspecting every advertisement in the sample (comprising 10 pages from each of the Google Ad Transparency Top 500 complaints) to identify whether it contained any mainstream advertising. Typically, a rogue site will have 3-4 ad panels, and in many cases, the ads were tailored to the local geographic context. In some cases, advertisements were blocked with an image stating the site was “blocked for Australians” indicating further evidence of geographic customisation for the advertising content. In some cases, domains associated with file sharing were “parked” and advertising displayed, even if no infringing content was actually displayed – especially where such sites had terms like “varez”, “anon” and “rapidshare” in their domain name.

Table 3: Australia-specific ads from breadth-first search

<table>
<thead>
<tr>
<th>Advertiser</th>
<th>Ad Network</th>
<th>Frequency</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Opencolleges.edu.au</td>
<td>Google Ad Services</td>
<td>23</td>
<td>14</td>
</tr>
<tr>
<td>Kia.com.au</td>
<td>Google Ad Services</td>
<td>21</td>
<td>13</td>
</tr>
<tr>
<td>Inspireeducation.com.au</td>
<td>Google Ad Services</td>
<td>15</td>
<td>9</td>
</tr>
<tr>
<td>Sommusic.com.au</td>
<td>Google Ad Services</td>
<td>12</td>
<td>8</td>
</tr>
<tr>
<td>Totalmusic.com.au</td>
<td>Google Ad Services</td>
<td>10</td>
<td>6</td>
</tr>
<tr>
<td>ANZ Bank</td>
<td>Google Ad Services</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>Optus</td>
<td>Google Ad Services</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>Luminosity.com</td>
<td>Google Ad Services</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>Suncorpbank.com.au</td>
<td>Google Ad Services</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>Thevocalistsway.com.au</td>
<td>Google Ad Services</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>Australianstockreport.com.au</td>
<td>Google Ad Services</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>Fxstrategies.com.au</td>
<td>Google Ad Services</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>Pizza Hut</td>
<td>elakiri.com</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>
The overall distribution of advertising agencies serving mainstream ads is shown in Table 4. Note that 87.42% of these were served by Google Ad Services (139 out of 159).

Only one site out of the 500 sampled consistently showed evidence of targeting Australian users through the presentation of mainstream advertising, even though the results from Table 3 indicate that there is a certain background level across a number of different sites. For example, the Pirate Bay often displays ads from the Exoclick ad network, but at times, it also displayed ads from other networks, including two ads from Walmart – clearly a mainstream advertiser. In a sense, this represents a type of leakage, since the mainstream ads (159 in total, across the entire sample of 12,638 ad panels) were such a small percentage of the overall ads displayed, which were overwhelmingly High-Risk. A breakdown by industry category is shown in Figure 2.

<table>
<thead>
<tr>
<th>Ad Network</th>
<th>Number</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Google Ad Services</td>
<td>139</td>
<td>87%</td>
</tr>
<tr>
<td>Unknown</td>
<td>8</td>
<td>5%</td>
</tr>
<tr>
<td>Speedy Ads</td>
<td>3</td>
<td>2%</td>
</tr>
<tr>
<td>Ads4vn.com</td>
<td>2</td>
<td>1%</td>
</tr>
<tr>
<td>Adhost2</td>
<td>2</td>
<td>1%</td>
</tr>
<tr>
<td>elakiri.com</td>
<td>2</td>
<td>1%</td>
</tr>
<tr>
<td>MediaMind</td>
<td>2</td>
<td>1%</td>
</tr>
<tr>
<td>Clicksor</td>
<td>1</td>
<td>1%</td>
</tr>
</tbody>
</table>

3.3 Mainstream Advertising- Depth First
The results for the depth-first search for the one site that contained many mainstream ads clearly targeting Australians, and met the criteria for mainstream advertising. The advertisers were “household names”. 100 page impressions were downloaded from a target page on each site, and the advertisers were manually identified, only if their logo or business name was clearly evident. Table 5 shows the results for the Top 20 Australia-specific advertisements for http://tehparadox.com/forum/f1/00/5Brg-su%5Djackreacher-2012-5Dp4yvizar-vizac-5277866/ - a copy of the Jack Reacher movie. This movie was not legally available on any of the sites found. 63% of the advertisements were specifically targeted at Australians.
Table 5: Australia-specific ads from depth-first search – Jack Reacher

<table>
<thead>
<tr>
<th>Advertiser</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quibids</td>
<td>37</td>
</tr>
<tr>
<td>Commonwealth Bank</td>
<td>19</td>
</tr>
<tr>
<td>AsiaRooms</td>
<td>15</td>
</tr>
<tr>
<td>NSW Lotteries</td>
<td>9</td>
</tr>
<tr>
<td>Marriott Hotels</td>
<td>4</td>
</tr>
<tr>
<td>Lumosity</td>
<td>3</td>
</tr>
<tr>
<td>iman Health Insurance</td>
<td>3</td>
</tr>
<tr>
<td>Travel Insurance Direct</td>
<td>2</td>
</tr>
<tr>
<td>AHM Insurance</td>
<td>2</td>
</tr>
<tr>
<td>Mitsubishi</td>
<td>1</td>
</tr>
<tr>
<td>Royal Automobile Association of South Australia</td>
<td>1</td>
</tr>
<tr>
<td>Clearly Contacts</td>
<td>1</td>
</tr>
<tr>
<td>RUOK Day</td>
<td>1</td>
</tr>
<tr>
<td>Le Meridien</td>
<td>1</td>
</tr>
<tr>
<td>Australian Super</td>
<td>1</td>
</tr>
</tbody>
</table>

4 Conclusion

The goal of this paper was to develop a systematic approach to analysing internet advertising, with a focus specifically on sites where DMCA complaints about movie and TV content were upheld by Google. The key findings from the analysis of the first Australian data set are discussed below:

- 99% of the ads were High-Risk; only 1% were mainstream.
- Only one site from the sample displayed only mainstream advertising; the remaining sites either had no ads or displayed only ads from High-Risk sources, or had a small number of mainstream ads.
- In the High-Risk ads, 46.49% were for malicious or suspected malicious code, while 20.18% were for the sex industry. A further 14.91% were for scams of various kinds, including premium rate SMS, investment and employment scams.
- The top ad networks serving ads to Australians include propellerads.com, adexpert.com and fhserv.com; while these may seem less mainstream, as the above results indicate, many ads from mainstream “household names” are being promoted through this means of advertising exclusively on 1 out of 500 sites in the sample, and a small number non-exclusively.
- Both breadth-first and depth-first searches reveal a significant number of household name brands in Australia choosing to advertise on sites and their pages which are promoting the distribution of infringing content (movies and TV shows). Further investigation is needed to uncover the mechanics of how these ads are selected to appear; are advertisers engaging directly with ad networks, or are ad networks operating at a wholesale level and distributing ads to other networks through a resale programme? Who, eventually, has control over the display of this type of advertising space?

- In the breadth-first search, top mainstream advertisers included Kia, Optus, ANZ Bank and Suncorp Bank.
- In the depth-first search, top mainstream advertisers were drawn from every sector in the Australian economy, including gambling companies (NSW Lotteries), car manufacturers (Mitsubishi), financial services (Commonwealth Bank), travel insurance (Travel Insurance Direct), health insurance (AHM), accommodation (Marriott Hotels), charity (RUOK Day), and optometrists (Clearly Contacts).

Drawing together these findings, some key lessons can be drawn:

- Advertisers need to take more ownership of where their advertising is ultimately displayed by negotiating better agreements – based around integrity – with their ad networks. Rather than further government regulation, establishing a code of conduct (such as the US industry is doing) would be a first step (Dredge, 2013). A set of best practices to be adopted by major web companies would even further isolate rogue websites, and ultimately, reduce the advertising revenue which in turns drives their ability to promote infringing content.

Facebook has recently responded to pressure from its advertisers to remove links to pages with offensive material under threat of a boycott (Cellan-Jones, 2013). In addition, Google recently acted to remove search results for pharmaceuticals without prescriptions (O’Donnell, 2013), after paying a $500 million fine 18 months previously. A recent study (Watters & Phair, 2012) indicated that the illicit drug trade is a growing problem online, as advertising to new customers is fast, easy, affordable and low risk, given that jurisdictional differences can be exploited by transnational organised crime. Rather than individual advertising networks responding on an ad-hoc basis, an industry wide code will ensure a consistent response across the board with a focus on integrity in advertising.

- However, any code of conduct must also be enforceable, be aimed at disrupting revenue streams for rogue sites, and not place a significant administrative burden on rightsholders. Another risk is that there will continue to be a shift of mainstream advertising away from rogue sites, and that High-risk advertising networks will simply fill the gap. Indeed, at this stage, none of the top 10 advertising networks supporting rogue websites are involved in the code of conduct project.

- Advertisers clearly need more transparency from ad networks about where their ads are being displayed, as most (if not all) would no doubt be very surprised about where their ads are being displayed. The potential for brand damage is enormous. In some cases, company names are also being employed without knowledge (eg, a number of Woolworths and Westfield $1,000 voucher ads were displayed on scam sites). To provide operational assurance, advertisers should implement systems to monitor the usage of their brand names and trademarks

6 http://www.bbc.co.uk/news/technology-23325627
on unauthorised sites. Existing brand protection services for corporates clearly need to consider the negative implications for mainstream advertisements appearing alongside the “scam” categories outlined earlier, as well as advertisers appearing to endorse the illegal distribution of infringing content.

• Future research should focus on developing better techniques for identifying sites hosting mainstream advertising on sites hosting infringing content, and then passing these across to more robust systems for extracting advertiser names. This is because many advertising networks use JavaScript obfuscation to try and hide the domain name and other identifying details of the advertisers. Short of implementing generic image recognition for brand names and logos, semi-supervised learning of patterns accompanied by expert judgements will provide the most accurate results over the short term.

• Finally, and perhaps most importantly, parents and educators need to be aware of that the sex industry and online gambling sites specifically target torrent search and file locker sites for advertising their services. Ads promoting scams, the sex industry and gambling compromised 37.72% of the ads examined. For example, upon visiting the “Top 100” page for the Pirate Bay, one employment scam was displayed (“I make $260 every day”) and one porn site (“Facebook of webcams”). However, upon clicking the “Porn” page, an animated sex ad is displayed (“LOCAL SLUTS WANT TO F**K. Why the F**K would you pay for sex? Sign Up and F**K”). There are absolutely no age warnings on these pages, and no attempt is made by the Pirate Bay to verify if users are adults. Parents need to be aware that this is the type of content that will be served up to their children, even if they are only intending to download torrent for music or less offensive content. The absence of traditional regulatory mechanisms for effectively controlling online content – including the Classification Board and Advertising Standards Bureau - mean that new subcultural norms are rapidly being established online, and these can have profoundly negative consequences; for example, a progression model of rising interest in child exploitation material has been linked to the rise of the online porn culture, particularly where young users are inadvertently exposed to pornography through advertising (Prichard et al, 2013).

5 References

Browser Media (2008). DoubleClick deal means Google controls 69% of the online ad market. Downloaded from http://www.browsermedia.co.uk/2008/04/01/doubleclick-deal-means-google-controls-69-of-the-online-ad-market/


Author Index

Arulanandam, Rexy, 31
Böttcher, Stefan, 3, 13
Brandenburg, Marc, 13
Bruza, Peter, 52
Cranefield, Stephen, iii, vii
Hartel, Rita, 3, 13
Maddalena, Eddy, 39
Mizzaro, Stefano, 39
Molfetas, Angelos, 21
Purvis, Maryam A., 31

Savarimuthu, Bastin Tony Roy, 31
Sitbon, Laurianne, 52
Symonds, Michael, 52
Thiele, Sebastian, 3
Trotman, Andrew, iii
Watters, Paul A., 61
Wirth, Anthony, 21
Yang, Jian, iii, vii
Zobel, Justin, 21
Volume 124 - Database Technologies 2012
Edited by Hai-Feng Chen, Flinders University, Australia and Rose T. Smith, University of South Australia, Australia. January 2012. 978-1-921770-07-4.

Volume 125 - Information Security 2012

Volume 126 - User Interfaces 2012
Edited by Hai-Feng Chen, Flinders University, Australia and Rose T. Smith, University of South Australia, Australia. January 2012. 978-1-921770-07-4.

Volume 127 - Parallel and Distributed Computing 2012
Edited by Junjun Chen, University of Technology, Sydney, Australia and Rajan Rajan, CSIRO ICT Centre, Australia. January 2012. 978-1-921770-08-1.

Volume 128 - Theory of Computing 2012

Volume 129 - Health Informatics and Knowledge Management 2012

Volume 130 - Conceptual Modelling 2012

Volume 133 - Australian System Safety Conference 2011

Volume 134 - Data Mining and Analytics 2012
Edited by Ying Zhang, Department of Immigration and Citizenship, Australia, Jinyong Li, University of South Australia, Australia, Paul J. Kennedy, University of Technology, Sydney, Australia and Peter Christen, Australian National University, Australia. December 2012. 978-1-921770-14-2.

Volume 135 - Computer Science 2013
Edited by Bruce Thomas, University of South Australia, Australia. January 2013. 978-1-921770-20-3.

Volume 136 - Computing Education 2013
Edited by Angela Carbone, Monash University, Australia and Jacqueline Whalley, AUT University, New Zealand. January 2013. 978-1-921770-21-0.

Volume 137 - Database Technologies 2013

Volume 138 - Information Security 2013

Volume 139 - User Interfaces 2013
Edited by Rose T. Smith, University of South Australia, Australia and Burkhard C. Wünche, University of Auckland, New Zealand. January 2013. 978-1-921770-24-1.

Volume 140 - Parallel and Distributed Computing 2013

Volume 141 - Theory of Computing 2013

Volume 142 - Health Informatics and Knowledge Management 2013

Volume 143 - Conceptual Modelling 2013

Volume 144 - The Web 2013