Document Clustering Algorithms, Representations and Evaluation for Information Retrieval

Christopher M. De Vries
chris@de-vries.id.au

Submitted For
Doctor of Philosophy in Computer Science
Electrical Engineering and Computer Science
Computational Intelligence and Signal Processing
Queensland University of Technology
Brisbane, Australia

Prior Degrees
Masters of Information Technology by Research
Queensland University of Technology
Brisbane, Australia

Bachelor of Information Technology
Software Engineering and Data Communications
Graduated with Distinction
Queensland University of Technology
Brisbane, Australia

Supervisor
Associate Professor Shlomo Geva
Electrical Engineering and Computer Science
Computational Intelligence and Signal Processing
Queensland University of Technology
Brisbane, Australia

Associate Supervisor
Associate Professor Andrew Trotman
Department of Computer Science
University of Otago
Dunedin, New Zealand

2014
The purpose of computing is insight, not numbers.

Abstract

Digital collections of data continue to grow exponentially as the information age continues to infiltrate every aspect of society. These sources of data take many different forms such as unstructured text on the world wide web, sensor data, images, video, sound, results of scientific experiments and customer profiles for marketing. Clustering is an unsupervised learning approach that groups similar examples together without any human labeling of the data. Due to the very broad nature of this definition, there have been many different approaches to clustering explored in the scientific literature.

This thesis addresses the computational efficiency of document clustering in an information retrieval setting. This includes compressed representations, efficient and scalable algorithms, and evaluation with a specific use case for increasing efficiency of a distributed and parallel search engine. Furthermore, it addresses the evaluation of document cluster quality for large scale collections containing millions to billions of documents where complete labeling of a collection is impractical. The cluster hypothesis from information retrieval is also tested using several different approaches throughout the thesis.

This research introduces novel approaches to clustering algorithms, document representations, and clustering evaluation. The combination of document clustering algorithms and document representations outlined in this thesis are able to process large scale web search data sets. This has not previously been reported in the literature without resorting to sampling and producing a small number of clusters. Furthermore, the algorithms are able to do this on a single machine. However, they can also function in a distributed and parallel setting to further increase their scalability.

This thesis introduces new clustering algorithms that can also be applied to problems outside the information retrieval domain. There are many large data sets being produced from advertising, social networks, videos, images, sound, satellites, sensor data, and a myriad of other sources. Therefore, I anticipate these approaches will become applicable to more domains, as larger data sets become available.

Keywords
document clustering, representations, algorithms, evaluation, information retrieval, clustering evaluation, clustering, cluster based search, hierarchical clustering, vector quantization, vector space model, dimensionality reduction, compressed representations, document signatures, random projections, random indexing, topology, hashing, machine learning, unsupervised learning, search, distributed search, similarity measures, pattern recognition, efficiency, indexing, text mining, signature files, distributed information retrieval, vector space IR, K-tree, EM-tree, TopSig, cluster hypothesis, large scale learning, INEX, Wikipedia, XML Mining, retrieval models, bit vectors
Acknowledgments

Many thanks go to my supervisor, Shlomo, who has provided valuable direction, advice and insights into this line of research. His advice and direction have been essential in ensuring the success of this research. I wish to thank my associate supervisor, Andrew, for providing feedback on many of the publications contained in this thesis. I wish to thank my family and friends for continued support during my candidature. I wish to thank all of the co-authors on the papers included in this thesis for their feedback and contributions. Finally, I thank QUT for providing an opportunity for me to undertake this research and the High Performance Computing center for supporting the Big Data Lab where many of the experiments were run.

External Contributions

All papers where I am first named author have been written entirely by myself. For papers where I am not first named author, I have made contributions to each of the papers. In the TopSig paper I wrote all material relating to document clustering and contributed to other sections. In the INEX 2009 overview paper I performed the analysis of results and created the graphs for the paper. The list of contributions I have made has been listed in Chapter 4 as research questions.

Statement of Original Authorship

The work contained in this thesis has not been previously submitted to meet requirements for an award at this or any other higher education institution. To the best of my knowledge and belief, the thesis contains no material previously published or written by another person except where due reference is made.

QUT Verified Signature

Signature August 26, 2014
# Contents

1 Introduction ........................................ 12  
   1.1 Document Clustering ................................ 13  
   1.2 Representations .................................... 13  
   1.3 Algorithms ......................................... 13  
   1.4 Evaluation ......................................... 14  
   1.5 Information Retrieval ............................... 15  
   1.6 How To Read This Thesis ......................... 16  
   1.7 Outline ........................................... 16  

2 High Level Overview ................................ 18  
   2.1 Publication Information ......................... 19  

3 Background ......................................... 22  
   3.1 Document Clustering ............................... 22  
      3.1.1 Evaluation .................................... 25  
   3.2 Document Representations ....................... 27  
   3.3 Ad Hoc Information Retrieval.................... 28  
      3.3.1 Collection Distribution and Selection .......... 30  
      3.3.2 Evaluation .................................... 31  
   3.4 Cluster Hypothesis ................................ 32  
      3.4.1 Measuring the Cluster Hypothesis ............ 33  
      3.4.2 Cluster Hypothesis in Ranked Retrieval ....... 34  
      3.4.3 Sub-document Cluster Hypothesis in Ranked Retrieval 36  

4 Research Questions ................................ 37  
   4.1 Representations ................................... 37  
   4.2 Algorithms ....................................... 39  
   4.3 Evaluation ....................................... 42  
   4.4 Information Retrieval ......................... 44  

5 Representations .................................... 46  
   5.1 TopSig: topology preserving document signatures . 46  
   5.2 Pairwise Similarity of TopSig Document Signatures 47  
   5.3 Clustering with Random Indexing K-tree and XML Structure 47
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>Algorithms</td>
<td>48</td>
</tr>
<tr>
<td>6.1</td>
<td>TopSig: topology preserving document signatures</td>
<td>48</td>
</tr>
<tr>
<td>6.2</td>
<td>Clustering with Random Indexing K-tree and XML Structure</td>
<td>48</td>
</tr>
<tr>
<td>6.3</td>
<td>Distributed Information Retrieval: Collection Distribution, Selection and the Cluster Hypothesis for Evaluation of Document</td>
<td>49</td>
</tr>
<tr>
<td>6.4</td>
<td>The EM-tree algorithm</td>
<td>49</td>
</tr>
<tr>
<td>7</td>
<td>Evaluation</td>
<td>50</td>
</tr>
<tr>
<td>7.1</td>
<td>Overview of the INEX 2009 XML Mining track: Clustering and Classification of XML Documents</td>
<td>50</td>
</tr>
<tr>
<td>7.2</td>
<td>Overview of the INEX 2010 XML Mining track: Clustering and Classification of XML Documents</td>
<td>51</td>
</tr>
<tr>
<td>7.3</td>
<td>Document Clustering Evaluation: Divergence from a Random Baseline</td>
<td>51</td>
</tr>
<tr>
<td>8</td>
<td>Information Retrieval</td>
<td>52</td>
</tr>
<tr>
<td>8.1</td>
<td>Distributed Information Retrieval: Collection Distribution, Selection and the Cluster Hypothesis for Evaluation of Document</td>
<td>52</td>
</tr>
<tr>
<td>9</td>
<td>Overview of the INEX 2009 XML Mining track: Clustering and Classification of XML Documents</td>
<td>53</td>
</tr>
<tr>
<td>9.1</td>
<td>Evaluation</td>
<td>53</td>
</tr>
<tr>
<td>10</td>
<td>Clustering with Random Indexing K-tree and XML Structure</td>
<td>69</td>
</tr>
<tr>
<td>10.1</td>
<td>Representations</td>
<td>69</td>
</tr>
<tr>
<td>10.2</td>
<td>Algorithms</td>
<td>69</td>
</tr>
<tr>
<td>11</td>
<td>Overview of the INEX 2010 XML Mining track: Clustering and Classification of XML Documents</td>
<td>80</td>
</tr>
<tr>
<td>11.1</td>
<td>Evaluation</td>
<td>80</td>
</tr>
<tr>
<td>12</td>
<td>TopSig: topology preserving document signatures</td>
<td>96</td>
</tr>
<tr>
<td>12.1</td>
<td>Representations</td>
<td>96</td>
</tr>
<tr>
<td>12.2</td>
<td>Algorithms</td>
<td>97</td>
</tr>
<tr>
<td>13</td>
<td>Document Clustering Evaluation: Divergence from a Random Baseline</td>
<td>110</td>
</tr>
<tr>
<td>13.1</td>
<td>Evaluation</td>
<td>110</td>
</tr>
<tr>
<td>14</td>
<td>Pairwise Similarity of TopSig Document Signatures</td>
<td>119</td>
</tr>
<tr>
<td>14.1</td>
<td>Representations</td>
<td>119</td>
</tr>
<tr>
<td>15</td>
<td>Distributed Information Retrieval: Collection Distribution, Selection and the Cluster Hypothesis for Evaluation of Document</td>
<td>127</td>
</tr>
<tr>
<td>15.1</td>
<td>Algorithms</td>
<td>127</td>
</tr>
<tr>
<td>15.2</td>
<td>Information Retrieval</td>
<td>128</td>
</tr>
</tbody>
</table>
List of Figures

2.1 A Map of Contributions in this Thesis ........................................... 19
9.1 The YAGO Category Distribution .................................................. 56
9.2 Purity and NCCG performance of different teams using the entire dataset ....................................................... 61
9.3 Purity performance of different teams using the subset data .......... 62
9.4 NCCG performance of different teams using the subset data ........ 63
9.5 The supervised classification task ................................................... 64
10.1 Random Indexing Example ............................................................. 73
11.1 Complicated and Noisy Wikipedia Category Graph ..................... 84
11.2 Algorithm to Extract Categories from the Wikipedia ................... 84
11.3 Legend ......................................................................................... 91
11.4 Micro Purity ................................................................................. 92
11.5 Micro Negentropy ......................................................................... 92
11.6 NMI ............................................................................................. 92
11.7 NCCG ........................................................................................ 93
12.1 RMSE Drop Precision .................................................................. 101
12.2 INEX 2009 Precision vs Recall .................................................... 104
12.3 INEX 2009 P@10n ...................................................................... 105
12.4 Wall Street Journal P@10 .................................................................. 105
12.5 INEX 2009 P@10 by Topic .............................................................. 105
12.6 INEX 2010 Micro Purity ................................................................. 107
12.7 INEX 2010 NCCG ...................................................................... 107
12.8 INEX 2010 Clustering Speed Up ................................................... 107
13.1 A Clustering Produced by a Clustering Algorithm ....................... 115
13.2 A Random Baseline Distributing Documents into Buckets the Same Size as a Clustering .............................................. 115
13.3 Legend .......................................................................................... 115
13.4 Purity ........................................................................................... 116
13.5 Purity Subtracted from a Random Baseline ................................ 116
13.6 NCCG .......................................................................................... 117
13.7 NCCG Subtracted from a Random Baseline ................................ 117
List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.1</td>
<td>Top-10 Category Distribution</td>
<td>56</td>
</tr>
<tr>
<td>10.1</td>
<td>K-tree Clusters</td>
<td>75</td>
</tr>
<tr>
<td>10.2</td>
<td>Clusters</td>
<td>78</td>
</tr>
<tr>
<td>11.1</td>
<td>Relevant Documents Missing from the XML Mining Subset</td>
<td>82</td>
</tr>
<tr>
<td>11.2</td>
<td>XML Mining Categories</td>
<td>85</td>
</tr>
<tr>
<td>11.3</td>
<td>Type I and II Classification Errors</td>
<td>93</td>
</tr>
<tr>
<td>11.4</td>
<td>Classification Results</td>
<td>94</td>
</tr>
<tr>
<td>12.1</td>
<td>Detailed Evaluation of 36 clusters</td>
<td>107</td>
</tr>
<tr>
<td>14.1</td>
<td>Number of Signatures Generated by TopSig</td>
<td>122</td>
</tr>
<tr>
<td>14.2</td>
<td>Nearest Neighbours Expected from $cdf_1$</td>
<td>124</td>
</tr>
<tr>
<td>14.3</td>
<td>Nearest Neighbours Expected from $cdf_2$</td>
<td>124</td>
</tr>
<tr>
<td>14.4</td>
<td>Signatures Expected within $\frac{2}{3}$</td>
<td>125</td>
</tr>
<tr>
<td>16.1</td>
<td>INEX 2010 XML Mining collection – Convergence</td>
<td>181</td>
</tr>
<tr>
<td>16.2</td>
<td>ClueWeb 2009 Category B – 104,000 Clusters</td>
<td>185</td>
</tr>
<tr>
<td>16.3</td>
<td>INEX 2010 XML Mining Collection – Streaming EM-tree vs TSVQ</td>
<td>188</td>
</tr>
</tbody>
</table>
List of Abbreviations

TopSig ........... Topology preserving document Signatures
IR ................. Information Retrieval
CPU ............... Central Processing Unit
LSA ............... Latent Semantic Analysis
NMF ............... Non-negative Matrix Factorization
DCT ............... Discrete Cosine Transform
SVD ............... Singular Value Decomposition
EM ............... Expectation Maximization
RMSE ............. Root Mean Squared Error
TSVQ ............ Tree Structured Vector Quantization
INEX ............. IInitiative for the Evaluation of XML retrieval
XML ............. eXtensible Markup Language
BM25 ............. Okapi Best Match 25
CBM625 ........ Cluster Best Match 625
NCCG ........... Normalized Cumulative Cluster Gain
TREC ............. Text REtrieval Conference
CLEF ........ Conference and Labs of the Evaluation Forum
NTCIR ........ NII Testbed and Community for Information access Research
MAP ............. Mean Average Precision
NMI ............... Normalized Mutual Information
TF-IDF .......... Term Frequency - Inverse Document Frequency
RI ............... Random Indexing
HITS ............. Hyperlink Induced Topic Search
VSM ............. Vector Space Model
TSM ............. Tensor Space Model
Chapter 1

Introduction

Digital collections of data continue to grow exponentially as the information age continues to infiltrate every aspect of society. These sources of data take many different forms such as unstructured text on the world wide web, sensor data, images, video, sound, results of scientific experiments and customer profiles for marketing. Clustering is an unsupervised learning approach that groups similar examples together without any human labeling of the data. Due to the very broad nature of this definition, there have been many different approaches to clustering explored in the scientific literature.

This thesis addresses the computational efficiency of document clustering in an information retrieval setting. This includes compressed representations, efficient and scalable algorithms, and evaluation with a specific use case for increasing efficiency of a distributed and parallel search engine. Furthermore, it addresses the evaluation of document cluster quality for large scale collections containing millions to billions of documents where complete labeling of a collection is impractical. The cluster hypothesis from information retrieval is also tested using several different approaches throughout the thesis. This thesis also introduces new clustering algorithms that can be applied to problems outside the information retrieval domain.

I briefly introduce the five main areas relating to the thesis. These are document clustering, representations, algorithms, evaluation and information retrieval. A more detailed introduction is contained in the high level overview, background and research questions. The results are then presented as publications for each research area.
1.1 Document Clustering

Document clustering analyses written language in unstructured text to place documents into topically related groups or clusters. Documents such as web pages are automatically grouped together so that pages talking about the same concepts are in the same cluster and those talking about different concepts are in different clusters. This is performed in an unsupervised manner where there is no manual labeling of the documents for these concepts, topics or other semantic information. All semantic information is derived from the documents themselves. The core concept that allows this to happen is the definition of a similarity between two documents. An algorithm uses this similarity measure and optimizes it so that the most similar documents are placed together.

1.2 Representations

Documents are often represented using high dimensional vectors where each term in the vocabulary of the collection represents a dimension. This allows the definition of similarity between documents using geometric measures such as Euclidean distance and cosine similarity. This thesis explores the use of TopSig document signatures for document clustering. TopSig produces a compressed representation of document vectors using random projections and numeric quantization. These document signatures preserve the topology of the document vectors. The properties of the topology of these document signatures have been investigated. TopSig vectors are represented as dense bit strings or bit vectors that can be operated on 64 dimensions at a time using modern 64-bit digital processors. This leads to gains in computational efficiency when comparing entire document vectors.

1.3 Algorithms

Clustering is often thought of as unfeasible for large scale document collections. Therefore, the ability for clustering algorithms to scale to large collections is of particular interest. Algorithms with an $O(n^2)$ time complexity pose problems when scaling to collections containing millions to billions of documents. This thesis explores the scalability of clustering algorithms in the setting of information retrieval.

A novel variant of the k-means algorithm is presented that operates directly with TopSig bit vectors. It allows a one to two magnitude performance increase over traditional sparse vector approaches. It does this without reducing the quality of the clusters produced. The K-tree algorithm uses the k-means al-
algorithm to perform splits in its tree structure. Building upon this result, the TopSig K-tree document clustering approach can scale to clustering 50 million documents, into over one hundred thousands of clusters, on a single machine, in a single thread of execution, in 10 hours. Clustering of documents at this scale has not previously been reported in the literature.

The EM-tree algorithm is introduced and it is shown to have further computational advantages over the K-tree when using the bit vectors produced by TopSig. Furthermore, it is much more amenable to parallel and distributed implementations due to the batch nature of its optimization process. The convergence of this new algorithm has been proven. The EM-tree also has advantages over previous similar tree structured clustering algorithms when the data set being clustered exceeds that of main memory and must be streamed from disk.

1.4 Evaluation

Category based document clustering evaluation does not have a specific use case. Clusters are compared to a known solution referred to as the ground truth. The ground truth is determined by humans who assign each document to one or more classes based on the topics they believe exist in a document. In this thesis, I offer a novel and alternative evaluation using ad hoc relevance judgments from information retrieval evaluation forums. This approach is motivated by the cluster hypothesis, which states that documents relevant to information needs tend to be more similar to each other than non-relevant documents – the relevant documents tend to cluster together. Therefore, if the cluster hypothesis holds, relevant documents for a given query will tend to appear in relatively few document clusters. This evaluation tests document clustering for a specific use case. It evaluates how effective a clustering is at increasing the throughput of an information retrieval system by only searching a subset of a document collection using collection selection. Using ad hoc relevance judgments for evaluation overcomes issues with labeling document collections containing millions to billions of documents. This has already been addressed for the evaluation of ad hoc information retrieval via the use of pooling. Furthermore, I suggest that evaluation of document clustering via the cluster hypothesis from information retrieval allows for a general evaluation of clustering. If the cluster hypothesis does not hold for a particular clustering, then the clustering is not likely to be useful in other contexts besides ad hoc information retrieval.

Evaluation of document clustering using ad hoc information retrieval has taken place at the INEX XML Mining track in 2009 and 2010. It is a collaborative forum where different groups of researchers work on different approaches to classification and clustering of semi-structured XML documents. In 2010, a new
approach to extracting category information from the Wikipedia was introduced that finds categories using shortest paths through the noisy and sometimes nonsensical category graph. It is motivated by Occam’s Razor where the simplest solution is often the best. In this case, the shortest paths through the category structure are simplest. The Normalized Cumulative Cluster Gain evaluation measure was introduced in this forum. It ranks clusters in an optimal order using ad hoc relevance judgments from the ad hoc track at INEX. Therefore, it is referred to as an “oracle” collection selection approach. It represents an upper bound on ranking of clusters for collection selection given a clustering and a set of queries and their relevance judgments. This evaluation indicated that as the number of clusters of a collection increased, less of the collection had to be searched to retrieve relevant documents.

The divergence from a random baseline meta-evaluation measure was introduced to deal with ineffective clusterings. It takes any existing measure of cluster quality, internal or external, and adjusts it for random chance by constructing a random baseline that has the same cluster size distribution as the clustering being evaluated. Each clustering being evaluated has its own baseline where the cluster size distribution matches. Ineffective clusterings exploit a particular cluster quality measure to assign high quality scores to clusterings that are of no particular use. Take the NCCG measure as an example. If one cluster contains the entire document collection, besides each other cluster containing one document, then this one large cluster will likely contain all the relevant documents and always be ranked first by the “oracle” collection selection approach. Therefore, it achieves the highest score possible but the entire collection still has to be searched. A random baseline with the same cluster size distribution will achieve the same score. Random guessing is not work and therefore this ineffective clustering is assigned a score of zero. When adjusting the internal measure of cluster quality, RMSE, or distortion, the divergence from random baseline approach provides a clear optimum for the number of clusters with respect to RMSE that is not apparent otherwise.

1.5 Information Retrieval

Web search engines have to deal with some of the largest data sets available. Furthermore, they are frequently updated and have a high query load. As document clustering places similar documents in the same cluster, the cluster hypothesis supports the notion that only a small fraction of document clusters need to be searched to fulfill a users information need when submitting a query to a search engine. Therefore, clustering of large document collections distributed over many machines can be exploited, so that only a fraction of the
collection has to be searched for a given query. The query can be dispatched to machines containing relevant parts of the index for a particular query. Within each individual machine, clustering can reduce the number of documents that must be searched. Furthermore, fine grained clustering allows the cluster hypothesis to be better exploited in terms of efficiency in a search engine. Using the TopSig K-tree and CBM625 cluster ranking approach outlined in this thesis, only 0.1% of the 50 million ClueWeb 2009 Category B document collection has to be searched. This is a 13 fold improvement over the previously best known approach.

1.6 How To Read This Thesis

This thesis is presented as a thesis by publication. One of the first things you may notice is repeated material between papers. Each paper has been published with relevant material for completeness. This way a reader does not have to refer to several different papers to get the whole picture for the results presented. Therefore, as you encounter repeated material, I encourage you to skip these parts if you have already read them.

The overall structure of the contributions is described in the high level overview section of the thesis in Chapter 2. The aim of this section is to allow the reader to piece together a high level overview of all the contributions in the thesis and how they fit together and build upon each other or support each other. The problems of knowledge representation, learning algorithms, their evaluation in an information retrieval context and applying them to ad hoc information retrieval are all highly inter-related. Each of the problems feed into problems in different areas.

1.7 Outline

Each of the major areas of contribution are introduced. These are representations, algorithms, evaluation, and, information retrieval. The relevant parts of each paper are highlighted in Chapters 5 through 8.

Conversely, each paper is introduced and it is explained how it fits into the topics of representations, algorithms, evaluation, and, information retrieval in Chapters 9 through 16.

I hope that this will allow you to understand the contributions from both points of view. From the abstract concepts of representations, algorithms, evaluation, and, information retrieval. And, also from their concrete instantiations in each of the papers and the associated open source software, which often touch
multiple of the abstract concepts. This way you can have your cake and eat it too.

In Chapter 2, the overall structure of the contributions are mapped out.

In Chapter 3, relevant research and background information related to the thesis is presented.

In Chapter 4, the research questions addressed by the research are defined. This chapter also provides a more detailed overview of how all the publications fit together.

In Chapter 5, the relevant sections of the papers to do with representations are highlighted.

In Chapter 6, the relevant sections of the papers to do with algorithms are highlighted.

In Chapter 7, the relevant sections of the papers to do with evaluation are highlighted.

In Chapter 8, the relevant sections of the papers to do with information retrieval are highlighted.

In Chapters 9 to 16, each paper is introduced and the reverse mapping of the previous four chapters is explained. This is how each paper maps onto the concepts of Representation, Algorithms, Evaluation, and, Information Retrieval.

In Chapter 17, the thesis is concluded and the implications of the research are discussed. The potential directions for further research are highlighted.
Chapter 2

High Level Overview

This thesis presents many inter-related results that build upon each other. New document representations are used for clustering. New clustering algorithms are used to improve ad hoc search. The INEX evaluation motivated the use of fine grained clusters for ad hoc search. A map of the contributions of this thesis are contained in Figure 2.1. Each arrow indicates that the following contribution is built upon the results in the previous contribution. Additionally, the relevant papers are listed in the bottom right of the relevant contribution. These correspond to the numbers listed in the bibliography. I highlight some examples from the map in the following paragraphs.

The TopSig representation was built upon random projections which lead to the TopSig k-means algorithm. Document clusters using TopSig k-means were used to evaluate the cluster hypothesis for ad hoc retrieval in the INEX evaluation forum. The results of the INEX evaluation indicated that fine grained clusters supported better ranking of relevant documents in clusters. Meanwhile, the TopSig K-tree and EM-tree algorithms were devised for scaling clustering of 50 million documents into hundreds of thousands of clusters in a single thread of execution. The combined result of the INEX evaluation and scalable clustering approaches fed into the CBM625 approach to cluster-based retrieval. This led to a 13 fold decrease in the number of documents retrieved in a cluster-based search engine over the previous best known result on the 50 million document ClueWeb09 Category B test collection.

The EM-tree algorithm provides further opportunities for efficiency and scaling of algorithms when using the TopSig representation. The EM-tree algorithm can optimize any cluster tree as input and therefore was combined with sampling and the TSVQ algorithm to produce lower distortion clusters in less time the TopSig K-tree approach. Furthermore, the batch optimization of a cluster tree
makes the EM-tree algorithm particularly suitable for distributed, streaming and parallel implementations of the algorithm. Based upon the results of the single threaded implementation, it is expected that the streaming, distributed and parallel implementation of the EM-tree algorithm will be able to cluster the entire searchable Internet of 45 billion documents into millions of clusters. As this scale of document clustering has not previously been possible, there are many unknown applications for clustering of the entire searchable Internet.

2.1 Publication Information

This thesis presents 8 papers. 6 conference or workshop papers have been peer reviewed and accepted for publication. 2 journal articles have been submitted for peer review. The papers are listed below in chronological order.
Overview of the INEX 2009 XML Mining track: Clustering and Classification of XML Documents

– Topics → Evaluation
– Chapter [7]

This paper appears in the proceedings of the INEX workshop in the Springer Lecture Notes on Computer Science. The paper was peer reviewed. It introduces the use of ad hoc relevance judgements for evaluating document clustering in the context of its use.

Clustering with Random Indexing K-tree and XML Structure

– Topics → Representations, Algorithms
– Chapter [10]

This paper appears in the proceedings of the INEX workshop in the Springer Lecture Notes on Computer Science. The paper was peer reviewed. It introduces approaches to clustering documents using XML structure and content using the combination of Random Indexing and K-tree.

Overview of the INEX 2010 XML Mining track: Clustering and Classification of XML Documents

– Topics → Evaluation
– Chapter [11]
– C.M. De Vries, R. Nayak, S. Kutty, S. Geva, and A. Tagarelli.

This paper appears in the proceedings of the INEX workshop in the Springer Lecture Notes on Computer Science. The paper was peer reviewed. This paper builds upon the evaluation of document clustering using ad hoc relevance judgements in the previous overview paper.

TopSig: topology preserving document signatures

– Topics → Representation, Algorithms
– Chapter [12]
– S. Geva, and C.M. De Vries.

This paper appears in the proceedings of the 20th ACM Conference on Information and Knowledge Management. The paper was peer reviewed. This paper introduces the TopSig representation and applies it to ad hoc retrieval and document clustering.

Document Clustering Evaluation: Divergence from a Random Baseline
This paper investigates an evaluation approach in detail that was introduced to differentiate pathological clusterings at INEX.

**Pairwise Similarity of TopSig Document Signatures**

This paper appears in the proceedings of the Seventeenth Australasian Document Computing Symposium in 2012 held at the University of Otago in New Zealand. The paper was peer reviewed. This paper investigates the topology of TopSig document signatures as a means to explain their performance characteristics as a retrieval model.

**Distributed Information Retrieval: Collection Distribution, Selection and the Cluster Hypothesis for Evaluation of Document Clustering**

This journal article has been submitted to the ACM Transactions on Information Systems for peer review. This paper investigates using scalable clustering algorithms to produce fine grained clusters of web scale document collections to reduce the number of documents that must be inspected by an information retrieval system.

**The EM-tree Algorithm**

This journal article has been submitted to the Journal of Machine Learning Research for peer review. This paper introduces the EM-tree algorithm and presents a proof for its convergence. It demonstrates improvements over K-tree in clustering document signatures in terms of quality and run-time efficiency.
This chapter introduces document clustering, clustering algorithms, ad hoc information retrieval and the cluster hypothesis. These are the primary areas of research investigated in this thesis. For an extensive review of clustering algorithms see “Pattern Classification” by Duda et al. [74]. For an extensive review of information retrieval, including document clustering, ad hoc information retrieval, the cluster hypothesis and their evaluation, see “Introduction to Information Retrieval” by Manning et al. [163].

3.1 Document Clustering

Document clustering is used in many different contexts, such as exploration of structure in a document collection for knowledge discovery [214], dimensionality reduction for other tasks such as classification [136], clustering of search results for an alternative presentation to the ranked list [102] and pseudo-relevance feedback in retrieval systems [135].

Recently there has been a trend towards exploiting semi-structured documents [172] [67]. This uses features such as the XML tree structure and hyperlink graphs to derive data from documents to improve the quality of clustering.

Document clustering groups documents into topics without any knowledge of the category structure that exists in a document collection. All semantic information is derived from the documents themselves. It is often referred to as unsupervised clustering. In contrast, document classification is concerned with the allocation of documents to predefined categories where there are labeled examples. Clustering for classification is referred to as supervised learning where a classifier is learned from labeled examples and used to predict the classes of unseen documents.
The goal of clustering is to find structure in data to form groups. As a result, there are many different models, learning algorithms, encoding of documents and similarity measures. Many of these choices lead to different induction principles [77] which result in discovery of different clusters. An induction principle is an intuitive notion as to what constitutes groups in data. For example, algorithms such as k-means [157] and Expectation Maximization [64] use a representative based approach to clustering where a prototype is found for each cluster. These prototypes are referred to as means, centers, centroids, medians and medoids [77]. A similarity measure is used to compare the representatives to examples being clustered. These choices determine the clusters discovered by a particular approach.

A popular model for learning with documents is the Vector Space Model (VSM) [198]. Each dimension in the vector space is associated with one term in the collection. Term frequency statistics are collected by parsing the document collection and counting how many times each term appears in each document. This is supported by the distributional hypothesis [101] from linguistics that theorizes that words that occur in the same context tend to have similar meanings. If two documents use a similar vocabulary and have similar term frequency statistics, then they are likely to be topically related. The result is a high dimensional, sparse document-by-term matrix with properties that can be explained by Zipf distributions [254] in term occurrence. The matrix represents a document collection where each row is a document and each column is a term in the vocabulary. In the clustering process, document vectors are often compared using the cosine similarity measure. The cosine similarity measure has two properties that make it useful for comparing documents. Document vectors are normalized to unit length when they are compared. This normalization is important since it accounts for the higher term frequencies that are expected in longer documents. The inner product that is used in computing the cosine similarity has non-zero contributions only from words that occur in both documents. Furthermore, sparse document representation allows for efficient computation.

Different approaches exist to weight the term frequency statistics contained in the document-by-term matrix. The goal of this weighting is to take into account the relative importance of different terms, and thereby facilitate improved performance in common tasks such as classification, clustering and ad hoc retrieval. Two popular approaches are TF-IDF [196] and BM25 [190, 235].

Clustering algorithms can be characterized by two properties. The first determines if cluster membership is discrete. Hard clustering algorithms only assign each document to one cluster. Soft clustering algorithms assign documents to one or more clusters in varying degree of membership. The second
determines the structure of the clusters found as being either flat or hierarchical. Flat clustering algorithms produce a fixed number of clusters with no relationships between the clusters. Hierarchical approaches produce a tree of clusters, starting with the broadest level clusters at the root and the narrowest at the leaves.

K-means \cite{157} is one of the most popular learning algorithms for use with document clustering and other clustering problems. It has been reported as one of the top 10 algorithms in data mining \cite{239}. Despite research into many other clustering algorithms, it is often the primary choice for practitioners due to its simplicity \cite{98} and quick convergence \cite{6}. Other hierarchical clustering approaches such as repeated bisecting k-means \cite{210}, K-tree \cite{56} and agglomerative hierarchical clustering \cite{210} have also been used. Note that the K-tree paper is by the author of thesis and other related content can be found in the associated Masters thesis \cite{52}. Further methods such as graph partitioning algorithms \cite{120}, matrix factorization \cite{241}, topic modeling \cite{25} and Gaussian mixture models \cite{64} have also been used.

The k-means algorithm \cite{157} uses the vector space model by iteratively optimizing \(k\) centroid vectors, which represent clusters. These clusters are updated by taking the mean of the nearest neighbors of the centroid. The algorithm proceeds to iteratively optimize the sum of squared distances between the centroids and the nearest neighbor set of vectors (clusters). This is achieved by iteratively updating the centroids to the cluster means and reassigning nearest neighbors to form new clusters, until convergence. The centroids are initialized by selecting \(k\) vectors from the document collection uniformly at random. It is well known that k-means is a special case of Expectation Maximization with hard cluster membership and isotropic Gaussian distributions \cite{182}.

The k-means algorithm has been shown to converge in a finite amount of time \cite{203} as each iteration of the algorithm visits a possible permutation without revisiting the same permutation twice, leading to the worst case analysis of exponential time. Arthur et al. \cite{6} have performed a smoothed analysis to explain the quick convergence of k-means theoretically. This is the same analysis that has been applied to the simplex algorithm, which has an \(n^2\) worst case complexity but usually converges in linear time on real data. While there are point sets that can force k-means to visit every permutation, they rarely appear in practical data. Furthermore, most practitioners limit the number of iterations k-means can run for, which results in linear time complexity for the algorithm. While the original proof of convergence applies to k-means using squared Euclidean distance \cite{203}, newer results show that other similarity measures from the Bregman divergence class of measures can be used with the same complexity guarantees \cite{14}. This includes similarity measures such as KL-divergence,
logistic loss, Mahalanobis distance and Itakura-Saito distance. Ding and He [72] demonstrate the relationship between k-means and Principle Component Analysis. PCA is usually thought of as a matrix factorization approach for dimensionality reduction whereas k-means is considered a clustering algorithm. It is shown that PCA provides a solution to the relaxed k-means problem by ignoring some constants, thus formally creating a link between k-means and matrix factorization methods.

3.1.1 Evaluation

Evaluating document clustering is a difficult task. Intrinsic or internal measures of quality such as distortion [179] or log likelihood [22] only indicate how well an algorithm optimized a particular representation. Intrinsic comparisons are inherently limited by the given representation and are not comparable between different representations. Extrinsic or external measures of quality compare a clustering to an external knowledge source such as a ground truth labeling of the collection or ad hoc relevance judgments. This allows comparison between different approaches. Extrinsic views of truth are created by humans and suffer from the tendency for humans to interpret document topics differently. Whether a document belongs to a particular topic or not can be subjective. To further complicate the problem, there are many valid ways to cluster a document collection. It has been noted that clustering is ultimately in the eye of the beholder [77].

Most of the current literature on clustering evaluation utilizes the classes-to-clusters evaluation which assumes that the classes of the documents are known. Each document has known category labels. Any clustering of these documents can be evaluated with respect to this predefined classification. It is important to note that the class labels are not used in the process of clustering, but only for the purpose of evaluation of the clustering results.

Evaluation Metrics

When comparing a cluster solution to a labeled ground truth, the standard measures of Purity, Entropy, NMI and F1 are often used to determine the quality of clusters with regard to the categories. Let \( \omega = \{w_1, w_2, \ldots, w_K\} \) be the set of clusters for the document collection \( D \) and \( \xi = \{c_1, c_2, \ldots, c_J\} \) be the set of categories. Each cluster and category are a subset of the document collection, \( \forall c \in \xi, w \in \omega : c, w \subset D \). Purity assigns a score based on the fraction of a cluster that is the majority category label,

\[
\argmax_{c \in \xi} \frac{|c \cap w_k|}{|w_k|},
\]

(3.1)
in the interval \([0, 1]\) where 0 is the absence of purity and 1 is total purity. Entropy defines a probability for each category and combines them to represent order within a clustering,

\[-\frac{1}{\log J} \sum_{j=1}^{J} \frac{|c_j \cap w_k|}{|w_k|} \log \frac{|c_j \cap w_k|}{|w_k|}, \tag{3.2}\]

which falls in the interval \([0, 1]\) where 0 is total order and 1 is complete disorder. F1 identifies a true positive \((tp)\) as two documents of the same category in the same cluster, a true negative \((tn)\) as two documents of different categories in different clusters and a false negative \((fn)\) as two documents of the same category in different clusters where the score combines these classification judgments using the harmonic mean,

\[\frac{2 \times tp}{2 \times tp + fn + fp}. \tag{3.3}\]

The Purity, Entropy and F1 scores assign a score to each cluster which can be micro or macro averaged across all the clusters. The micro average weights each cluster by its size, giving each document in the collection equal importance, in the final score. The macro average is simply the arithmetic mean, ignoring the size of the clusters. NMI makes a trade-off between the number of clusters and quality in an information theoretic sense. For a detailed explanation of these measures please consult [163].

**Single and Multi Label Evaluation**

Both the clustering approaches and the ground truth can be single or multi label. Examples of algorithms that produce multi label clusterings are soft or fuzzy approaches such as fuzzy c-means [21], Latent Dirichlet Allocation [25] or Expectation Maximisation [64]. A ground truth is multi label if it allows more than one category label for each document. Any combination of single or multi label clusterings or ground truths can be used for evaluation. However, it is only reasonable to compare approaches using the same combination of single or multi label clustering and ground truths. Multi label approaches are less restrictive than single label approaches as documents can exist in more than one category. There is redundancy in the data whether it is clustering or a ground truth. A ground truth can be viewed as a clustering and compared to another ground truth to measure how well the ground truths fit each other.
3.2 Document Representations

There are many different techniques that effect the generation of representations for clustering. By pre-processing documents, using new representations for different sources of information, and improving the representation of knowledge for learning, both the quality of clusters discovered and the run-time cost of producing clusters can be improved.

When representing natural language, different components of the language lead to different representations. Popular models include the bag-of-words approach [210], n-grams [167], part of speech tagging [3], and, semantic spaces [159].

Before any term based representation can be built, what constitutes a term must be determined via a number of factors such as term tokenization, conversion of words to their stems [181], removal of stop words [82], and, conversion of characters such as removing accents. Additionally, automated techniques may remove irrelevant boilerplate content from documents such as navigation structure on a web page are used [243].

There are many approaches to weight terms in the bag-of-words model. Whissel and Clarke [234] investigate different weighting strategies for document clustering. TF-IDF [196] BM25 [190, 235] are two popular approaches to weighting terms in the bag-of-words model.

Links between documents can be represented for use in document clustering. There are algorithms that work directly with the graph structure in an iterative fashion to produce clusters [223]. Other approaches embed the graph structure in a document representation such as vector space representations [187] [160]. The representations are then used with general clustering algorithms. The links can also be weighted by popular algorithms from ad hoc information retrieval such as PageRank [176] and HITS [126].

Other internal structural information about documents can be represented implicitly or explicitly. One effective implicit approach is to double the weight of title words in a bag-of-words representation [44]. Other approaches explicitly represent structure, such as those that represent the XML structure of documents. For example, Kutty et al. [134] represent both content and structure explicitly using a Tensor Space Model. Tagarelli et al. combine structure and content together using tree mining and modified similarity measures.

There are many dimensionality reduction techniques aimed at reducing the number of dimensions required to represent a document in a vector space model. They all aim to preserve the information available in documents. However, some particular approaches such as Latent Semantic Analysis [63] based upon the Singular Value Decomposition have been suggested to improve document repre-
sentations by finding “concepts” in text. Popular approaches include Principle Component Analysis [23], Singular Value Decomposition [92], Non-negative Matrix Factorization [241], Wavelet transform [170], Discrete Cosine Transform [83], Semantic Hashing [195], Locality Semantic Hashing [108], Random Indexing [194], and, Random Manhattan Indexing [184].

Random Indexing or Random Projections can be implemented using different strategies. Achlioptas [1] introduced an approach that is friendly for database environments that uses simpler uniform random distributions and ternary vectors. This approach is also used by Random Indexing [194] and TopSig [90], which is a contribution of this thesis. TopSig further compresses these representations by quantizing each dimension to a single bit. Kanerva [116] begins with binary vectors and combines them into a dense binary representation. Plate [180] uses holographic representations for nested compositional structure using real valued vectors.

With the rise of the social web, many web pages now have social annotation such as tags. These can be introduced into a document representation to further disambiguate the topic of a document [185].

Various other multimedia sources of information can be represented in documents such as audio [158], video [242] and images [153]. The processing of these signals constitutes many research fields to themselves. But there have been approaches incorporating this information into the document clustering process [18].

Documents can be mapped onto ontologies to provide a conceptual framework for the representation of meaning in documents. This has been used to improve the quality of document clusters [105].

Additionally, incorporating information from other category systems such as the Wikipedia has shown to improve the quality of clusters [106].

There are many approaches trying to use a single additional approach to improve document clustering via improved representation. However, there seems to be few approaches combining multiple sources and comparing how they contribute to the identification of clusters.

3.3 Ad Hoc Information Retrieval

Ad hoc information retrieval is concerned with the retrieval of documents given a user query. The phrase ad hoc is of Latin origin and literally means “for this”. Users create a query specific to their information need and retrieval systems return relevant documents. Much of modern IR research has been concerned with ranked retrieval of documents. Each document is assigned an estimate of relevance for a given query and the results are displayed in descending order of
relevance.

Retrieval systems or search engines provide automated ad hoc retrieval. The systems are typically implemented in software, but there have been hardware based retrieval systems [233]. The inverted file is the data structure that underpins most modern retrieval systems [163]. Like an index at the back of a textbook, an inverted file maps words to instances where they appear. The inverted file maps each term in the collection vocabulary to a postings list that identifies documents containing the term and the frequency with which it occurs. Various ranking functions re-weight query and document scores using the inverted file to improve ad hoc retrieval effectiveness. Popular approaches include TF-IDF [196], Language Models [247, 161], Divergence from Randomness [4] and Okapi BM25 [189]. However, signature file based approaches have been popular in the past [78] and have recently reappeared in the mainstream IR literature using random projections or random indexing [90], which is a contribution of this thesis. Signature files use a \( n \) bit binary string to represent documents and queries which are compared using the Hamming distance. The motivation of this approach is to exploit efficient bit-wise instructions on modern CPUs.

The study of retrieval models most commonly as different similarity functions focusses on the quality aspects of information retrieval systems. This studies how can the quality of results with respect to human judgement of a search engine be improved. In contrast, indexing implementations focus on the efficiency and approximate indexing that make a trade-off between quality of search results and efficiency.

Document signatures have been missing from mainstream publications about search engines and the Information Retrieval field in general for many years. Zotzel et al. clearly demonstrate the inferior performance of traditional document signatures in their paper, “Inverted Files Versus Signature Files for Text Indexing” [255]. They conclude that signature files or document signatures are slower, offer less functionality, and require more storage. Witten [237] come to the same conclusion. Traditional signature based indexes are not competitive with inverted files. They offer no distinct advantages over inverted indexes.

Traditional bit slice signature files [78] use efficient bit-wise operators. This is presented in an ad hoc manner and is motivated by efficient bit-wise processing without respect to Information Retrieval theory. Each document is allocated a signature of \( N \) bits. Each term is assigned a random signature where only \( n << N \) bits are set. These are combined using the bit-wise XOR operation. This is a Bloom filter [26] for the set of terms in a document. To avoid collisions resulting in errors, the difference between \( n \) and \( N \) must be very large. In contrast, the TopSig signatures [90] presented in this thesis compress the same representation that underlies the inverted index, the document-by-term matrix. It first begins
with full precision vectors and compresses them via random projections and numeric quantization. This is founded in the theory of preservation of the topological relationships of low dimensional embeddings of vectors presented in the Johnson-Lindenstrauss lemma \[114\].

Other recent approaches to similarity search \[249\] have focussed on mapping documents to \(N\) bit strings for comparison using the Hamming distance. However, they have only focussed on short strings for document-to-document search for tasks such as near duplicate detection or the “find other documents like these” queries. They do not investigate use of such structures for ad hoc information retrieval or clustering. Additionally they use machine learning approaches to find a better mapping for signatures, which may be computationally prohibitive for web scale document collections containing billions of documents.

### 3.3.1 Collection Distribution and Selection

A distributed information retrieval system consists of many different machines connected via a data communications network working together as one to provide search services for documents stored on each of the machines.

In a distributed retrieval setting, several different information resources can be stored in different retrieval systems and indexes. These resources are combined to provide a single query interface. This often results in an uncooperative distributed system. Uncooperative systems only allow use of the standard query interface provided by each system. All necessary information about a system that is acting as part of the distributed system has to be gained via these interfaces. The collection selection approach has to build an appropriate model by querying the systems individually.

Alternatively, a large collection such as the World Wide Web must be stored on more than one machine so it can be stored and processed. A large collection such as this is often processed in the setting of a cooperative distributed information retrieval system. Full access to the indexes of each system is possible. This enables use of global statistics and any other approaches that require more than the use of the standard query interface.

Collection distribution is the strategy that determines which documents are assigned to which machine. Common approaches include random \[128, 73\], source \[240\] and topic based allocation \[128, 240\]. A common criticism of topic based allocation is that automated methods for document clustering are not scalable for large collections.

Collection selection is the process that determines which machines to search in a distributed system given a query. There are many approaches to collection selection such as SHIRE \[130\], gGlOSS \[94\] and Cue-Validity Variance \[236\].
that use collection statistics to summarize the collection stored on a particular machine.

### 3.3.2 Evaluation

Information retrieval evaluation allows comparative evaluation of retrieval systems. System effectiveness is quantified by one of two approaches. User based evaluations measure user satisfaction with the system whereas system based evaluation determines how well a system can rank documents with respect to relevance judgments of users. The system based approach has become most popular due to its repeatability in a laboratory setting and the difficulty and cost of performing user based evaluations [229].

The Cranfield paradigm [42] is an experimental approach to the evaluation of retrieval systems. It has been adapted over the years and provides the theory that supports modern information retrieval evaluation efforts such as CLEF [81], NTCIR [115], INEX [8] and TREC [39]. It is a system based approach that evaluates how different systems rank relevant documents. For systems to be compared, the same set of information needs and documents have to be used. A test collection consists of documents, statements of information need, and relevance judgments [229]. Relevance judgments are often binary. A document is considered relevant if any of its contents can contribute to the satisfaction of the specified information need. Information needs are also referred to as topics and contain a textual description of the information need, including guidelines as to what may or may not be considered relevant. Typically, only the keyword based query of a topic is given to a retrieval system.

Many different measures exist for evaluating the distribution of relevant documents in ranked lists. The simplest measure is precision at \( n \) (P@\( n \)), defined as the fraction of the documents that are relevant in the first \( n \) documents in the ranked list. A strong justification for the use of P@\( n \) is that it reflects a real use case [256]. Most users only view the first few pages of results presented by a retrieval system. On the web, users may not even view the entire first page of results. Users reformulate their queries rather than continue searching the result list. Zobel et al. [256] argue that recall does not reflect a user’s need except in some specific circumstances such as medical and legal retrieval that require exhaustive searching. Furthermore, achieving total recall requires complete knowledge of the collection. The mean average precision measure or MAP takes into account both precision and recall. It is the area under the precision versus recall curve. Precision is the fraction of relevant documents at a given recall level, and a recall level is the fraction of returned relevant documents in the set of all relevant documents. As recall increases, precision
tends to drop since systems have a higher density of relevant results early in the ranked list. This naturally is the goal of a search engine.

The original Cranfield experiments required complete relevance judgments for the entire collection. This approach rules out large collections due to the necessity of exhaustive evaluation. On large scale collections with millions to billions of documents it is not possible to evaluate the relevance of every single document to every query. Therefore, an approach called pooling was developed \cite{209}. Each system in the experiment returns a ranked list of the first 1,000 to 2,000 documents in decreasing order of relevance. A proportion of the most relevant documents, often around 100 documents, are taken from each system and pooled together, and duplicates are removed since often there is a large overlap between the results lists of different systems. These pooled results are presented to the assessors in an unranked order. The assessors then determine relevance on this relatively small pool.

There has been much debate and experimentation to determine the effect of pooling. As test collections are designed to be reused, new approaches may lead to the return of relevant documents that were not represented in the pool that was assessed. These are un-judged relevant documents. Zobel \cite{255} found this is true for un-judged runs, but it does not unfairly affect a system’s performance on TREC collections. However, thought is required when designing an information retrieval evaluation experiment. The pool depth and diversity may need to change based on the type and size of the collection. Furthermore, tests such as those suggested by Zobel \cite{255} can be carried out to determine the reliability and reusability of a test collection.

Saracevic \cite{200} performed a meta-analysis of judge consistency. They concluded that consistency changes dramatically depending on the expertise of the assessors. Assessors with higher expertise agreed more often on the relevance of a document. Consistency also changed from experiment to experiment using the same assessors. Cranfield based evaluation of information retrieval has been scrutinized heavily over the years. However, assessor disagreement does not affect this type of evaluation because the rank order of each retrieval system does not often change when using different assessments from different judges. The stability only applies when averaging retrieval system performance over a set of queries. The retrieval quality can change drastically from query to query given a different assessor.

### 3.4 Cluster Hypothesis

The cluster hypothesis connects ad hoc information retrieval and document clustering. Manning et. al. \cite{163} states that “documents in the same cluster
behave similarly with respect to relevance to information needs”. If a document from a cluster is relevant to a query, then it is likely other documents in the same cluster are also relevant to the same query. This is due to the clustering algorithm grouping documents with similar terms together. There may also be some higher order correlation due to effects caused by the distributional hypothesis [101] and limitation of analysis to the size of a document; i.e. words are defined by the company they keep. It should be noted that the relevance of a document is assumed in an unsupervised manner in an operational retrieval system. While our evaluation of document clustering uses relevance judgments from humans, these are not always available or are noisy as they are derived from click through data.

The cluster hypothesis has been analyzed in three different forms. Originally the cluster hypothesis was tested at the collection level by clustering all documents in the collection. It was then extended to cluster only search results to improve display and ranking of documents. Finally, it has been extended to the sub-document level to deal with the multi-topic nature of longer documents.

3.4.1 Measuring the Cluster Hypothesis

There have been direct and indirect approaches to measuring the cluster hypothesis. Direct approaches measure similarity between documents to determine if relevant documents are more similar than non-relevant documents. Indirect measures use other IR related tasks such as ad hoc retrieval or document clustering to evaluate the hypothesis. If the cluster hypothesis is true for ad hoc search, then only a small fraction of the collection needs to be searched while the quality of search results will not be impacted or potentially improved. The same idea applies to the spread of relevant documents over a clustering of a collection. If the hypothesis is true, then the documents will only appear in a few clusters.

Voorhees [227] proposed the nearest neighbor (NN) test. It analyzes the five nearest neighbors of relevant documents according to a similarity measure. It determines the fraction of relevant documents located nearby other relevant documents. The NN test is comparable across different test collections and similarity measures making it useful for comparative evaluations. Voorhees [227] states that the test chose five nearest neighbors arbitrarily. If the number of neighbors is increased, then the test becomes less local in nature. The NN test overcame problems with the original cluster hypothesis test suggested by [111]. The original test plots the similarity between all pairs of relevant–relevant and relevant–non-relevant documents separately using a histogram approach. It does so for each query and then the results are averaged over all queries. A
The separation between the distributions indicates that the cluster hypothesis holds. The criticism of this approach is that there will always be more relevant–non-relevant pairs than relevant–relevant pairs. Voorhees [227] found that these two different measures give very different pictures of the cluster hypothesis. However, van Rijsbergen and Jones [225] found that the original measure was useful for explaining the different performance of retrieval systems on different collections.

Smucker and Allan [207] introduce a global measure of the cluster hypothesis based on the relevant document networks and measures from network analysis. The NN test is a local measure of clustering as it only inspects the five nearest neighbors of each relevant document. The measure proposed builds a directed, weighted, fully connected graph between all relevant documents where vertices are documents and edges represent the rank of the destination document when the source document is used as a query. This represents all relationships between relevant documents and not just those that fall within the five nearest as in the NN test. The normalized, mean reciprocal distance (nMRD) from small world network analysis [143] is used to measure the global efficiency based upon all shortest path distances between pairs of documents in the network. Each measure of the cluster hypothesis allows for a different view, allowing for multiple comparisons across multiple approaches to ranking.

In the following sections, we review the measurement of the cluster hypothesis via indirect measures.

### 3.4.2 Cluster Hypothesis in Ranked Retrieval

Many retrieval systems return a ranked list of documents in descending order of estimated relevance. Clustering provides an alternative approach to the organization of retrieved results that aims to reduce the cognitive load of analyzing retrieved documents. It achieves this by clustering documents into topical groups. The cluster hypothesis implies that most relevant documents will appear in a small number of clusters. A user can easily identify these clusters and find most of the relevant results. Alternatively, clusters are used to improve ranked results such as pseudo-relevance feedback and re-ranking using clusters.

Early studies in clustering ad hoc retrieval results by Croft [48] and Voorhees [228] performed static clustering of the entire collection. More recent studies found that clustering results for each query is more effective for improving the quality of results [102, 150, 215, 145]. However, clustering of an entire document collection allows for increased efficiency and its discussion has reappeared recently in the literature [171, 62, 128].

Evaluation of the cluster hypothesis changed significantly after the result
of Hearst and Pedersen [102] that confirmed that the cluster hypothesis can improve result ranking significantly. It did this by clustering results for each query rather than entire collections. Hearst and Pedersen [102] revise the cluster hypothesis. If two documents are similar to one query, they are not necessarily similar to another query. As documents have very high dimensionality, the definition of nearest neighbors changes depending on the query. Their system dynamically clusters retrieval results for each query using the interactive Scatter/Gather approach. The Scatter/Gather approach displays five clusters of the top n documents retrieved by a system. A user selects one or more clusters indicating that only those documents are to be viewed. This process can be applied recursively, performing the Scatter/Gather process again on the selected subset. The experiments using this system were performed on the TREC/Tipster collection using the TREC-4 queries. The top ranked clusters always contained at least 50 percent of the relevant documents retrieved. The results retrieved by the Scatter/Gather system were shown to be statistically significantly better than not using the system according to the t-test.

Crestani and Wu [47] investigate the cluster hypothesis in a distributed information retrieval environment. Even though there are many issues that introduce noise when presenting results from heterogeneous systems in a distributed environment, they demonstrate that clustering is still an effective approach for displaying search results to users.

Lee et al. [145] present a cluster-based resampling method for pseudo-relevance feedback. Previous attempts to use clusters for pseudo-relevance feedback have not resulted in statistically significant increases in retrieval efficiency. This is achieved by repeatedly selecting dominant documents, which is motivated by boosting from machine learning and statistics that repeatedly selects hard examples to move the decision boundary toward hard examples. It uses overlapping or soft clusters where documents can exist in more than one cluster. A dominant document is one that participates in many clusters. Repeatedly sampling dominant documents can emphasize the topics in a query and thus this approach clusters the top 100 documents in the ranked list. The most relevant terms are selected from the most relevant clusters and are used to expand the original query. It was most effective for large scale document collections where there are likely to be many topics present in the results for a query.

Re-ranking of the result list using clusters has found to be effective using both vector space approaches [147, 146], language models [156] and local score regularization [71]. In these approaches, the original ranked list is combined with the rank of clusters, rather than just the original query-to-document score.

Kulkarni and Callan [128] investigate the use of topical clusters for efficient and effective retrieval of documents. This work finds similar results to those
discovered at the INEX XML Mining track [171] when clustering large document collections, that relevant results for a given query cluster tightly and only fall in a few topical document clusters. Kulkarni and Callan solve the scalability problem of clustering by using sampling using k-means with the Kullback-Leibler divergence similarity measure. This work also implements and evaluates a collection selection approach called ReDDE and finds that topical clusters outperform random partitioning and that as little as one percent of a collection needs to be searched to maintain early precision.

3.4.3 Sub-document Cluster Hypothesis in Ranked Retrieval

Lamprier et al. [138] investigate the use of sub-document segments or passages to cluster documents in information retrieval systems. They extend the notion of the cluster hypothesis to the sub-document level where different segments of a document can have different topics that are relevant to different queries. If the cluster hypothesis holds at the sub-document level, then relevant passages will tend to cluster together. The approach proposed by Lamprier et al. [138] returns whole documents but uses passages to cluster multi-topic documents more effectively.

Lamprier et al. [138] suggest several different approaches to the segmentation of documents into passages. The authors chose to select an approach based on an evolutionary search algorithm that minimizes similarity between segments called SegGen [137]. They dismissed using arbitrary regions of overlapping text even though this was shown to be effective for passage retrieval by Kaszkiel and Zobel [121]. The reasons given are added complexity and that only disjoint subsets of the documents are appropriate in this case.

Lamprier et al. [138] performed experiments using sub-document clustering on ad hoc retrieval results with the aim of improving the quality. The authors did not consider sub-document clustering of the entire collection because earlier studies suggested this was not effective for whole document clustering. They found that sub-document clustering allowed better thematic clustering that placed more relevant results together than whole document clustering.
Research Questions

The research questions relating to the publications presented in this thesis are broken into the topics of representations, algorithms, evaluation and information retrieval.

4.1 Representations

The major research question for representations is, **How can compressed representations improve the scalability and efficiency of document clustering?**

How can compressed document representations be generated efficiently and effectively for use with document clustering algorithms?

Document vectors are often represented for learning using sparse vector approaches. Dense document representations are typically more computationally efficient when comparing entire documents. A dense representation that uses the same number of bytes is often faster because the entire vector is allocated contiguously and therefore, prefetching, branch prediction, and caches work more effectively on modern CPUs. Additionally, dimensionality reduction may require less bytes to represent the same information. However, dense representations such as those produced by dimensionality reduction techniques such as the Singular Value Decomposition, Principle Component Analysis and Non-Negative Matrix Factorization require many iterations of optimization. Furthermore, they require the entire document-by-term matrix to be analyzed.

In this thesis, I have investigated the use of random projections or random indexing with the combination of numeric quantization for document clustering. This resulted in the TopSig document representation being used for
document clustering. It represents documents as binary vectors that are compared using the Hamming distance. This allows the vectors to be processed 64 dimensions at a time in a single CPU instruction. Unlike iterative matrix factorization approaches to dimensionality reduction such as SVD, PCA and NMF, TopSig can encode documents as dense binary vectors without global analysis of the document-by-term matrix. Each document can be mapped onto a lower dimensional dense vector independently of all other documents in the document-by-term matrix. TopSig provides state-of-the-art compression when compared to inverted index approaches to compressing document vectors. Typically, document hashes or binary signatures are 64 or 128 bits in length and are used for near duplicate detection. TopSig produces much longer signatures from 1024 to 4096 bits that allow it to be effective for document clustering. However, this requires algorithms that work directly with the binary vectors produced by TopSig and this problem is addressed by later research questions.

**How can compressed document representations be generated for clustering in isolation without using global statistics?**

The use of random indexing allows document vectors to be produced using only the term frequencies within a single document in the TopSig model. There is some evidence to suggest that the quality of these vectors offer a trade-off between the cost of using global statistics and the quality of the representation. Any machine indexing documents can simply encode the document as a binary vector and append it to a file, or store it with the object it represents. The primary advantage of this approach is that a document signature can be generated completely in isolation and compared using the universal Hamming distance metric. Alternatively, statistics can be gathered from a smaller heterogeneous general knowledge document collection such as the Wikipedia. Terms that do not appear in the Wikipedia can be assumed to only appear in the given document for calculation of global statistics. This approach was undertaken to index the 50 million document ClueWeb09 Category B document collection [61].

**Can compressed document representations improve the efficiency of document clustering?**

Binary document vectors such as those produced by TopSig have been extensively used for nearest neighbor search. Many clustering algorithms perform a nearest neighbor search as part of their optimization process. However, how to implement algorithms efficiently using binary vectors requires the intermediate representations used in the algorithms to also be represented as binary vectors. These questions are answered in the algorithms section and have lead to a one to two orders of magnitude increase in clustering speed by using the
binary representations of TopSig \cite{58} without a reduction in document cluster quality. However, other algorithms have been developed that offer a further trade-off between efficiency and cluster quality. The combination of the TopSig representation and clustering algorithms is addressed in more detail in the algorithms section of the research questions.

**Why do TopSig signatures exhibit start-of-the-art performance at early recall but not at deep recall?**

The TopSig representation has been used for ad hoc information retrieval and has been found to be competitive with state-of-the-art approaches at early recall. However, at higher recall it suffers in comparison to probabilistic and language models. This question has been answered by analyzing the topology of the document vectors produced by TopSig \cite{58}. The pairwise distances between document vectors were compared. The TopSig representation does bias the distribution of random binary vectors as represented by the binomial distribution, so that there is non-uniformity and therefore clustering of the feature space. However, many documents become equidistant around the middle of the distribution. It is the tail of the distribution that allows differentiation at early recall. At higher recall most of the documents become equidistant and this is often referred to as the “curse of dimensionality”.

**How can the entity based XML structure of the INEX 2009 Wikipedia improve the quality of document clustering?**

The XML entity structure of the INEX 2009 Wikipedia was extracted using a bag of features approach. The terms used to cluster documents were restricted to those occurring inside the XML entity tags; i.e. the terms associated with entities. Furthermore, the entity tags were represented as a bag-of-tags where each tag represented the type of entity such as city or person. These document vectors were represented using random indexing to produce dense document vectors. They were then clustered using the random indexing K-tree \cite{59}. Each representation was tried individually and in combination with each other. The entity text approach proved most effective when using a category based evaluation. The combination of entity text and tags proved most effective when using ad hoc relevance judgements for evaluation.

### 4.2 Algorithms

The major research question for algorithms is, **How can binary document representations be exploited to increase the scalability and efficiency**
of clustering algorithms?

How can document signatures be clustered efficiently?

One of the most popular clustering algorithms is the k-means algorithm that finds clusters by iteratively updating prototypes that are called means, centroids, cluster representatives or cluster centers. Comparison of binary vectors is only efficient if the centroids are also binary vectors. Therefore, in the TopSig paper [90] I introduced an algorithm inspired by k-means. Every centroid is a binary vector the same length as the signatures being clustered. Centroids are updated by taking the median of all vectors. For binary valued vectors this is the most frequently occurring bit value, 0 or 1, in the dimension of nearest neighbors associated with the centroid. While there is no proof to show that this approach will converge as is the case for k-means when using squared Euclidean distance and updating of centroids as means, the algorithm has converged in practice. The convergence can be seen in the EM-tree paper [55]. When using 4096-bit vectors the TopSig k-means approach that works directly with binary vectors offers a 10 to 20 fold improvement in clustering speed. There is no statistically significant difference of the quality of the results.

What are the trade-offs that can be made between the efficiency and quality of clustering? Can efficiency be gained without sacrificing quality?

A trade-off between cluster quality and speed can be made by using shorter document signatures with 1024-bit signatures providing most of the quality of 4096-bit signatures. This is highlighted in the experiments in the TopSig paper [90].

From a different perspective, algorithms that attempt to approximate the k-means clustering algorithm such as the delayed update K-tree, the EM-tree algorithm and TSVQ [55] can be used to make a trade-off between cluster quality and the efficiency of finding clusters. The difference in quality of clusters found by k-means and the delayed update K-tree offer very little difference in terms of search quality [61], although there is a difference in RMSE [55]. The TopSig EM-tree algorithm when using 4096-bit vectors starts with an RMSE of approximately 1950 bits when producing 100 clusters. It converges to a solution approximately 280 bits lower of 1675 bits. The TSVQ algorithm converges to a slightly better solution 290 bits lower than the original RMSE. The k-means algorithm converges approximately another 10 bits lower. Considering there is a total of 300 bits of optimization completed, a difference of 10 to 50 bits is not a huge sacrifice considering the algorithms allow clustering of document collections that k-means can not perform without distributed and parallel com-
How can document clusters be found when the document representations are too large to fit in main memory and must be streamed from disk? How can clustering algorithms be effectively parallelized and distributed among many machines?

The EM-tree is suitable to streaming implementations where the data set is streamed from disk. The entire cluster tree is optimized at each iteration allowing an iteration of optimization to be performed with one pass over the data set. It is the batch nature of the optimization that makes this possible, and is also advantageous for distributed and parallel implementations. The entire cluster tree is frozen at each iteration. Therefore, the data set can be divided among the units of execution and then once all examples have been inserted into the tree, the results can be combined to update the tree. The details of this approach are discussed in the EM-tree paper \[55\].

How can Terabytes to Petabytes of text be clustered into fine grained clusters?

It is expected that a parallel, distributed and streaming EM-tree algorithm can cluster billions of documents in under a day, given the projections calculated based on the efficiency of a single threaded implementation \[55\]. Considering it is estimated that Google indexes approximately 45 billion pages, this approach is likely to be able to cluster the entire searchable Internet into millions of fine grained clusters! This problem has been attacked from 3 angles. Firstly, computationally efficient document representations were created for use with clustering, namely TopSig. Secondly, the k-means algorithm was adapted to work with TopSig, offering 1 to 2 orders of magnitude of performance increase over sparse vector approaches. Thirdly, the approach used in the TopSig k-means algorithm was combined with approaches based on the $m$-way nearest neighbor search tree to further improve the scalability of these algorithms. The EM-tree algorithm paper \[55\] provides results that indicate the algorithm is suitable for a parallel, distributed and streaming implementation. To prove beyond any doubt that the entire searchable Internet can be clustered, this needs to be implemented. This is an exciting avenue of research for the results presented in this thesis.

How can a data structure used for geometric clustering algorithms be formalized so that several algorithms, including flat and hierarchical algorithms can be constructed using the same data structure?

The EM-tree algorithm paper \[55\] introduces the $m$-way nearest neighbor search tree data structure and defines it formally. It is demonstrated how k-
means, TSVQ, EM-tree, agglomerative hierarchical clustering and the K-tree algorithms can be constructed using the data structure.

**How can the convergence of the EM-tree algorithm be proven?**

When the EM-tree algorithm was first devised it was not certain if the algorithm was guaranteed to converge. The EM-tree paper [55] proves that the k-means algorithm is being performed in the root node of the tree. This is then used in combination with structural induction to prove that all tree nodes will converge in finite time.

### 4.3 Evaluation

The major research question for evaluation is, **How can document clustering be evaluated in the context of Information Retrieval?**

**How can document clusters be evaluated in the context of a specific use-case instead of using the proxy of classification accuracy for evaluation?**

Most of the current literature evaluates different clustering approaches using a ground truth set of categories as if it is a classification problem. This thesis presents an alternative evaluation using ad hoc relevance judgements from information retrieval evaluation. This is motivated by the cluster hypothesis, stating that relevant documents tend to cluster together. Documents that are similar to each other are likely to be relevant to the same query. Therefore, it is expected that relevant results for a query will exist in relatively few document clusters. If the clustering is learning something useful with respect to the cluster hypothesis, then relevant documents will be expected in fewer clusters than generating clusters by random allocation completely ignoring document content. This approach to clustering evaluation has been tested throughout the thesis. It began at the INEX evaluation forum [171][62]. It was examined further with a publication on the Divergence from a Random Baseline approach to evaluation [60]. Finally, it was shown that the use-case indeed works with the implementation of a cluster-based search engine [61].

**Can the cluster hypothesis be verified through document clustering and ad hoc retrieval?**

This research question has been answered directly by all of the relevant material in the evaluation section of the thesis. All experimental results indicate that the cluster hypothesis holds and that only a fraction of a document collec-
Are categories appropriate for evaluating document clusters?

The goal of clustering is to place similar documents in the same cluster. Using categories for evaluation of clustering makes the assumption that two documents of the same category are similar. This may not strictly be the case for the broad definition of categories often used for evaluation of document clustering. For example, while help desk support and VLSI chip design both fall in the broad category of Information Technology, the disciplines have very little in common. Additionally, the number of categories typically used during evaluation does not reflect the hundreds of thousands of potential categories that exist for a heterogeneous general knowledge document collection such as the Wikipedia or the entire Internet. These arguments and more are presented throughout the publications in this thesis. Firstly, the evaluation approach is highlighted in Chapters 7, 9, and 11. Then its use-case is implemented as a search engine in Chapters 8 and 15. Additionally, apart from classification, the use of categories has no use-case. Indeed, if you want to perform document classification, it is more likely useful to train a classifier than use document clusters. The predictions will be more accurate.

How can large scale document collections be annotated with human generated information for evaluation?

One of the largest drawbacks to using category based evaluation is the human cost associated with labeling millions to billions of documents. One of the largest document collections available for evaluation using categories is the RCV1 \(^{152}\) collection. It consists of approximately 800,000 documents with hundreds of categories. It was reported that this process of labeling documents was terrible for the assessors involved. However, ad hoc retrieval evaluation offers a solution to this problem via the use of pooling. It reduces assessor load and has already been validated in information retrieval forums using large scale collections. This argument is most directly covered by the paper “Distributed Information Retrieval” \(^{61}\).

How can a clustering be evaluated to measure how effective it is for collection selection while reducing number of clusters that need to be searched while maintaining search result quality?

The Normalized Cumulative Cluster Gain (NCCG) measure was introduced at the evaluation at INEX in 2009 \(^{171}\) and 2010 \(^{62}\). It measures the spread of relevant documents over a clustering for the optimal collection selection. It uses an oracle to rank clusters according to the number of relevant documents they
contain. This is the best possible way to rank a clustering given a query. This evaluation is driven by the use-case of reducing search load in a distributed information retrieval system by only searching the first few most relevant clusters.

**How can a global optimum to the number of clusters be found when measuring distortion using RMSE?**

When varying the number of clusters found by a particular approach from 1 to the number of examples being clustered, \( n \), the RMSE continues to improve until there are \( n \) clusters. The Divergence from Random Baseline approach [60] augments the RMSE value such that it reaches a peak somewhere in between 1 and \( n \). It then proceeds to decrease when too many clusters are created. This provides a clear optimum for the number of clusters with respect to RMSE with respect to a particular clustering approach.

**How can the Wikipedia be used to obtain a reliable and useful ground truth for classification evaluation?**

The XML Mining track at INEX has used the Wikipedia for clustering and classification of documents for several years. Prior to 2010 the categories were extracted from the Wikipedia portals. The drawback to this approach is that only documents related to portals can be used. However, the Wikipedia contains a category graph but the problem is that it is noisy and contains many categories that are not useful for a particular document. Therefore, categories were extracted from the Wikipedia category graph that follow the shortest path between a particular page and one of the “Main Topic Classifications”. This is motivated by Occam’s Razor where the simplest explanation is often the best. As the subset of Wikipedia used for clustering and classification was defined by the ad hoc reference run in 2010, this allowed categories to be extracted for all pages. Furthermore, these categories were consistent in evaluation when reducing the quality of a clustering; the quality according to the categories was also reduced [90].

### 4.4 Information Retrieval

The major research question for information retrieval is, **How can fine grained clusters of large scale document collections be used so that only a fraction of the collection has to be searched?**

**How can clusters be ranked given a query?**

An approach to ranking the clusters is used once a document collection has
been clustered. Only the documents in the most relevant clusters are searched. There are many existing approaches to ranking clusters in the literature. A new approach named CBM625 [61] was introduced where cluster representatives are built using BM25 document weights. It squares the BM25 weights before combining them. This improves the ranking of relevant results in clusters on the Wikipedia. Note that the clusters were not found using BM25. They were found using the TopSig representation. The CBM625 approach builds a new representation for clusters where there is a cluster inverted index. It does this based on the groupings of the clusters; i.e. which documents belong to which cluster.

What is the benefit of document clustering in terms of reduced processing costs? How can a search engine be implemented to exploit document clusters to reduce search load?

The results in Chapters 8 and 15 of this thesis [61] evaluated the search load in the same manner as a prior publication that uses cluster-based retrieval [128]. It measures the number of documents searched in the minimum number of clusters that are required to produce P@10, P@20 and P@30 scores that are not statistically significantly different from exhaustive retrieval. The approach taken by Kulkarni and Callan [128] was published during the development of the approaches outlined in this thesis. However, the approach outlined in [61] is able to retrieve 13 times less documents than the approach of Kulkarni and Callan [128] on large scale document collections while not reducing the quality when compared to exhaustive search. Both of these separate lines of research implement working retrieval systems that use queries with relevance judgements for evaluation. Different approaches to clustering and retrieval are taken, however, the same results confirming the cluster hypothesis are found.

Do the results of the optimal collection selection based evaluation of document clusters transfer to application in a working search engine?

The optimal collection selection results from INEX suggested that a finer grained clustering would allow less of a document collection to be searched and still rank most of the relevant documents. This was validated by implementation of a cluster-based search engine that does exactly that [61].
Chapter 5

Representations

Different representations of documents can improve the quality of clusterings and run-time efficiency of processing documents when clustering. If a representation does not allow adequate separation of the topics in a document collection, then no learning algorithm is going to magically fix this. However, by exploiting different sources of information the documents may become more separable and therefore increase the quality of the clustering. Dimensionality reduction, such as the approach presented in TopSig, can improve the run-time efficiency of processing document vectors by reducing the total number of bytes required to represent documents for learning.

This chapter presents different approaches for improving the quality and efficiency when processing document vectors for document clustering.

5.1 TopSig: topology preserving document signatures

This paper introduces the TopSig document signature representation. TopSig produces binary strings, bit vectors or document signatures by applying random indexing or random projections and numeric quantization to document vectors. This creates a dense compressed representation of documents. The compressed binary representation of TopSig is one of the factors that allows the improvement of computational efficiency of clustering approaches outlined in this thesis. The binary representation allows efficient processing on 64-bit processors by processing 64 dimensions of each bit vector in a single CPU instruction. The compressed nature of the representation allows for less data to be processed than when using other vector representations for document clustering.
This also improves the performance and scalability of clustering approaches as less data has to be processed.

5.2 Pairwise Similarity of TopSig Document Signatures

This paper investigates the topology of TopSig document signatures by analyzing the distributions of the pairwise distances between signatures. It highlights that the indexing process of TopSig is influencing the random codes used to perform dimensionality reduction. It skews the binomial distribution for uniform random binary strings such that the feature space is no longer uniform and is clustered. It is this non-uniformity that allows the differentiation of meaning. Furthermore, it highlights that the curse of dimensionality still applies for document signatures with most signatures being equidistant to each other around the middle of the distribution. Only the signatures in the left tail of the distribution allow adequate separation of documents; i.e. only the local neighborhood of the signatures are interpretable. This also gives a reasonable explanation of why the TopSig model is only competitive at early precision for ad hoc information retrieval when compared to language and probabilistic models. There is only adequate separation between documents in the head of the ranked list. When proceeding down the ranked list, documents become equidistant at higher recall.

5.3 Clustering with Random Indexing K-tree and XML Structure

This paper investigates using the semantic XML structure of the INEX 2009 Wikipedia collection for improving the quality of clustering. The INEX 2009 Wikipedia contains entity markup as XML. The text inside the entity tags and the tags are represented as a bag of features. They are then clustered by Random Indexing K-tree. It was found that by combining the entity tags and text, the NCCG measure for collection selection, could be improved over using the entity text alone. However, the entity text was clearly better as discovering the categories used for evaluation as measured by Micro Purity.
Chapter 6

Algorithms

This chapter addresses efficient clustering algorithms for scaling to large document collections. The processing of ever increasing amounts of data allows better estimation of the probability distributions discovered during the unsupervised learning process.

6.1 TopSig: topology preserving document signatures

This paper introduces a modified version of the k-means algorithm that works directly with the binary string representation of TopSig. All cluster centers and documents are binary strings. This allows the Hamming distance measure to be used for comparison which can be computed 64 dimensions at a time using 64-bit processors. This allows a one to two order magnitude order increase in computational efficiency when compared to the sparse vector approaches used in the popular clustering toolkit, CLUTO. When using 4096 bit document signatures, the quality of the clustering is not statistically significantly different to the sparse vector approach used in CLUTO.

6.2 Clustering with Random Indexing K-tree and XML Structure

This paper investigates the use of Random Indexing K-tree for clustering the entire 2.7 million document INEX 2009 XML Wikipedia. Random Indexing uses random projections to create reduced dimensionality dense real valued vector representations from sparse high dimensional vectors. This paper demonstrates
that the Random Indexing K-tree can scale to the relatively large Wikipedia document collection while producing tens of hundreds of thousands of clusters. Note that most researchers who participated in the evaluation at INEX chose to only submit clusterings of the smaller 50,000 document subset. This may indicate that many other clustering approaches have issues when scaling to larger collections.

6.3 Distributed Information Retrieval: Collection Distribution, Selection and the Cluster Hypothesis for Evaluation of Document Clustering

The TopSig K-tree is introduced to scale clustering to 50 million documents while producing approximately 140,000 clusters in 10 hours in a single thread of execution. To the best of my knowledge, clustering of a document collection this large into this many clusters has not been reported in the literature. A delayed update mechanism is introduced as the K-tree algorithm regularly updates means along the insertion path of vectors inserted into the tree. When using binary vectors the cost of updating a mean is high as all the bits in vectors associated with the mean must be unpacked and repacked.

6.4 The EM-tree algorithm

A new algorithm called the EM-tree algorithm is introduced in this paper. It iteratively optimizes an entire $m$-way nearest neighbor search tree. It is proven to converge. Furthermore, it provides efficiency advantages over the TopSig K-tree when using TopSig bit vectors because means are updated less often.
Evaluation

This chapter investigates the evaluation of document clustering using ad hoc information retrieval relevance judgments. It proposes that the often narrow and well defined queries used for evaluation may be more meaningful than the broad and lofty categories often found in category based evaluation such as “Arts”. The use of pooling addresses the problem of creating external human created information for evaluation for collections containing millions to billions of documents. The evaluation of document clustering using relevance judgments is undertaken for a specific use, increasing the throughput of an information retrieval system by only searching a subset of a collection contained in the most relevant document clusters for a given query.

7.1 Overview of the INEX 2009 XML Mining track: Clustering and Classification of XML Documents

This paper introduces the collaborative evaluation at INEX. It uses ad hoc relevance judgments to evaluate document clustering for collection selection. The Normalized Cumulative Cluster Gain measure is introduced to evaluate a clustering with an “oracle” collection selection approach. This represents an upper bound on collection selection performance given a clustering solution.
7.2 Overview of the INEX 2010 XML Mining track: Clustering and Classification of XML Documents

This paper continues the evaluation of document clustering at INEX. It also introduces a new method for extracting categories from the Wikipedia category graph.

7.3 Document Clustering Evaluation: Divergence from a Random Baseline

This paper introduces an approach to the correction of ineffective clusterings called Divergence from a Random Baseline. It produces a baseline that looks identical to the clustering except that the documents are allocated to clusters uniformly randomly. The cluster size distribution of the baseline matches that of the clustering being evaluated. This is able to differentiate problematic clusterings using the NCCG evaluation measure. It also provides an optimum to distortion as measured by RMSE that is not apparent otherwise.
Chapter 8

Information Retrieval

8.1 Distributed Information Retrieval: Collection Distribution, Selection and the Cluster Hypothesis for Evaluation of Document Clustering

This paper combines all of the previous sections of representations, algorithms and evaluation to implement the ideas as a cluster-based search engine. The final result is that only 0.1% of the 50 million document ClueWeb 2009 Category B collection has to be searched. This is a 13 fold improvement over the previously best known result. It also addresses issues surrounding the evaluation of document clustering and demonstrates the use case presented in the evaluation at INEX.
Overview of the INEX 2009 XML Mining track: Clustering and Classification of XML Documents

I made a large proportion of the contributions in this paper relating to the clustering task. I wrote the evaluation software used for the clustering task,[1] defined and expanded the scope of the evaluation, performed the analysis of submissions, created plots, performed data preparation, aided in organization of XML Mining track, and, aided in writing of the paper.

9.1 Evaluation

This paper introduces the collaborative evaluation at INEX. It uses ad hoc relevance judgments to evaluate document clustering for collection selection. The Normalized Cumulative Cluster Gain measure is introduced to evaluate a clustering with an “oracle” collection selection approach. This represents an upper bound on collection selection performance given a clustering solution.

Overview of the INEX 2009 XML Mining Track: Clustering and Classification of XML Documents

Richi Nayak  Christopher M. De Vries  Sangeetha Kutty  Shlomo Geva

Faculty of Science and Technology
Queensland University of Technology
GPO Box 2434, Brisbane Qld 4001, Australia
{r.nayak, christopher.devries, s.kutty, s.geva}@qut.edu.au

Ludovic Denoyer  Patrick Gallinari

University Pierre et Marie Curie
LIP6 – 104 avenue du président Kennedy
75016 PARIS - FRANCE
{Ludovic.denoyer, Patrick.gallinari}@lip6.fr

Abstract. This report explains the objectives, datasets and evaluation criteria of both the clustering and classification tasks set in the INEX 2009 XML Mining track. The report also describes the approaches and results obtained by the different participants.

Keywords: XML document mining, INEX, Wikipedia, Structure and content, Clustering, Classification.

1 Introduction

The XML Document Mining track was launched for exploring two main ideas: (1) identifying key problems and new challenges of the emerging field of mining semi-structured documents, and (2) studying and assessing the potential of Machine Learning (ML) techniques for dealing with generic ML tasks in the structured domain i.e. classification and clustering of semi structured documents. This track has run for five editions during INEX 2005, 2006, 2007, 2008 and 2009. The four first editions have been summarized in [2, 3, 4] and we focus here on the 2009 edition.

INEX 2009 included two tasks in the XML Mining track: (1) unsupervised clustering task and (2) semi-supervised classification task where documents are organized in a graph. The clustering task requires the participants to group the documents into clusters without any knowledge of cluster labels using an unsupervised learning algorithm. On the other hand, the classification task requires the participants to label
the documents in the dataset into known classes using a supervised learning algorithm and a training set. This report gives the details of clustering and classifications tasks.

2 The Clustering Track

In the last decade, we have observed a proliferation of approaches for clustering XML documents based on their structure and content [9,12]. There have been many approaches developed for diverse application domains. Many applications require data objects to be grouped by similarity of content, tags, paths, structure and semantics. The clustering task in INEX 2009 evaluates clustering approaches in the context of XML information retrieval.

The INEX 2009 clustering task is different from the previous years due to its incorporation of a different evaluation strategy. The clustering task explicitly tests the Jardine and van Rijsbergen cluster hypothesis (1971) [8], which states that documents that cluster together have a similar relevance to a given query. It uses manual query assessments from the INEX 2009 Ad Hoc track. If the cluster hypothesis holds true, and if suitable clustering can be achieved, then a clustering solution will minimise the number of clusters that need to be searched to satisfy any given query. There are important practical reasons for performing collection selection on a very large corpus. If only a small fraction of clusters (hence documents) need to be searched, then the throughput of an information retrieval system will be greatly improved. INEX 2009 clustering task provides an evaluation forum to measure the performance of clustering methods for collection selection on a huge scale test collection. The collection consists of a set of documents, their labels, a set of information needs (queries), and the answers to those information needs.

2.1 Corpus

The INEX XML Wikipedia collection is used as a dataset in this task. This 60 Gigabyte collection contains 2.7 million English Wikipedia XML documents. The XML mark-up includes explicit tagging of named entities and document structure. In order to enable participation with minimal overheads in data-preparation the collection was pre-processed to provide various representations of the documents. For instance, a bag-of-words representation of terms and frequent phrases in a document, frequencies of various XML structures in the form of trees, links, named entities, etc. These various collection representations made this task a lightweight task that required the participants to submit clustering solutions without worrying about pre-processing this huge data collection.

There are a total of 1,970,515 terms after stemming, stopping, and eliminating terms that occur in a single document for this collection. There are 1,900,075 unique terms that appear more than once enclosed in entity tags. There are 5213 unique entity tags in the collection. There are a total of 110,766,016 unique links in the collection. There
are a total of 348,552 categories that contain all documents except for a 118,685 document subset containing no category information. These categories are derived by using the YAGO ontology [16]. The YAGO categories appear to follow a power law distribution as shown in Figure 1. Distribution of documents in the top-10 cluster category is shown in Table 1.

![Graph showing YAGO category distribution](image.png)

**Figure 1: The YAGO Category Distribution**

<table>
<thead>
<tr>
<th>Category</th>
<th>Documents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Living people</td>
<td>307304</td>
</tr>
<tr>
<td>All disambiguation pages</td>
<td>143463</td>
</tr>
<tr>
<td>Articles with invalid date parameter in template</td>
<td>77659</td>
</tr>
<tr>
<td>All orphaned articles</td>
<td>34612</td>
</tr>
<tr>
<td>All articles to be expanded</td>
<td>33810</td>
</tr>
<tr>
<td>Year of birth missing (living people)</td>
<td>32499</td>
</tr>
<tr>
<td>All articles lacking sources</td>
<td>21084</td>
</tr>
<tr>
<td>Human name disambiguation pages</td>
<td>18652</td>
</tr>
<tr>
<td>United States articles missing geocoordinate data</td>
<td>15363</td>
</tr>
<tr>
<td>IUCN Red List least concern species</td>
<td>15241</td>
</tr>
</tbody>
</table>

**Table 1: Top-10 Category Distribution**
A subset of collection containing about 50,000 documents (of the entire INEX 2009 corpus) was also used in the task to evaluate the categories labels results only, for teams that were unable to process such a large data collection.

2.2 Tasks and Evaluation Measures

The task was to utilize unsupervised classification techniques to group the documents into clusters. Participants were asked to submit multiple clustering solutions containing different numbers of clusters such as 100, 500, 1000, 2500, 5000 and 10000. The clustering solutions are evaluated by two means. Firstly, we utilise the classes-to-clusters evaluation which assumes that the classification of the documents in a sample is known (i.e., each document has a class label). Then any clustering of these documents can be evaluated with respect to this predefined classification. It is important to note that the class labels are not used in the process of clustering, but only for the purpose of evaluation of the clustering results.

The standard criterion of purity is used to determine the quality of clusters. These evaluation results were provided online and ongoing, starting from mid-October. Entropy and F-Score were not used in evaluation. The reason behind was that a document in the corpus maps to more than one category. Due to multi labels that a document can have, it was possible to obtain higher value of Entropy and F-Score than the ideal solution. Purity measures the extent to which each cluster contains documents primarily from one class. Each cluster is assigned with the class label of the majority of documents in it. The macro and micro purity of a clustering solution $cs$ is obtained as a weighted sum of the individual cluster purity. In general, larger the value of purity, better the clustering solution is.

$$
Purity\ (k) = \frac{\text{Number of documents with the majority label in cluster } k}{\text{Number of documents in cluster } k}
$$

$$
\text{Micro-Purity (cs)} = \frac{\sum_{k=0}^{n} \text{Purity}(k) \times \text{TotalFoundByClass}(k)}{\sum_{k=0}^{n} \text{TotalFoundByClass}(k)}
$$

$$
\text{Macro-Purity (cs)} = \frac{\sum_{k=0}^{n} \text{Purity}(k)}{\text{Total Number of Categories}}
$$

The clustering solutions are also evaluated to determine the quality of cluster relative to the optimal collection selection goal, given a set of queries. Better clustering solutions in this context will tend to (on average) group together relevant results for (previously unseen) ad-hoc queries. Real Ad-hoc retrieval queries and their manual assessment results are utilised in this evaluation. This novel approach evaluates the clustering solutions relative to a very specific objective - clustering a large document collection in an optimal manner in order to satisfy queries while minimising the search space. The Normalised Cumulative Gain is used to calculate the score of the...
best possible collection selection according to a given clustering solution of $n$ number of clusters. Better the score when the query result set contains more cohesive clusters. The cumulative gain of a cluster (CCG) is calculated by counting the number of relevant documents in a cluster, $c$, for a topic, $t$, where $c$ is the set of documents in a cluster and $t$ is the set of relevant documents for a topic.

$$CCG(c, t) = |c \cap t|$$

For a clustering solution for a given topic, a (sorted) vector CG is created representing each cluster by its CCG value. Clusters containing no relevant documents are represented by a value of zero. The cumulated gain for the vector CG is calculated which is then normalized on the ideal gain vector. Each clustering solution $cs$ is scored for how well it has split the relevant set into clusters using $CCG$ for the topic $t$.

$$SplitScore(t, cs) = \frac{\text{cumsum}(CG)}{nr^2}$$

$nr = \text{Number of relevant documents in the returned result set for the topic } t$.

A worst possible split is assumed to place each relevant document in a distinct cluster. Let $CG1$ be a vector that contains the cumulative gain of every cluster with a document each.

$$MinSplitScore(t, cs) = \sum_{i=1}^{n} \text{cumsum}(CG1)/nr^2$$

The normalized cluster cumulative gain (nCCG) for a given topic $t$ and a clustering solution $cs$ is given by,

$$nCCG(t, cs) = \frac{SplitScore(t, cs) - MinSplitScore(t, cs)}{1 - MinSplitScore(t, cs)}$$

The mean and the standard deviation of the nCCG score over all the topics for a clustering solution $cs$ are then calculated. $n$ is total number of topics.

$$\text{Mean nCCG}(cs) = \frac{\sum_{t=1}^{n} nCCG(t, cs)}{n}$$

$$\text{Std Dev nCCG}(cs) = \frac{\sum_{t=1}^{n} (nCCG(t, cs) - \text{Mean nCCG}(cs))^2}{n}$$

A total of 68 topics were used to evaluate the quality of clusters generated on the full set of collection of about 2.7 million documents. A total of 52 topics were used to evaluate the quality of clusters generated on the subset of collection of about 50,000
documents. A total number of 4858 documents were found relevant by the manual assessors for the 68 topics. An average number of 71 documents were found relevant for a given topic by manual assessors. The nCCG value varies from 0 to 1.

2.3 Participants, Submissions and Evaluation

A total of six research teams have participated in the INEX 2009 clustering task. Two of them submitted the results for the subset data only. We briefly summarised the approaches employed by the participants.

Exploiting Index Pruning Methods for Clustering XML Collections [1]
[1] used Cover-Coefficient Based Clustering Methodology (C3M) to cluster the XML documents. C3M is a single-pass partitioning type clustering algorithm which measures the probability of selecting a document given a term that has been selected from another document. As another approach, [1] adapted term-centric and document-centric index pruning techniques to obtain more compact representations of the documents. Documents are clustered with these reduced representations for various pruning levels, again using C3M algorithm. All of the experiments are executed on the subset of INEX 2009 corpus including 50K documents.

Clustering with Random Indexing K-tree and XML Structure [5]
The Random Indexing (RI) K-tree has been used to cluster the entire 2,666,190 XML documents in the INEX 2009 Wikipedia collection. Clusters were created as close as possible to the 100, 500, 1000, 5000 and 10000 clusters required for evaluation. The algorithm produces clusters of many sizes in a single pass. The desired clustering granularity is selected by choosing a particular level in the tree. In the context of document representation, topology preserving dimensionality reduction is preserving document meaning – or at least this is the conjecture which the team tests here. Document structure has been represented by using a bag of words and a bag of tags representation derived from the semantic markup in the INEX 2009 collection. The term frequencies were weighted with BM25 where K1 = 2 and b = 0.75. The tag frequencies were not weighted.

Exploiting Semantic tags in XML Clustering [10]
This technique combines the structure and content of XML documents for clustering. Each XML document in the INEX Wikipedia corpus is parsed and modeled as a rooted labeled ordered document tree. A constrained frequent subtree mining algorithm is then applied to extract the common structural features from these document trees in the corpus. Using the common structural features, the corresponding content features of the XML documents are extracted and represented in a Vector Space Model (VSM). The term frequencies in the VSM model were weighted with both TF-IDF and BM25. There were 100, 500 and 1000 clusters created for evaluation.
Performance of K-Star at the INEX'09 Clustering Task [13]
The employed approach was quite simple and focused on high scalability. The team used a modified version of the Star clustering method which automatically obtains the number of clusters. In each iteration, this clustering method brings together all those items whose similarity value is higher than a given threshold T, which is typically assumed to be the similarity average of the whole document collection and, therefore, the clustering method "discover" the number of clusters by its own. The run submitted to the INEX clustering task split the complete document collection into small subsets which are clustered with the above mentioned clustering method.

Evaluation

Figure 2, Figure 3 and Figure 4 show the performance of various teams in the clustering task. The legends are formatted in the following fashion, [metric] – [institution] (username) [method].
Figure 2: Purity and NCCG performance of different teams using the entire dataset
Figure 3: Purity performance of different teams using the subset data
Figure 4: NCCG performance of different teams using the subset data
3. The Classification Track

Dealing with XML document collections is a particularly challenging task for ML and IR. XML documents are defined by their logical structure and their content (hence the name semi-structured data). Moreover, in a large majority of cases (Web collections for example), XML documents collections are also structured by links between documents (hyperlinks for example). These links can be of different types and correspond to different information: for example, one collection can provide hierarchical links, hyperlinks, citations, ..... Most models developed in the field of XML categorization simultaneously use the content information and the internal structure of XML documents (see [2] and [3] for a list of models) but they rarely use the external structure of the collection i.e the links between documents. Some methods using both content and links have been proposed in [4].

The XML Classification Task focuses on the problem of learning to classify documents organized in a graph of documents. Unlike the 2008 track, we consider here the problem of Multiple labels classification where a document belongs to one or many different categories. This task considers a transductive context where, during the training phase, the whole graph of documents is known but the labels of only a part of them are given to the participants (Figure ).

![Training set and Final labelling](image)

**Figure 5:** The supervised classification task. Colors/Shapes correspond to categories, circle/white nodes are unlabeled nodes. Note that in this track, documents may belong to many categories
3.1 Corpus

The corpus provided is a subset of the INEX 2009 Corpus. We have extracted a set of 54,889 documents and the links between these documents. These links correspond to the links provided by the authors of the Wikipedia articles. The documents have been transformed into TF-IDF vectors by the organizers. The corpus thus corresponds to a set of 54,889 vectors of dimension 186,723. The documents belong to 39 categories that correspond to 39 Wikipedia portals. We have provided the labels of 20 % of the documents. The corpus is composed of 4,554,203 directed links that correspond to hyperlinks between the documents of the corpus. Each document is concerned by 84.1 links on average.

<table>
<thead>
<tr>
<th>Number of documents</th>
<th>54,889</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of training documents</td>
<td>11,028</td>
</tr>
<tr>
<td>Number of test documents</td>
<td>43,861</td>
</tr>
<tr>
<td>Number of categories</td>
<td>39</td>
</tr>
<tr>
<td>Number of links</td>
<td>4,554,203</td>
</tr>
<tr>
<td>Number of distinct words</td>
<td>186,723</td>
</tr>
</tbody>
</table>

3.2 Evaluation Measures

In order to evaluate the submissions of the participants, we have used different measures. The first set of measures are computed over each category and then averaged over the categories (using a micro or a macro average):

- **Accuracy** (ACC) corresponds to the classification error. Note that a system that returns zero relevant category for each document has a quite good accuracy.
- **F1 score** (F1) corresponds to the classical F1 measure and measures the ability of a system to find the relevant categories.

The second set of measures are computed over each document and then averaged over the documents:

- **Average precision** (APR) corresponds to the Average Precision computed over the list of categories returned for each document. It measures the ability of a
system to rank correctly the relevant categories. This measure is based on a ranking score of each category for each document.

3.3 Participants and Submissions

Five different teams have participated to the track. They have submitted different runs and we present here only the best results obtained by each team. Note that, due to additional experiments made after the submission deadline, the results presented here and the results presented in the participants’ articles can be different.

<table>
<thead>
<tr>
<th>Team</th>
<th>Micro ACC</th>
<th>Macro ACC</th>
<th>Micro F1</th>
<th>Macro F1</th>
<th>APR</th>
</tr>
</thead>
<tbody>
<tr>
<td>University of Wollongong</td>
<td>92.5</td>
<td>94.6</td>
<td>51.2</td>
<td>47.9</td>
<td>68</td>
</tr>
<tr>
<td>University of Peking</td>
<td>94.7</td>
<td>96.2</td>
<td>51.8</td>
<td>48</td>
<td>70.2</td>
</tr>
<tr>
<td>XEROX Research Center</td>
<td>96.3</td>
<td>97.4</td>
<td>60</td>
<td>57.1</td>
<td>67.8</td>
</tr>
<tr>
<td>University of Saint Etienne</td>
<td>96.2</td>
<td>97.4</td>
<td>56.4</td>
<td>53</td>
<td>68.5</td>
</tr>
<tr>
<td>University of Granada</td>
<td>67.8</td>
<td>75.4</td>
<td>26.2</td>
<td>25.3</td>
<td>72.9</td>
</tr>
</tbody>
</table>

3.4 Summary of the methods

We give here a brief description of the methods submitted by the participants. Please refer to the participants articles for a detailed description of the methods and for the final results obtained by the different teams.

Multi-label Wikipedia classification with textual and graph features [6]

This paper proposes to evaluate different classification methods used on both the textual features of the pages to classify, and also on graph features computed from the structure of the graph. These features include for example the mean centrality, the degree centrality, etc...Different classifiers have been tested to handle the multi-label problem.
Supervised Encoding of Graph-of-Graphs for Classification and Regression Problems [7]

This article proposes a novel method which aims at encoding graph of graph structures where data correspond to a graph of elements which are also composed of graphs. The graph to graph structure is described and then used as a classification model based on a back-propagation of the error through the different level of the nested structure.

UJM at INEX 2009 XML Mining Track [11]

The authors use different classification strategies based on a set of content features to handle the classification problem. They mainly compare different features selection methods and thresholding strategies.

Link-based text classification using Bayesian networks [14]

The article presents a Bayesian network model that is able to handle both content and links between documents. The proposed model is an extension of the Naïve Bayes model to documents organized in a graph.

Extended VSM for XML Document Classification using Frequent Subtrees [15]

The last paper proposes the structured link vector model which aims at modeling both the content and the structure of the documents in a vector. Mainly, the authors propose to insert into classical content-based features vectors information about the frequent XML subtrees and the links between documents.

4. Conclusion

The XML Mining track in INEX 2009 brought together researchers from Information Retrieval, Data Mining, Machine Learning and XML fields. The clustering task allowed participants to evaluate clustering methods against a real use case and with significant volumes of data. The task was designed to facilitate participation with minimal effort by providing not only raw data, but also pre-processed data which can be easily used by existing clustering software. The classification task allowed participant to explore algorithmic, theoretical and practical issues regarding the classification of interdependent XML documents.
5. Acknowledgments

We would like to thank all the participants for their efforts and hard work.

6. References

12. Nayak – Book chapter
Chapter 10

Clustering with Random Indexing K-tree and XML Structure

For this paper I completed almost all tasks related to contributions. Lance contributed some parts of the software related to Random Indexing. I wrote the manuscript, defined the experimental setup, conducted the experiments, performed data analysis, and, wrote software.

10.1 Representations

This paper investigates using the semantic XML structure of the INEX 2009 Wikipedia collection for improving the quality of clustering. The INEX 2009 Wikipedia contains entity markup as XML. The text inside the entity tags and the tags are represented as a bag of features. They are then clustered by Random Indexing K-tree. It was found that by combining the entity tags and text, the NCCG measure for collection selection, could be improved over using the entity text alone. However, the entity text was clearly better as discovering the categories used for evaluation as measured by Micro Purity.

10.2 Algorithms

This paper investigates the use of Random Indexing K-tree for clustering the entire 2.7 million document INEX 2009 XML Wikipedia. Random Indexing uses random projections to create reduced dimensionality dense real valued vector representations from sparse high dimensional vectors. This paper demonstrates

http://ktree.sf.net
that the Random Indexing K-tree can scale to the relatively large Wikipedia
document collection while producing tens of hundreds of thousands of clusters.
Note that most researchers who participated in the evaluation at INEX chose
to only submit clusterings of the smaller 50,000 document subset. This may
indicate that many other clustering approaches have issues when scaling to
larger collections.
Abstract. This paper describes the approach taken to the clustering task at INEX 2009 by a group at the Queensland University of Technology. The Random Indexing (RI) K-tree has been used with a representation that is based on the semantic markup available in the INEX 2009 Wikipedia collection. The RI K-tree is a scalable approach to clustering large document collections. This approach has produced quality clustering when evaluated using two different methodologies.

Key words: INEX, XML, Mining, Documents, Clustering, Structure, K-tree, Random Indexing, Random Projection

1 Introduction

The cluster hypothesis suggests that documents that cluster together tend to have relevance to similar queries. The clustering task at INEX 2009 aims to evaluate the utility of clustering in collection selection. The goal of clustering is to minimise the spread of relevant results of ad-hoc queries over a clustering solution. The purpose of clustering in this context is to determine the distribution of a collection over multiple machines. We have a dual optimisation problem - it is desirable to maximise the number of clusters while minimising the spread of relevant results of ad-hoc queries over the clusters. Search efficiency can be increased with the distribution of clusters (sub-collections) on more machines. However, since it is not possible to produce clusters that split the collection to perfectly satisfy all conceivable ad-hoc queries, a good clustering solution is expected to optimise the distribution such that for most ad-hoc queries most of the results can be found in a small set of clusters. The goal of collection selection is then to rank the clusters (sub-collections) to identify the order in which they should be searched to satisfy any given query.

We have used K-tree [1, 2] to generate clustering solutions. The scalability of K-tree in a document clustering setting has been discussed by De Vries and Geva [3, 4]. The original contribution to K-tree in INEX 2009 is the use of Random Indexing (RI) to represent the documents. The K-tree algorithm has also been modified to work with the RI representation. RI facilitates an efficient and economical vector space representation. The RI K-tree provides a scalable approach
to clustering large collections at multiple granularities. The latest Wikipedia collection has included semantic markup that is based on the YAGO ontology. This markup had been used in encoding the documents, and two simple approaches are described in Section 4.

This paper introduces and defines Random Indexing in Section 2 and explains its use with K-tree in Section 3. The representation of semantic markup is discussed in Section 4. The combination of RI K-tree and representation of semantic markup introduced in earlier sections is applied to the INEX clustering task in Sections 5, 6 and 7. The paper ends with a conclusion in Section 8.

2 Random Indexing

RI [5] is an efficient, scalable and incremental approach to the implementation of a word space model. Word space models use the distribution of terms in documents to create high dimensional document vectors. The directions of these document vectors represent various semantic meanings and contexts.

Latent Semantic Analysis (LSA) [6] is a popular word space model. LSA creates context vectors from a document term occurrence matrix by performing Singular Value Decomposition (SVD). Dimensionality reduction is achieved through projection of the document term occurrence vectors onto the subspace spanned by the vectors with the largest singular values in the decomposition. This projection is optimal in the sense that it minimises the sum of squares of the difference between the original matrix and the projected matrix components. In contrast, Random Indexing first creates random context vectors of lower dimensionality and then combines them to create a term occurrence matrix in the dimensionally reduced space. Each term in the collection is assigned a random vector and the document term occurrence vector is then a superposition of all the term random vectors. There is no matrix decomposition and hence the process is efficient.

RI is also known as Random Projection and is explained by the Johnson and Linden-Strauss lemma [7]. It states that if points in a high dimensional space are projected into a lower dimensional, randomly selected subspace of sufficient dimensions they will approximately retain the same topology. Any n point set in Euclidean space can be embedded in $O(\log n/\epsilon^2)$ dimensions without distorting the pair-wise distances between points by more than $1 \pm \epsilon$, where $0 < \epsilon < 1$. Dasgupta and Gupta [8] have provided a proof for the Johnson and Linden-Strauss lemma, showing that the proposed bounds of the lemma hold.

The RI mapping is performed by producing r dimensional index vectors for each term in a collection, where r is the desired dimensionality of the reduced space. We have chosen these vectors to be sparse and ternary. Ternary index vectors were introduced by Achlioptas [9] as being friendly for database environments. Bingham and Mannila [10] have found that the sparsity of index vectors does not effect the distortion of the embedding via experimental analysis. Sparse index vectors reduce the time to complete RI as only 10 percent of the dimensions are non-zero. However, other choices exist for index vectors such as binary
spatter codes \[11\] which are randomly selected binary vectors and holographic
reduced representations \[12\] that are dense randomly selected real valued
vectors. When indexing the INEX 2009 collection, the index vectors are multiplied
by the BM25 weight for each term in each document and added to the RI docu-
ment vector. The document vector becomes a superposition of the index vectors
multiplied by the term weights as determined by BM25.

RI can be viewed as a matrix multiplication of a document by term matrix \(D\)
and a random projection matrix \(I\) resulting in a reduced matrix \(R\). Row vectors
of \(I\) contain index vectors of \(r\) dimensions for each term in \(D\). Moreover, \(n\) is
the number of documents, \(t\) is the number of terms and \(r\) is the dimensionality
of the reduced spaced. \(R\) is the reduced matrix where each row vector represents a
document. Equation 1 defines RI as a matrix multiplication.

\[
D_{n \times t} I_{t \times r} = R_{n \times r}
\]

(1)

Note that the RI document vectors themselves are not random. They are
composed of a superposition of random term vectors and the superposition result
depends on BM25 term weights and document content.

Another way to view RI is to interpret each index vector as a code. These
codes are nearly orthogonal to all other codes produced, resulting in minimal
interference between terms in the reduced vector space. Orthogonality can be
measured by creating a pair-wise distance matrix between index vectors using
cosine similarity as a distance measure. If two vectors are orthogonal their cosine
similarity will be zero. The closer the vectors are to orthogonal the closer their
cosine similarity will be to zero. Therefore, it is expected that the pair-wise
distance matrix will contain values close to zero in every position except the
main diagonal. Finding truly orthogonal codes is computationally expensive and
therefore avoided. Nearly orthogonal codes are found by drawing values in the
vector from a normal distribution. Figure 1 shows the addition of index vectors
(nearly orthogonal codes) to create a document representation.

<table>
<thead>
<tr>
<th>Full Representation</th>
<th>Compressed Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRAVEL</td>
<td>(1, -5, -7, -9)</td>
</tr>
<tr>
<td>MARS</td>
<td>(-3, 5, 8, 10)</td>
</tr>
<tr>
<td>SPACE</td>
<td>(-2, -6, -9, 11)</td>
</tr>
<tr>
<td>TELESCOPE</td>
<td>(-4, -5, 10, 12)</td>
</tr>
<tr>
<td>Document</td>
<td>(1, -1, -1, 1, 1, -1, -1, -2, 2, 1, 1)</td>
</tr>
</tbody>
</table>

Fig. 1. Random Indexing Example
3 Random Indexing K-tree

The K-tree is an online and dynamic clustering algorithm that scales well by making many local decisions resulting in a hierarchical tree structure. It is a hybrid of the B+-tree and k-means algorithms where the B+-tree has been adapted for multi-dimensional data and the k-means algorithm is used to perform splits in the tree. It is built in a bottom-up manner as data arrives. De Vries and Geva [2–4] discuss the algorithm and its application to document clustering, including the scalability of the algorithm. K-tree was compared to the popular CLUTO clustering toolkit and found to cluster significantly faster when a large number of clusters are required [4]. The Random Indexing (RI) K-tree [13] combines K-tree with RI to improve the quality of results and run-time performance.

The time complexity of K-tree depends on the length of the document vectors. K-tree insertion incurs two costs, finding the appropriate leaf node for insertion and k-means invocation during node splits. It is therefore desirable to operate with a lower dimensional vector representation.

The combination of RI with K-tree is a good fit. Both algorithms operate in an on-line and incremental mode. This allows it to track the distribution of data as it arrives and changes over time. K-tree insertions and deletions allow flexibility when tracking data in volatile and changing collections. Furthermore, K-tree performs best with dense vectors, such as those produced by RI.

Given the scalable and dynamical properties of the RI K-tree algorithm we propose it is a good solution for clustering large volatile document collections. The logarithmic lookup time of K-tree [13] to find the most similar cluster is also of use in a functioning information retrieval system relying on collection selection. This allows a query broker to direct queries to the most relevant sub-collection in sub-linear time with respect to the size of the collection.

4 Document Representation

Document structure has been represented by using a bag of words and a bag of tags representation derived from the semantic markup in the INEX 2009 collection. Both are vector space representations.

The bag of words is made up of term frequencies contained within any entity tags in the collection. The term frequencies were weighted with BM25 [14] where \( K1 = 2 \) and \( b = 0.75 \). We hypothesise that terms contained within entity tags are more likely to indicate the topic of a document. Therefore, documents with the same topic will fall closer together in the vector space representation. This indirectly exploits the XML structure.

The bag of tags representation is made up in an analogous manner of XML entity tag frequencies. The tag frequencies were not weighted. Entity tags consist of concepts such as scientist, location and person. We conjecture that documents with similar tags will belong to the same topic. Future work may compare tag frequency based vectors to set based vectors. In set based vectors each tag would be recorded as existing in a document or not. This way it can be determined if...
the use of tag frequencies is worthwhile. If a power law distribution exists in tag frequencies the Inverse Document Frequency heuristic may also prove useful as it did with link graphs [3]. The entity tags directly exploit structure by indexing it.

The bag of words and tags representations were combined. This is done by adding the two vector space representations together and then normalising the resulting vector to unit length. As both of the representations are based on RI, the codes between representations will be nearly orthogonal. However, a larger number of dimensions may be required to accommodate the extra information.

5 Run-time Performance

The performance of RI K-tree has been measured when operating in main memory. The concern is with the performance of the clustering algorithm. Efficiency was not taken into account when indexing or loading the final representation into memory.

All performance figures are for processing all 2,666,190 XML Wikipedia documents. The RI operations took a total of 1860 seconds for the entity text representation. The randomly selected lower dimensional space had 1000 dimensions. The run time of the K-tree algorithm varies between 1200 and 1500 seconds depending on the tree order selected between 15 and 50. This includes the process of re-inserting all vectors to their nearest neighbour leaves upon completion of the tree building process. This produces clustering at many different granularities at once. Table 1 lists the different sized clusters found by trees of order 20, 40, 60, 80 and 100, where \( m \) is the tree order.

<table>
<thead>
<tr>
<th>Level</th>
<th>20</th>
<th>40</th>
<th>60</th>
<th>80</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>12</td>
<td>3</td>
<td>8</td>
<td>3</td>
<td>92</td>
</tr>
<tr>
<td>2</td>
<td>111</td>
<td>89</td>
<td>356</td>
<td>129</td>
<td>2011</td>
</tr>
<tr>
<td>3</td>
<td>542</td>
<td>1260</td>
<td>5610</td>
<td>3090</td>
<td>53174</td>
</tr>
<tr>
<td>4</td>
<td>2161</td>
<td>12865</td>
<td>89612</td>
<td>67794</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>8529</td>
<td>154934</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>37230</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>197299</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1. K-tree Clusters

6 Experimental Setup

The Random Indexing (RI) K-tree has been used to cluster all 2,666,190 XML documents in the INEX 2009 Wikipedia collection. Bag of words and tags representations were used to create different clusters. Both representations were also
combined to create a third set of clusters. Clusters were created as close as possible to the 100, 500, 1000, 5000 and 10000 clusters required for evaluation. The RI K-tree produces clusters in an unsupervised manner where the exact number of clusters can not be precisely controlled. It is determined by the tree order and the randomised seeding process. The algorithm produces clusters of many sizes in a single pass. The desired clustering granularity is selected by choosing a particular level in the tree.

Random Indexing (RI) is an efficient dimensionality reduction technique that projects points in a high dimensional space onto a randomly selected lower dimensional space. It is able to preserve the topology of the points. In the context of document representation, topology preserving dimensionality reduction is preserving document meaning, or at least this is the conjecture which we test here. The RI projection produces dense document vectors that work well with the K-tree algorithm.

Cluster quality has been measured with two metrics this year. Purity is a commonly used metric and it is measured against an external ground truth. In the case of the INEX 2009 collection, the categories were created by YAWN. Purity is the fraction of documents with the majority category in a cluster. Micro purity is the average across all clusters in a solution where each cluster’s contribution is weighted by the fraction of documents it contains from the whole collection. Thus, smaller clusters have less influence and larger clusters have more influence on the average. Normalised Cumulative Cluster Gain (NCCG) is a new measure based on relevance judgments from search queries in the ad-hoc track. The ad-hoc track at INEX provides most relevant documents for each topic based on manual human evaluations. Given the relevant results an oracle cluster ranking system can be built, where clusters are sorted in descending order by the number of relevant documents they contain. NCCG measures the spread of relevant documents over the clusters. A score of one is achieved if all relevant documents appear in the first cluster and a score of zero is achieved if relevant documents are evenly spread across all clusters. NCCG rewards placing all relevant documents together. Therefore, it is testing the clustering hypothesis that states that relevant documents for a query tend to cluster together.

7 Experimental Results

Table 2 lists micro purity and NCCG scores for all submissions that clustered the full INEX 2009 collection. The table is split into sections corresponding to the required cluster sizes specified for the track. The RI K-tree, using the entity text representation is clearly the best approach when it comes to finding high purity clusters using an approach that can scale to the full collection at all cluster sizes. The NCCG metric for collection selection favours the combination of entity text and tags over either representation. It changes the ordering of results when compared to the traditional ground truth based approach. The C3m based approach produced higher quality clusters with respect to the NCCG metric on two occasions at 100 and 10,000 clusters.
Guyon et. al. [15] argue that the context of clustering needs to be taken into account during evaluation. The evaluation of this INEX task tests the clustering hypothesis in the information retrieval specific. Clustering is intended to facilitate document distribution and collection selection for ad-hoc retrieval, and it is tested in that setting. This differs greatly from evaluation where authors assign categories to documents and the categories are then used as the ground truth for the evaluation of clustering. Guyon et. al. [15] argue, and we agree, that ground truth based evaluations are unsound. This is particularly true when it comes to an information retrieval setting where the number of potential topics (clusters) is virtually unconstrained. It is a virtually impossible task to compare alternative clustering possibilities by inspecting large numbers of documents in clusters. In contrast, the evaluation of topics represented as queries in an ad-hoc retrieval system achieves high levels of inter-judge agreement. These relevance judgments have been the backbone of ad-hoc information retrieval system evaluations for many years. They have also been exposed to criticism and review by many of the top researchers in the field. By exploiting this high quality, human generated information, we can have great confidence that we are testing clustering in the context of its use. The context is specifically clustering of documents in an information retrieval setting.

Guyon et. al. go as far to say “In our opinion, this approach [ground truth approach] is dangerous. The underlying assumption is that points with the same class labels form clusters. This might be the case for some data sets but not for others.”. If the ground truth reflected the application of clustering in an information retrieval context, then the scores would agree between purity and NCCG. However, they do not. Therefore, we argue that the NCCG scores based on ad-hoc queries are more meaningful in an information retrieval setting.

Relevance of documents to queries can also be derived from click-through data in an operational search engine. This provides a potential mountain of relevance judgments.

8 Conclusion

In conclusion the RI K-tree provided a scalable approach to clustering at multiple granularities in a single pass with quality comparable to other approaches. The hypothesis that combining entity text and tag based representations will improve quality held true for the new ad-hoc based evaluation. Furthermore, the evaluation provided insights into why it is important to take context of use into account when evaluating clustering.
<table>
<thead>
<tr>
<th>Method</th>
<th>Clusters</th>
<th>Micro</th>
<th>Purity</th>
<th>NCCG</th>
</tr>
</thead>
<tbody>
<tr>
<td>RI K-tree Text</td>
<td>88</td>
<td>0.1744</td>
<td>0.7859</td>
<td></td>
</tr>
<tr>
<td>RI K-tree Tags</td>
<td>99</td>
<td>0.1427</td>
<td>0.7851</td>
<td></td>
</tr>
<tr>
<td>RI K-tree Text and Tags</td>
<td>105</td>
<td>0.1450</td>
<td>0.8003</td>
<td></td>
</tr>
<tr>
<td>C3m Content Only (bildb)</td>
<td>101</td>
<td>0.1566</td>
<td>0.8205</td>
<td></td>
</tr>
<tr>
<td>RI K-tree Text</td>
<td>420</td>
<td>0.1918</td>
<td>0.6770</td>
<td></td>
</tr>
<tr>
<td>RI K-tree Tags</td>
<td>477</td>
<td>0.1526</td>
<td>0.7546</td>
<td></td>
</tr>
<tr>
<td>RI K-tree Text and Tags</td>
<td>509</td>
<td>0.1608</td>
<td>0.7330</td>
<td></td>
</tr>
<tr>
<td>RI K-tree Text</td>
<td>1009</td>
<td>0.2140</td>
<td>0.6450</td>
<td></td>
</tr>
<tr>
<td>RI K-tree Tags</td>
<td>1026</td>
<td>0.1609</td>
<td>0.7021</td>
<td></td>
</tr>
<tr>
<td>RI K-tree Text and Tags</td>
<td>963</td>
<td>0.1690</td>
<td>0.7092</td>
<td></td>
</tr>
<tr>
<td>C3m Content Only (bildb)</td>
<td>1001</td>
<td>0.1617</td>
<td>0.6614</td>
<td></td>
</tr>
<tr>
<td>RI K-tree Text</td>
<td>2450</td>
<td>0.2136</td>
<td>0.6100</td>
<td></td>
</tr>
<tr>
<td>RI K-tree Tags</td>
<td>2407</td>
<td>0.1769</td>
<td>0.6348</td>
<td></td>
</tr>
<tr>
<td>RI K-tree Text and Tags</td>
<td>2536</td>
<td>0.1928</td>
<td>0.6575</td>
<td></td>
</tr>
<tr>
<td>BM25 BicMsGrowingKMeans (mark)</td>
<td>2263</td>
<td>0.1698</td>
<td>0.6349</td>
<td></td>
</tr>
<tr>
<td>RI K-tree Text</td>
<td>4914</td>
<td>0.2384</td>
<td>0.5581</td>
<td></td>
</tr>
<tr>
<td>RI K-tree Tags</td>
<td>4993</td>
<td>0.2020</td>
<td>0.5729</td>
<td></td>
</tr>
<tr>
<td>RI K-tree Text and Tags</td>
<td>4978</td>
<td>0.2038</td>
<td>0.6003</td>
<td></td>
</tr>
<tr>
<td>RI K-tree Text</td>
<td>9725</td>
<td>0.2719</td>
<td>0.4736</td>
<td></td>
</tr>
<tr>
<td>RI K-tree Tags</td>
<td>10453</td>
<td>0.2321</td>
<td>0.5274</td>
<td></td>
</tr>
<tr>
<td>RI K-tree Text and Tags</td>
<td>9896</td>
<td>0.2509</td>
<td>0.5492</td>
<td></td>
</tr>
<tr>
<td>C3m Content Only (bildb)</td>
<td>10001</td>
<td>0.1942</td>
<td>0.6035</td>
<td></td>
</tr>
<tr>
<td>BM25 BicMsGrowingKMeans (mark)</td>
<td>12636</td>
<td>0.2416</td>
<td>0.5885</td>
<td></td>
</tr>
</tbody>
</table>

Table 2. Clusters
References

Overview of the INEX 2010 XML Mining track: Clustering and Classification of XML Documents

For this paper I completed almost all tasks related to the contributions. Unlike the previous INEX overview paper, I was involved in organizing both the clustering and the classification tasks. I wrote the manuscript, created the plots, performed data preparation and analysis, aided in organization of XML Mining track, wrote the evaluation software\(^1\) proposed and implemented the technique to extract categories from the Wikipedia graph, and, proposed and implemented the divergence from a random baseline approach to identify pathological clusterings.

11.1 Evaluation

This paper continues the evaluation of document clustering at INEX. It also introduces a new method for extracting categories from the Wikipedia category graph.

\(^1\)!http://mloss.org/software/view/468/
Overview of the INEX 2010 XML Mining Track: Clustering and Classification of XML Documents

Christopher M. De Vries\textsuperscript{1} Richi Nayak\textsuperscript{1} Sangeetha Kutty\textsuperscript{1} Shlomo Geva\textsuperscript{1} Andrea Tagarelli\textsuperscript{2} 

Faculty of Science and Technology, Queensland University of Technology, Brisbane, Australia\textsuperscript{1} 
University of Calabria, Italy\textsuperscript{2} 

chris@de-vries.id.au \{r.nayak, s.kutty, s.geva\}@qut.edu.au 
tagarelli@deis.unical.it 

Abstract. This report explains the objectives, datasets and evaluation criteria of both the clustering and classification tasks set in the INEX 2010 XML Mining track. The report also describes the approaches and results obtained by participants.

Key words: XML document mining, INEX, Wikipedia, Structure, Content, Clustering, Classification

1 Introduction

The XML Document Mining track was launched for exploring two main ideas: (1) identifying key problems and new challenges of the emerging field of mining semi-structured documents, and (2) studying and assessing the potential of Machine Learning (ML) techniques for dealing with generic ML tasks in the structured domain, i.e., classification and clustering of semi-structured documents. This track has run for six editions during INEX 2005, 2006, 2007, 2008, 2009 and 2010. The first five editions have been summarized in [1,2,3,4] and we focus here on the 2010 edition.

INEX 2010 included two tasks in the XML Mining track: (1) unsupervised clustering task and (2) semi-supervised classification task where documents are organized in a graph. The clustering task requires the participants to group the documents into clusters without any knowledge of category labels using an unsupervised learning algorithm. On the other hand, the classification task requires the participants to label the documents in the dataset into known categories using a supervised learning algorithm and a training set. This report gives the details of clustering and classification tasks.

2 Corpus

Working with XML documents is a particularly challenging task for ML and IR. XML documents are defined by their logical structure and content. The
The current Wikipedia collection contains structure as (1) document structure such as sections, titles and tables, (2) semantic structure as entities mined by YAWN, and (3) navigation structure as document to document links. In 2008 and 2009 the classification task focused on exploiting the link structure of the Wikipedia and continues to do so this year. The clustering task has continued in the same manner as previous years and uses any available content or structure.

A 146,225 document subset of the INEX XML Wikipedia collection was used as a data set for the clustering and classification tasks. The subset is determined by the reference run used for the ad hoc track. The reference run contains the 1500 highest ranked documents for each of the queries in the ad hoc track. The queries were searched using an implementation of Okapi BM25 in the ANT search engine. Using the reference run reduced the collection from 2,666,190 to 146,225 documents. This is a new approach for selecting the XML Mining subset. In previous years it was selected by choosing documents from Wikipedia portals.

The clustering evaluation uses ad hoc relevance judgements for evaluation and most of the relevant documents are contained in the subset. Table 1 contains details of documents relevant to queries missing from the subset. The reference run contains approximately 90 percent of the relevant documents.

<table>
<thead>
<tr>
<th>Topic</th>
<th>Relevant</th>
<th>Missing</th>
<th>Topic</th>
<th>Relevant</th>
<th>Missing</th>
</tr>
</thead>
<tbody>
<tr>
<td>2010003</td>
<td>231</td>
<td>24 (10.39%)</td>
<td>2010035</td>
<td>16</td>
<td>3 (18.75%)</td>
</tr>
<tr>
<td>2010004</td>
<td>124</td>
<td>29 (23.39%)</td>
<td>2010036</td>
<td>94</td>
<td>0 (0.00%)</td>
</tr>
<tr>
<td>2010006</td>
<td>151</td>
<td>20 (13.26%)</td>
<td>2010037</td>
<td>11</td>
<td>0 (0.00%)</td>
</tr>
<tr>
<td>2010007</td>
<td>49</td>
<td>6 (12.24%)</td>
<td>2010038</td>
<td>433</td>
<td>8 (1.85%)</td>
</tr>
<tr>
<td>2010010</td>
<td>251</td>
<td>6 (2.39%)</td>
<td>2010039</td>
<td>138</td>
<td>0 (0.00%)</td>
</tr>
<tr>
<td>2010014</td>
<td>64</td>
<td>7 (10.94%)</td>
<td>2010040</td>
<td>60</td>
<td>3 (5.00%)</td>
</tr>
<tr>
<td>2010016</td>
<td>506</td>
<td>72 (14.23%)</td>
<td>2010041</td>
<td>35</td>
<td>0 (0.00%)</td>
</tr>
<tr>
<td>2010017</td>
<td>5</td>
<td>0 (0.00%)</td>
<td>2010043</td>
<td>130</td>
<td>11 (8.46%)</td>
</tr>
<tr>
<td>2010018</td>
<td>34</td>
<td>0 (0.00%)</td>
<td>2010045</td>
<td>159</td>
<td>60 (37.74%)</td>
</tr>
<tr>
<td>2010019</td>
<td>6</td>
<td>0 (0.00%)</td>
<td>2010046</td>
<td>53</td>
<td>0 (0.00%)</td>
</tr>
<tr>
<td>2010020</td>
<td>34</td>
<td>0 (0.00%)</td>
<td>2010047</td>
<td>18</td>
<td>0 (0.00%)</td>
</tr>
<tr>
<td>2010021</td>
<td>203</td>
<td>28 (13.79%)</td>
<td>2010048</td>
<td>72</td>
<td>11 (15.28%)</td>
</tr>
<tr>
<td>2010023</td>
<td>115</td>
<td>31 (26.96%)</td>
<td>2010049</td>
<td>42</td>
<td>6 (14.29%)</td>
</tr>
<tr>
<td>2010025</td>
<td>19</td>
<td>0 (0.00%)</td>
<td>2010050</td>
<td>147</td>
<td>5 (3.40%)</td>
</tr>
<tr>
<td>2010026</td>
<td>54</td>
<td>5 (9.26%)</td>
<td>2010054</td>
<td>292</td>
<td>42 (14.38%)</td>
</tr>
<tr>
<td>2010027</td>
<td>77</td>
<td>4 (5.19%)</td>
<td>2010056</td>
<td>269</td>
<td>37 (13.75%)</td>
</tr>
<tr>
<td>2010030</td>
<td>80</td>
<td>32 (40.00%)</td>
<td>2010057</td>
<td>74</td>
<td>0 (0.00%)</td>
</tr>
<tr>
<td>2010033</td>
<td>18</td>
<td>1 (5.56%)</td>
<td>2010061</td>
<td>13</td>
<td>0 (0.00%)</td>
</tr>
<tr>
<td>2010032</td>
<td>23</td>
<td>2 (8.70%)</td>
<td>2010068</td>
<td>222</td>
<td>2 (0.90%)</td>
</tr>
<tr>
<td>2010033</td>
<td>134</td>
<td>16 (11.94%)</td>
<td>2010069</td>
<td>358</td>
<td>82 (22.91%)</td>
</tr>
<tr>
<td>2010034</td>
<td>115</td>
<td>9 (7.83%)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Total 5451 587 (10.77%)

Table 1. Relevant Documents Missing from the XML Mining Subset
3 Categories

In previous years, document categories have been selected using Wikipedia portals where each portal becomes a category. The drawback of this approach is that it only finds categories for documents related to portals. Last year the categories used for clustering evaluation were produced by YAWN that creates categories based on entities found from the YAGO ontology. These categories are very fine grained and narrow and were found not to be particularly useful.

A new approach for extracting categories was taken this year. The Wikipedia categories listed for each document are very similar to the YAGO categories as YAGO contains entities based on Wikipedia information. Both the Wikipedia and YAGO categories are noisy and very fine grained. However, the Wikipedia categories exist in a category graph where there are 24 high level topical categories called the “main topic classifications” \(^1\). Unfortunately, the category graph is not a hierarchy and contains cycles. Many of the paths from a document to the main topic classifications do not make sense. Additionally, users who add categories to Wikipedia pages often attach them to fine grained categories in the graph. They may not realize what links the internal structure of the graph contains when choosing particular categories. The category graph can be changed over time also changing the original intent of the author. Therefore, categories were extracted by finding the shortest paths through the graph between a document and any of the main topic classifications. This is motivated by Occam’s Razor where the simplest explanation is often the correct one. Figure 1 illustrates the Wikipedia category graphs and highlights a hypothetical shortest category path for the document Hydrogen.

For INEX 2010 the category graph from the 22nd of June 2010 Wikipedia dump was used. The graph consists of Wikipedia pages with the “Category:” prefix such as “Category:Science”. The graph is extracted by finding links between category pages. Generally speaking, a category page links to another category page that is broader in scope. Wikipedia pages indicate their categories by linking to a category page.

Figure 2 lists the algorithm used to extract the categories. The INEX 2010 categories were extracted where only the 2 broadest levels of categories were extracted \((t = 2)\). Only categories containing more than 3000 documents were used. This approach extracts multiple categories for a document resulting in a multi-label set of documents for INEX 2010. Note that paths that contain the “Category:Hidden” category were not used. Table 2 lists the categories that were extracted.

In Figure 2, \(P\) is the set of Wikipedia pages (articles) to find categories for. \(C\) is the set of Categories in the Wikipedia. \(M\) the set of categories in the main topic classifications. \(G = (V, E)\) is the Wikipedia category graph consisting of a set of vertices \(V\) and edges \(E\) where the vertices consist of pages \(P\) and categories \(C\). Where \(P \subset V, C \subset V, M \subset V\) and \(M \subset C\). Moreover, \(t\) is a parameter indicating the broadest \(t\) levels to consider as categories; if \(t\) is 1 then only the main topic

\(^1\) http://en.wikipedia.org/wiki/Category:Main_topic_classifications
classifications are considered; if $t$ is 2 then the main topic classifications and any categories 1 edge away in the graph are considered and so on.

Note that a path is a sequence of graph vertices visited from page $p \in P$ to main topic $m \in M$. For example, Hydrogen $\rightarrow$ Category:Elements $\rightarrow$ Category:Chemistry $\rightarrow$ Category:Science, is the hypothetical path for the Wikipedia document Hydrogen.

**ExtractCategories**($G, M, P, t$)
1. $E$ = a map from page $p \in P$ to a list of categories for $p$
2. for $p \in P$
3. $S$ = the set of shortest paths between $p$ and any category in $M$
4. for $s \in S$
5. if path $s$ does not contain Category:Hidden
6. $B$ = the set of last $t$ vertices in path $s$
7. for $b \in B$
8. append $b$ to list $E[p]$
9. return $E$

**Fig. 2.** Algorithm to Extract Categories from the Wikipedia

The category extraction process could be enhanced in the future using frequent pattern mining to find interesting repeated sequences in the shortest paths. Other graph algorithms such as the Minimum Spanning Tree algorithm could be
Table 2. XML Mining Categories

<table>
<thead>
<tr>
<th>Category</th>
<th>Documents</th>
<th>Category</th>
<th>Documents</th>
</tr>
</thead>
<tbody>
<tr>
<td>People</td>
<td>48186</td>
<td>Agriculture</td>
<td>5975</td>
</tr>
<tr>
<td>Society</td>
<td>34912</td>
<td>Education</td>
<td>4367</td>
</tr>
<tr>
<td>Culture</td>
<td>27986</td>
<td>Companies</td>
<td>4314</td>
</tr>
<tr>
<td>Geography</td>
<td>22747</td>
<td>Biology</td>
<td>4309</td>
</tr>
<tr>
<td>Politics</td>
<td>18519</td>
<td>Recreation</td>
<td>4276</td>
</tr>
<tr>
<td>Humanities</td>
<td>14738</td>
<td>Environment</td>
<td>4216</td>
</tr>
<tr>
<td>Countries</td>
<td>13966</td>
<td>Musical culture</td>
<td>4195</td>
</tr>
<tr>
<td>Arts</td>
<td>11979</td>
<td>Geography stubs</td>
<td>4052</td>
</tr>
<tr>
<td>History</td>
<td>10821</td>
<td>Information</td>
<td>3919</td>
</tr>
<tr>
<td>Business</td>
<td>10249</td>
<td>American musicians</td>
<td>3845</td>
</tr>
<tr>
<td>Applied sciences</td>
<td>9278</td>
<td>Language</td>
<td>3764</td>
</tr>
<tr>
<td>Life</td>
<td>9018</td>
<td>Literature</td>
<td>3660</td>
</tr>
<tr>
<td>Technology</td>
<td>8920</td>
<td>Belief</td>
<td>3412</td>
</tr>
<tr>
<td>Entertainment</td>
<td>8887</td>
<td>Creative works</td>
<td>3395</td>
</tr>
<tr>
<td>Nature</td>
<td>7400</td>
<td>Human geography</td>
<td>3370</td>
</tr>
<tr>
<td>Science</td>
<td>7311</td>
<td>Places</td>
<td>3202</td>
</tr>
<tr>
<td>Computing</td>
<td>6835</td>
<td>Law</td>
<td>3156</td>
</tr>
<tr>
<td>Health</td>
<td>6329</td>
<td>Cultural history</td>
<td>3117</td>
</tr>
</tbody>
</table>

used to simplify the graph. The browsable category tree starting at the “main topic classifications” appears to have processed the category graph as well. Using this post-processed graph could also improve the categories.

4 Clustering Task

The task was to utilize unsupervised machine learning techniques to group the documents into clusters. Participants were asked to submit multiple clustering solutions containing 50, 100, 200, 500 and 1000 clusters. The categories extracted contained 36 categories due to only using categories with greater than 3000 documents. This choice was arbitrary and the decision for cluster sizes was made based on the number of documents in the collection before the categories were extracted. As there are not really 36 “true” categories, a direct comparison of 36 clusters with 36 categories is not necessary. The number of categories in a document collection is extremely subjective. Measuring how the categories behave over multiple cluster sizes indicates the quality of clusters and the trend can be visualized.

4.1 Clustering Evaluation Measures

The clustering solutions are evaluated by two means. Firstly, we utilize the categories-to-clusters evaluation which assumes that the categorization of the
documents in a sample is known (i.e., each document has known category labels). Any clustering of these documents can be evaluated with respect to this predefined categorization. It is important to note that the category labels are not used in the process of clustering, but only for the purpose of evaluation of the clustering results.

The standard measures of Purity, Entropy, NMI and F1 are used to determine the quality of clusters with regard to the categories. Negentropy [5] is also used. It measures the same system property as Entropy but it is normalized and inverted so scores fall between 0 and 1 where 0 is the worst and 1 is the best. The evaluation measures the mapping of categories-to-clusters where the categories are multi-label but the clusters are not. A document can have multiple categories but documents can only belong to one cluster. Each measure is defined to deal with a multi-label ground truth.

**Purity.** The standard criterion of purity is used to determine the quality of clusters by measuring the extent to which each cluster contains documents primarily from one category. The simplicity and the popularity of this measure means that it has been used as the only evaluation measure for the clustering task in the INEX 2006 and INEX 2009. In general, the larger the value of purity, the better the clustering solution.

Let $\omega = \{w_1, w_2, \ldots, w_K\}$, denote the set of clusters for the dataset $D$ and $\xi = \{c_1, c_2, \ldots, c_J\}$ represent the set of categories. The purity of a cluster $w_k$ is defined as:

$$P(w_k) = \frac{\max_{j} |w_k \cap c_j|}{|w_k|}$$

where $w_k$ is the set of documents in cluster $w_k$ and $c_j$ is the set of documents that occurs in category $c_j$. The numerator indicates the number of documents in cluster $k$ that occurs most in category $j$ and the denominator is the number of documents in the cluster $w_k$.

The purity of the clustering solution $\omega$ can be calculated based on micro-purity and macro-purity. Micro-purity of the clustering solution $\omega$ is obtained as a weighted sum of individual cluster purity. Macro-purity is the unweighted arithmetic mean based on the total number of categories [5].

$$Micro-Purity(\omega, \xi) = \frac{\sum_{k=0}^{K} P(w_k) \times |w_k|}{\sum_{k=0}^{K} |w_k \cap c_j|}$$

$$Macro-Purity(\omega, \xi) = \frac{\sum_{k=0}^{K} P(w_k)}{J}$$

**Entropy.** It is used to measure the distribution of the documents on various categories. Given a particular cluster $\omega_k$ of size $n_k$, the entropy of this cluster is defined to be:

$$E(\omega_k) = -\frac{1}{\log J} \sum_{j=1}^{J} \frac{n_j^k}{n_k} \log \frac{n_j^k}{n_k}$$
where $J$ is the number of categories in the dataset, and $n_{jk}$ is the number of documents of the $j$th category that were assigned to the $k$th cluster \[6\]. The clustering solution can then be measured by the sum of the individual cluster entropies weighted according to the clustering size as defined below:

$$\text{Entropy} = \sum_{k=1}^{K} \frac{n_k}{K} E(\omega_k)$$ \hspace{1cm} (5)$$

It is scaled from 0 to 1. A perfect clustering solution will have an entropy value of 0.

**F1-measure** Another standard measure that is used to evaluate the clustering solution is the F1-measure. It helps to calculate not only the number of documents that are correctly classified together in a cluster but also the number of documents that are misclassified from the cluster.

In order to calculate the F1-measure, three types of decisions are used. Among them there are two types of correct decisions: True Positives (TP) and True Negatives (TN). A TP decision assigns two similar documents to the same cluster; a TN decision assigns two dissimilar documents to different clusters. On the other hand, a False Positive (FP) is an error decision that assigns two dissimilar documents to the same cluster \[7\]. Though there is another error decision, FN, that assigns two similar documents to different clusters, it is not used in calculating F1-measure.

Using the TP, TN and FP decisions, the precision and the recall for the micro-F1 are defined as:

$$\text{precision}_{\text{micro-F1}} = \frac{\sum_{j=1}^{J} TP_j}{\sum_{j=1}^{J} TP_j + FP_j}$$ \hspace{1cm} (6)$$

$$\text{recall}_{\text{micro-F1}} = \frac{\sum_{j=1}^{J} TP_j}{\sum_{j=1}^{J} TP_j + TN_j}$$ \hspace{1cm} (7)$$

The precision and the recall for the macro-F1 are defined as

$$\text{precision}_{\text{macro-F1}} = \frac{\sum_{j=1}^{J} TP_j}{J}$$ \hspace{1cm} (8)$$

$$\text{recall}_{\text{macro-F1}} = \frac{\sum_{j=1}^{J} TP_j}{J}$$ \hspace{1cm} (9)$$

where $TP_j$ is the number of documents in category $c_j$ that exists in cluster $w_k$, $TP_j$ is the number of documents that is not in category $c_j$ but that exists in cluster $w_k$ and $TN_j$ is the number of documents that is in category $c_j$ but does not exist in cluster $w_k$.

F1 can now be defined as:

$$F1 = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$ \hspace{1cm} (10)$$

87
Micro-F1 = \frac{2 \times \text{precision}_{\text{micro-F1}} \times \text{recall}_{\text{micro-F1}}}{\text{precision}_{\text{micro-F1}} + \text{recall}_{\text{micro-F1}}} \quad (11)

Macro-F1 = \frac{2 \times \text{precision}_{\text{macro-F1}} \times \text{recall}_{\text{macro-F1}}}{\text{precision}_{\text{macro-F1}} + \text{recall}_{\text{macro-F1}}} \quad (12)

**Normalized Mutual Information (NMI).** Another evaluation measure is the Normalized Mutual Information (NMI) which helps to identify the trade-off between the quality of the clusters against the number of clusters [7].

NMI [7] is defined as,

\[ NMI(\omega; \xi) = \frac{I(\omega; \xi)}{H(\omega) + H(C)} \quad (13) \]

\[ I(\omega; \xi) = \sum_k \sum_j P(w_k \cap c_j) \log \frac{P(w_k \cap c_j)}{P(w_k)P(c_j)} \quad (14) \]

where \(P(w_k), P(c_j)\) and \(P(w_k \cap c_j)\) indicate the probabilities of a document in cluster \(w_k\), category \(c_j\) and in both \(w_k\) and \(c_j\).

\(H(\omega)\) is the measure of uncertainty given by,

\[ H(\omega) = -\sum_k (P(w_k) \log P(w_k)) - \sum_k \frac{|w_k|}{N} \log \frac{|w_k|}{N} \quad (15) \]

**Collection selection evaluation using NCCG measure**

This evaluation measure was used in evaluating the INEX 2009 dataset [4] and is based on Van Rijsbergen’s clustering hypothesis. Van Rijsbergen and his co-workers [8] conducted intensive study on the use of the clustering hypothesis test on information retrieval, which states that documents which are similar to each other may be expected to be relevant to the same requests; dissimilar documents, conversely, are unlikely to be relevant to the same requests. If the hypothesis holds true, then relevant documents will appear in a small number of clusters and the document clustering solution can be evaluated by measuring the spread of relevant documents for the given set of queries.

To test this hypothesis on a real-life dataset, the INEX 2009 dataset, the clustering task was evaluated by determining the quality of clusters relative to the optimal collection selection [4]. Collection selection involves splitting a collection into subsets and recommending which subset needs to be searched for a given query. This allows a search engine to search fewer documents, resulting in improved runtime performance over searching the entire collection.

The evaluation of collection selection was conducted using the manual query assessments for a given set of queries from the INEX 2009 Ad Hoc track [4]. The manual query assessment is called the relevance judgment in Information Retrieval (IR) and has been used to evaluate ad hoc retrieval of documents. It involves defining a query based on the information need, a search engine returning results for the query and humans judging whether the results returned by the search engine are relevant to the information need.
Better clustering solutions in this context will tend to (on average) group together relevant results for (previously unseen) ad hoc queries. Real ad hoc retrieval queries and their manual assessment results are utilised in this evaluation. This approach evaluates the clustering solutions relative to a very specific objective – clustering a large document collection in an optimal manner in order to satisfy queries while minimising the search space. The metric used for evaluating the collection selection is called the Normalized Cumulative Cluster gain (NCCG) [4].

The NCCG is used to calculate the score of the best possible collection selection according to a given clustering solution of \( n \) number of clusters. The score is better when the query result set contains more cohesive clusters. The Cumulative Gain of a Cluster (CCG) is calculated by counting the number of documents of the cluster that appear in the relevant set returned for a topic by manual assessors.

\[
CCG(c, t) = \sum_{i=1}^{n} (Rel_i)
\]  

(16)

For a clustering solution for a given topic, a (sorted) vector CG is created representing each cluster by its CCG value. Clusters containing no relevant documents are represented by a value of zero. The cumulated gain for the vector CG is calculated, which is then normalized on the ideal gain vector. Each clustering solution \( c \) is scored for how well it has split the relevant set into clusters using CCG for the topic \( t \).

\[
SplitScore(t, c) = \frac{\text{cumsum}(CG)}{\text{Nr}^2}
\]  

(17)

\( \text{Nr} = \) Number of relevant documents in the returned result set for the topic \( t \).

A scenario with worst possible split is assumed to place each relevant document in a distinct cluster. Let CG1 be a vector that contains the cumulative gain of every cluster with each document.

\[
\text{MinSplitScore}(t, c) = \frac{\text{cumsum}(CG1)}{\text{Nr}^2}
\]  

(18)

The normalized cluster cumulative gain (nCCG) for a given topic \( t \) and a clustering solution \( c \) is given by,

\[
nCCG(t, c) = \frac{\text{SplitScore}(t, c) - \text{MinSplitScore}(t, c)}{1 - \text{MinSplitScore}(t, c)}
\]  

(19)

The mean and the standard deviation of the nCCG score over all the topics for a clustering solution are then calculated.

\[
\text{Mean}(nCCG(c)) = \frac{\sum_{t} nCCG(t, c)}{\text{Total Number of topics}}
\]  

(20)
The NCCG value varies from 0 to 1. A larger value of NCCG for a given clustering solution is better, since it represents the fact that an increased number of relevant documents are clustered together in comparison to a smaller number of relevant documents. Further details of this metric can be found in [4]

**Divergence from Random** Most measures of cluster quality can be tricked by changing the number of clusters or documents in the submission. The Purity and Entropy measures can be fooled if each document is placed in its own cluster. Every cluster becomes pure because it only contains one document. The NCCG measure can be fooled by creating one cluster with all the documents except for every other cluster containing one document. The NCCG measure orders clusters by the number of relevant documents they contain. A large cluster containing most documents will almost always be ranked first. Therefore, almost all relevant documents will exist in one cluster, achieving the highest score possible.

Any measure that can be tricked by creating a pathological clustering solution can be adjusted for by subtracting a cluster solution from a uniform randomly generated solution with the same number of clusters with the same number of documents in each cluster. Apart from how documents are assigned to clusters, the random baseline appears the same as the real solution. Therefore, each solution needs a uniform random baseline to be generated. This is done by shuffling the document IDs uniformly randomly and splitting them into clusters the same size as the solution being measured. The score for the uniform random solution is subtracted from the matching solution being measured. The graphs and tables in the following section contain the results for all metrics where this approach was taken.

The submissions this year from BUAP contained several large clusters and many other small clusters. This tricked the NCCG metric into giving arbitrarily high scores. When the scores are subtracted from a uniform random baseline with the same properties they performed no better than a randomly generated solution. This can be seen in Figure 7.

### 4.2 Clustering Participants, Submissions and Evaluations

The clustering tasks had submissions from three participants from Peking University, BUAP and Queensland University of Technology. The submissions labeled Random are a random solution that does not use any information about the documents. A cluster for each document is chosen uniformly at random from one of the k clusters required. Figures 4 to 7 graph the best performing submissions from each participant for Purity, Negentropy, NMI and NCCG. The divergence from random for each metric is also graphed. Figure 3 contains the legend for all these graphs.
The full details of the results are listed in tables in a separate document available from http://de-vries.id.au/inex10/full_results.pdf. The tables have been broken into sections matching the required numbers of clusters 50, 100, 200, 500 and 1000. Some submissions were outside the 5 percent tolerance of number of clusters. These form separate groups in the tables.

The group from Peking University [9] made a submission based on the structured link vector model (SLVM). It incorporated document structure, links and content. This year they focused on the preprocessing step for document structure and links. They modified two popular clustering algorithms, AHC clustering algorithm and K-means algorithm, to work with this model.

The group from BUAP [10] proposed an iterative clustering method for grouping the Wikipedia documents. The recursive clustering process iteratively brings together subsets of the complete collection by using two different clustering methods: k-star and k-means. In each iteration, they select representative items for each group which are then used for the next stage of clustering.

The group from the Queensland University of Technology used a 1024 bit document signature representation generated by quantizing random indexing or random projections of TF-IDF vectors. The k-means algorithm was modified to cluster binary strings of data using the hamming distance, including a different approach to calculating means of binary vectors.

5 Classification Task

The goal of the classification task was to utilize supervised or semi-supervised machine learning techniques to predict categories of documents from a set of known categories described in Section 3. The training set of documents contained 17 percent of the collection where each category had at least 20 percent of the category labels available.

5.1 Classification Evaluation Measures

Classification is evaluated using Type I and II errors made by classifiers. Each category is transformed into a binary classification problem. One category is evaluated at a time using all documents. The scores are calculated based on...
Fig. 4. Micro Purity

Fig. 5. Micro Negentropy

Fig. 6. NMI
the Type I and II errors and then micro and macro averaged. Micro averaging weights the average by the category size and macro averaging does not. Table 3 defines the Type I and II errors for a category.

<table>
<thead>
<tr>
<th>Predicted in Category</th>
<th>In Category</th>
<th>Not in Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>True Positive (tp)</td>
<td>False Positive (fp)</td>
<td></td>
</tr>
<tr>
<td>False Negative (fn)</td>
<td>True Negative (tn)</td>
<td></td>
</tr>
</tbody>
</table>

Table 3. Type I and II Classification Errors

The F1, Precision and Recall scores are calculated as described in Equations 22 to 24. F1 is the harmonic mean of precision and recall.

\[
F1 = \frac{2 \times tp}{2 \times tp + fn + fp} \quad (22)
\]

\[
\text{Precision} = \frac{tp}{tp + fp} \quad (23)
\]

\[
\text{Recall} = \frac{tp}{tp + fn} \quad (24)
\]

5.2 Classification Participants, Submissions and Evaluations

Two groups from Peking University and the Queensland University of Technology (QUT) made submissions for the classification task. The results are listed in Table 4.

The group from Peking University [9] made a submission based on the the structured link vector model (SLVM). It incorporated document structure, links
and content. This year they focused on the preprocessing step for document structure and links.

The group from QUT made a submission using content only to provide a baseline approach. Documents were represented in the bag of words vector space model using the BM25 weighting for each term where the tuning parameters $K_1 = 2$ and $b = 0.75$. A Support Vector Machine (SVM) was used to classify each document by treating each category as a binary classification problem.

<table>
<thead>
<tr>
<th>Submission</th>
<th>F1</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Micro</td>
<td>Macro</td>
<td>Micro</td>
</tr>
<tr>
<td>QUT BM25 SVM</td>
<td>0.536</td>
<td>0.473</td>
<td>0.562</td>
</tr>
<tr>
<td></td>
<td>0.523</td>
<td>0.440</td>
<td></td>
</tr>
<tr>
<td>Peking tree1 sim3 linkTxt</td>
<td>0.460</td>
<td>0.380</td>
<td>0.553</td>
</tr>
<tr>
<td></td>
<td>0.535</td>
<td>0.436</td>
<td>0.359</td>
</tr>
<tr>
<td>Peking tree2 sim3 linkTxt</td>
<td>0.452</td>
<td>0.371</td>
<td>0.562</td>
</tr>
<tr>
<td></td>
<td>0.423</td>
<td>0.321</td>
<td></td>
</tr>
<tr>
<td>Peking tree1 sim1 linkTxt</td>
<td>0.399</td>
<td>0.314</td>
<td>0.582</td>
</tr>
<tr>
<td></td>
<td>0.363</td>
<td>0.252</td>
<td></td>
</tr>
<tr>
<td>Peking tree2 sim2</td>
<td>0.508</td>
<td>0.435</td>
<td>0.422</td>
</tr>
<tr>
<td></td>
<td>0.653</td>
<td>0.612</td>
<td></td>
</tr>
<tr>
<td>Peking tree1 sim2</td>
<td>0.521</td>
<td>0.452</td>
<td>0.480</td>
</tr>
<tr>
<td></td>
<td>0.574</td>
<td>0.510</td>
<td></td>
</tr>
<tr>
<td>Peking tree2 sim3</td>
<td>0.518</td>
<td>0.444</td>
<td>0.443</td>
</tr>
<tr>
<td></td>
<td>0.635</td>
<td>0.582</td>
<td></td>
</tr>
<tr>
<td>Peking tree1 sim3 linkTxt N</td>
<td>0.517</td>
<td>0.444</td>
<td>0.432</td>
</tr>
<tr>
<td></td>
<td>0.656</td>
<td>0.613</td>
<td></td>
</tr>
<tr>
<td>Peking tree2 sim2 linkTxt N</td>
<td>0.521</td>
<td>0.454</td>
<td>0.456</td>
</tr>
<tr>
<td></td>
<td>0.615</td>
<td>0.559</td>
<td></td>
</tr>
</tbody>
</table>

Table 4. Classification Results

6 Conclusion

The XML Mining track in INEX 2010 brought together researchers from Information Retrieval, Data Mining, Machine Learning and XML fields. The clustering task allowed participants to evaluate clustering methods against a real use case and with significant volumes of data. The task was designed to facilitate participation with minimal effort by providing not only raw data, but also pre-processed data which can be easily used by existing clustering software. The classification task allowed participants to explore algorithmic, theoretical and practical issues regarding the classification of interdependent XML documents.
References

Chapter 12

TopSig: topology preserving
document signatures

In this paper I contributed all sections related to document clustering with TopSig document signatures. I proposed and implemented the variant of k-means that performs optimization directly with the binary signatures produce by TopSig. Outside of the sections related to document clustering, I also contributed to the writing of the manuscript, experimental design, execution of experiments, and data analysis.

12.1 Representations

This paper introduces the TopSig document signature representation. TopSig produces binary strings, bit vectors or document signatures by applying random indexing or random projections and numeric quantization to document vectors. This creates a dense compressed representation of documents. The compressed binary representation of TopSig is one of the factors that allows the improvement of computational efficiency of clustering approaches outlined in this thesis. The binary representation allows efficient processing on 64-bit processors by processing 64 dimensions of each bit vector in a single CPU instruction. The compressed nature of the representation allows for less data to be processed than when using other vector representations for document clustering. This also improves the performance and scalability of clustering approaches as less data has to be processed.
12.2 Algorithms

This paper introduces a modified version of the k-means algorithm that works directly with the binary string representation of TopSig. All cluster centers and documents are binary strings. This allows the Hamming distance measure to be used for comparison which can be computed 64 dimensions at a time using 64-bit processors. This allows a one to two order of magnitude increase in computational efficiency when compared to the sparse vector approaches used in the popular clustering toolkit, CLUTO. When using 4096 bit document signatures, the quality of the clustering is not statistically significantly different to the sparse vector approach used in CLUTO.
TopSig: Topology Preserving Document Signatures

Shlomo Geva
Faculty of Science and Technology
Queensland University of Technology
Brisbane, Australia
s.geva@qut.edu.au

Christopher M. De Vries
Faculty of Science and Technology
Queensland University of Technology
Brisbane, Australia
chris@de-vries.id.au

ABSTRACT
Performance comparisons between File Signatures and Inverted Files for text retrieval have previously shown several significant shortcomings of file signatures relative to inverted files. The inverted file approach underpins most state-of-the-art search engine algorithms, such as Language and Probabilistic models. It has been widely accepted that traditional file signatures are inferior alternatives to inverted files. This paper describes TopSig, a new approach to the construction of file signatures. Many advances in semantic hashing and dimensionality reduction have been made in recent times, but these were not so far linked to general purpose, signature file based, search engines. This paper introduces a different signature file approach that builds upon and extends these recent advances. We are able to demonstrate significant improvements in the performance of signature file based indexing and retrieval, performance that is comparable to that of state of the art inverted file based systems, including Language models and BM25. These findings suggest that file signatures offer a viable alternative to inverted files in suitable settings and from the theoretical perspective it positions the file signatures model in the class of Vector Space retrieval models.

Categories and Subject Descriptors
H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval—Retrieval Models, Relevance Feedback, Search Process, Clustering

General Terms
Algorithms, Experimentation, Performance, Theory

Keywords
Signature Files, Random Indexing, Topology, Quantisation, Vector Space IR, Search Engines, Document Clustering, Document Signatures

1. INTRODUCTION
Document signatures have been largely absent from mainstream IR publications about general-purpose search engines and ranking models for several years. The decline in the attention paid to this approach, which had received a lot of attention earlier, started with the publication of the paper “Inverted Files Versus Signature Files for Text Indexing” by Zobel et al [25]. This paper offers an extensive comparison between Signature Files and Inverted Files for text indexing. The authors have systematically and comprehensively evaluated Signature files and Inverted File approaches. Having examined several general approaches they concluded that inverted files are distinctly superior to signature files. Signature files are found, in their studies, to be slower, to offer less functionality, and to require larger indexes. They conclude that the Bit Sliced signature files under-perform on almost all counts and offer very little if any advantages over inverted files. Further discussion of signature files is offered in [21], and a similar picture emerges there too. It is clear from the experimental evidence that Bit Sliced signatures are not able to compete with state of the art inverted file approaches in terms of retrieval performance. Furthermore, the presumed advantages of efficient bit-wise processing and the potential for index compression are not generally achievable in practice. Signature files are found to be larger than inverted file indexes. This is perhaps surprising because Signature files were largely motivated by the desire to represent entire documents as relatively short bit strings, and having fixed the signature length, the document signature is independent of actual document length. As it turns out, it is not possible to achieve competitive performance goals with compact signatures and consequently signatures require even more space than compressed indexes.

For the sake of completeness, and since we offer a radically different approach to the construction of file signatures, it is necessary to describe the conventional approach first. Conventional Bit Sliced signature files, as described in [7] exploit efficient bit-wise operators that are available on standard digital processors. Unlike probabilistic models of IR and Language Models, the Signature File approach is presented in an ad-hoc manner and is computationally motivated by efficient bit-wise processing, and without specific grounding in Information Retrieval theory. In traditional signature files a document is allocated a fixed-size signature of N bits. Each term that appears in the collection is assigned a random signature of width N, where only a small number of n << N bits are set to 1 with the use of a suitable hash function. Naturally, term signatures tend to collide on some bit positions, but this is of course unavoidable unless the number N is extremely large, and the number n is very small. Given that the vocabulary of a document collection typically contains millions of distinct terms, collisions will occur, and frequently. The document signature in this approach is derived as the conjunction of all the term signatures within the document (bit-wise ORed). Query terms are similarly assigned a signature. A query is then evaluated
by comparing the query signature to each document signature. Any document whose signature matches every bit that is set in a query term, is taken as a potential match. It is only a potential match because hash collisions in generating term signatures can lead to false matches — situations where all the bits match, but the actual query term is not present in the document. Consequently, documents are retrieved and checked directly against the query to eliminate false matches. This is a very expensive operation even if the collection fits in memory, but with a disk based collection — the most likely scenario — this is prohibitively expensive.

Indeed, the method used in traditional file signatures is known in other domains as a Bloom Filter. B.H. Bloom in first described bloom filters 1970 [4], and this well predates file signatures.

It is clear from the experimental evidence that Zobel et al [25] and Witten et al [21] provide, that such signatures are not likely to compete with state of the art inverted file approaches in terms of retrieval performance. Furthermore, the presumed advantages of efficient bit-wise processing and the potential for index compression are not generally achieved in practice. Signature files are found to be larger than inverted file indexes.

Recent approaches to similarity search [23] have explored similar ideas to TopSig for mapping documents to N bit strings for comparison using Hamming distance. The approach taken by Zhang et al [23] and prior publications focus on similarity comparisons between documents. Their models have not been applied to general-purpose ad-hoc retrieval. More importantly, Zhang et al [23] use a complicated approach to the static, off-line derivation of signatures, and which involves supervised and unsupervised learning to generate document signatures. This in effect prevents the application of the approach to ad-hoc retrieval where a query signature has to be derived at run-time. It is not practical in a very large collection due to the excessive computational load of supervised and unsupervised learning.

Unlike earlier attempts, we approach the design of TopSig document signatures from basic principles. TopSig is radically different from a Bloom filter in the construction of file signatures and in the manner in which the search is performed. We present results of extensive experiments, performed with large standard IR collections, where we compare TopSig with standard retrieval models such as BM25 and various Language Models. We also describe document clustering experiments that demonstrate the effectiveness of the approach relative to standard document representation for clustering.

The remainder of this paper is organised as follows. Section 2 introduces the TopSig approach in detail. Sections 3, 4 and 5 define and evaluate the use of this approach for ad-hoc retrieval and clustering. The paper is concluded with a discussion in Section 6.

2. TOPSIG

TopSig represents a radically different approach to the construction of signature files. Unlike the traditional ad-hoc approach [7], TopSig is principled and signature files emerge naturally from a highly effective compression of the well understood and commonly used Vector Space representation of documents.

We approach the design of document signatures from the perspective of dimensionality reduction. TopSig starts from a straight forward application of a vector space representation of the collection — the term-by-document weight matrix. We then derive the signatures through extreme and lossy compression, in two steps, to produce topology preserving binary document signatures. While the actual mechanism that is proposed is highly efficient in signature construction and in searching, we first focus the discussion on the conceptual approach, its justification and theoretical grounding, while leaving the implementation and performance analysis details for later in the paper.

In this section we describe the concepts that underpin TopSig. These concepts are not new — Random Indexing and Numeric Quantisation — but when put together to form file signatures, the results are remarkable.

2.1 Random Indexing vs LSA

Latent Semantic Analysis [8] is a popular technique that is used with word space models. LSA [8] creates context vectors from a document-by-term occurrence matrix by performing Singular Value Decomposition (SVD). Dimensionality reduction is achieved through projection of the document-by-term occurrence vectors onto the subspace spanned by relatively few vectors having the largest singular values in the decomposition. This projection is optimal in the sense that it minimises the Frobenius norm of the difference between the original and the projected matrix. SVD is computationally expensive and this limits its application in large collections. For instance, in our own experiments, the SVD of a collection of 25,000 English Wikipedia articles — less than 1% of the collection — using the highly efficient parallel multi-processor implementation of the MATLAB svds function, took about 7 hours on a top-end quad-processor workstation with sufficient memory to be completely processor bound.

Random Indexing (RI) [17] is an efficient, scalable and incremental approach to dimensionality reduction. Word space models often use Random Indexing as an alternative to Latent Semantic Analysis. Both LSA and RI start from the term-by-document frequency matrix. Often term frequencies are replaced by term weights — for instance, one of the many TF-IDF variants. With LSA, Singular Value Decomposition is used to derive an optimal projection onto a lower dimensional space. Random Indexing is based on a random projection - avoiding the computational cost of matrix factorisation. Having obtained a projection matrix, both LSA and RI proceed to project the term occurrence matrix onto a subspace of significantly reduced dimensionality.

In practice, RI works with one document at a time, and one term at a time within the document. Terms are assigned random vectors, and the projected document vector is then the arithmetic sum of all term signatures within. The process is somewhat similar to the traditional signature file approach of [7], but the document vector is real valued; it is a superposition of all the random term vectors. There is no matrix factorisation and hence the process is efficient. It has linear complexity in the number of terms in a document and also in the collection size. This is a significant advantage over LSA whose time complexity is prohibitive in large collections. As stated by Manning et al [14] in 2008, in relation to LSA — “The computational cost of the SVD is significant; at the time of this writing, we know of no successful experiment with over one million documents".
The RI process is conceptually very different from LSA and does not carry the same optimality guarantees. At the foundation of RI is the Johnson-Lindenstrauss lemma [9]. It states that if points in a high-dimensional space are projected into a randomly chosen subspace, of sufficiently high-dimensionality, then the distances between the points are approximately preserved. Although strictly speaking an orthogonal projection is ideal, nearly orthogonal vectors can be used and have been found to perform similarly [3]. These vectors are usually drawn from a random uniform distribution. This property of preserving relative distances between points is useful when comparing documents in the reduced space. RI offers dimensionality reduction at low computational cost and complexity while still preserving the topological relationships amongst document vectors under the projection.

In RI, each dimension in the original space is given a randomly generated index vector. The index vectors are high dimensional, sparse, and ternary. Sparseness is controlled via a parameter that specifies the number of randomly selected non-zero dimensions. Ternary term vectors consist of randomly and sparsely distributed +1 and -1 values in a vector that otherwise consists mostly of zeros. This choice ensures that the random vectors are near orthogonal.

RI can be expressed as a matrix multiplication of a randomly generated term-by-signature matrix $T$ by a term-by-document matrix $D$ where $R$ is the randomly projected term-by-document matrix.

$$R_{N \times d} = T_{N \times t} D_{t \times d}$$ (1)

Each of the $d$ column vectors in $D$ represents a document of dimensionality $t$, each of the $t$ column vectors in $T$ is a randomly generated term vector of dimensionality $N$. $R$ is the reduced matrix where each of the $d$ column vectors represents a randomly projected document vector of dimensionality $N$.

RI has several advantages over LSA. It can be performed incrementally and online as data arrives. Any document can be indexed independently from all other documents in the collection. This eliminates the need to build and store the entire term-by-document matrix. Additionally, newly encountered terms are naturally accommodated without having to recalculate any of the projections of previously encoded documents. By contrast, LSA requires global analysis where the number of documents and terms are fixed. The time complexity of RI is also very attractive. It is linear in the number of terms in a document and hence linear in the collection size. RI makes virtually no demands on computer memory since each document is indexed in turn and the signatures are independent of each other.

TopSig deviates from Sahlgren’s basic Random Indexing by introducing term weights into the projection. In Sahlgren’s scheme, the term-by-document matrix contains unweighted term counts. Search engine evaluation consistently shows that unweighted term frequencies do not produce the best performance. Better results are obtained if the terms frequencies are weighted and this of course underlies the most successful search engine models, such as BM25 and Language models. The weighting of terms in TopSig is described in Section 2.3.

Term weighting has an apparent drawback – it may appear to compromise the ability to encode new documents independently. The calculation of term weights, such as with TF-IDF or Language Models, requires global statistics. We observe however that in a large collection new documents have very little impact on these global statistics. Upon inserting a new document these global statistics are updated and the new document is encoded. As the collection grows, it is periodically re-indexed from scratch to bring all signatures into line, but this is a relatively cheap operation. On a modern multi-processor PC using the ATIRE search engine [18] we can index the entire English Wikipedia of 2.7 million documents, spanning 50 gigabytes of XML documents, in under 15 minutes.

### 2.2 Random Indexing and Other Approaches

Random Indexing shares many properties with other approaches. In this section we will highlight some of the more interesting properties shared with other dimensionality reduction approaches.

RI or random projections are closely related to compressed sensing from the field of signal processing. Compressed sensing is able to reconstruct signals with less samples than required by the Nyquist rate. Baraniuk et al [2] construct a proof showing how the Restricted Isometry Property that underlies compressed sensing is linked to the Johnson-Lindenstrauss lemma [9] which underlies RI.

A conceptually similar approach to RI is used for a spread spectrum approach in Carrier Division Multiple Access [16]. In contrast, CDMA uses orthogonal vectors for codes and increases the bandwidth of the signal. In CDMA, the use of random orthogonal codes allows for division of the radio spectrum that is more resistant to noise introduced in radio frequency transmission.

Many other approaches to dimensionality reduction exist. Again, many come from the field of signal processing. Many of these approaches iteratively optimise an objective function. LSA offers an optimal linear projection in preserving the Frobenius norm. Other well known approaches include the Discrete Cosine Transform, Wavelet Transform, Non-Negative Matrix Factorisation, Principal Component Analysis and Cluster Analysis. The advantage to RI is that it still preserves the topological relationships between the vectors without having to directly optimise an objective function. This is where its computational efficiency comes from.

### 2.3 TopSig Signatures

Document Signatures are fixed length bit patterns. In order to transform the real-valued projected term-by-document matrix into a signatures matrix, we ask what numerical precision is required to represent the term-by-document matrix. It is obvious that there is no need for double precision and one obtains identical results when evaluating searching or clustering performance with single precision. One quickly finds that even when scaling the values to the range $[0,255]$ – i.e. a single byte – there is no appreciable difference. Even Nibbles (4-bit integers) have been shown to be sufficient with little appreciable difference in performance. This is exploited by all state of the art search engines to compress indexes. The reduction in precision still leaves the term-by-document matrix with a highly faithful representation of the similarity relationships between the original documents.

Both clustering and ranking applications are concerned not with the actual similarity values, but rather with their rank.
order. As long as rank order is preserved the distortion due to reduced numerical precision is not problematic.

In section 2.1 we described how a real-valued document vector is obtained through random projection, as the sum of random term signatures within. TopSig now takes the reduction in numerical precision to its ultimate conclusion, by taking this real-valued randomly indexed document, and reducing the precision all the way to a single bit. Binary signatures are obtained by taking only the sign-bits of the projected document vectors (!!). This is a key step in TopSig signature calculation; it may appear to be highly excessive precision reduction, but it is in fact surprisingly effective, as we shall demonstrate with search and clustering experiments in the following sections.

2.3.1 Topological Distortion

In order to measure the impact of aggressive dimensionality reduction we conduct the following experiment. We take 1000 randomly chosen Wikipedia document vectors, in full TF-IDF representation, and compute their mutual distance matrix. Each element in the matrix represents the distance between a pair of document vectors in the full space. We then randomly project the vectors onto a lower dimensional subspace and compute the corresponding mutual distance matrix in the projection subspace. The mutual distance matrices are normalised such that the sum of elements in each matrix is equal to 1. If the mutual distances are perfectly preserved then the normalised matrices will be identical. However, with aggressive compression we expect a topological distortion due to information loss. To measure the impact, we calculate the topological distortion as the root mean squared differences (RMSE) between distances in full precision, and the corresponding distances in the reduced dimensionality and reduced precision. This calculation is performed for various dimensionality reduction values and various numeric precision values.

Figure 1 depicts the results of our experiment. On the y-axis is the topological distortion, measured by RMSE. On the x-axis is the number of dimensions in the projection. Each of the curves on the plot corresponds to a different numerical precision. The bottom curve corresponds to double precision, and then the plots above correspond to 8-bit quantisation, through 7-bit quantisation, and so on all the way down to 1-bit quantisation. First we observe that as the dimensionality of the projected subspace is increased (moving to the right with the curves), the distortion becomes smaller. This is true regardless of numerical precision and it is expected. We also observe that most of the gain is achieved quite early with relatively small dimensionality. This is the expected behaviour of both RI and LSA, where a relatively small number of dimensions typically is required to achieve good results with text documents. What is perhaps less expected is that as we reduce the numerical precision the deterioration is very small. The lowest curve in Figure 1 corresponds to double precision. It is only when precision is dropped to 3-bit that the difference in RMSE becomes noticeable. The curves from 8-bit down to 4-bit quantisation are barely separated. The distortion only increases significantly when we drop to 3-bit, 2-bit and 1-bit precision, corresponding to the 3 higher curves in the figure. Even with 1-bit precision we are still able to significantly preserve topology quite early with very small dimensionality.

2.3.2 Packing Ternary Vectors onto Binary Strings

To complete the generation of a document signature we need to pack the \( \pm 1 \) representation of signatures, onto binary strings. This is done by representing positive signs as 1s, and negative signs as 0s. The final result is thus a binary digital signature, but it still conceptually represents \( \pm 1 \) signatures.

We note that there is a possibility that very short documents will not occupy all bit positions in a signature. We can safely ignore this situation and encode zeros as positive (i.e., binary 1) although it may introduce some noise into the representation. The effect is negligible and studying it is outside the scope of this paper. Suffice to say that in our experiments circumventing this by complicating the representation to also record the unoccupied positions resulted in no appreciable difference at all and there was no practical advantage to maintaining this information.

2.3.3 Summary of Binary Signatures

To summarise, TopSig introduces a principled approach to the generation of binary file signatures. The underlying data representation starts exactly as with inverted files, from the term-by-document weight matrix. This matrix is then subjected to aggressive lossy compression. Topology preserving dimensionality reduction is first achieved through Random Indexing and it is immediately followed by aggressive numerical precision reduction by keeping only the sign bits of the projected term-by-document weight matrix. Unlike traditional signatures, TopSig does not emerge from bit-wise processor efficiency considerations, but rather, it emerges as a consequence of aggressive compression of a well understood document representation. In this scheme, document signatures are no more than highly concise approximations of vector space document representations. TopSig maps an entire document collection onto corners of the \( \{ \pm 1 \}^N \) hypercube.

3. AD-HOC RETRIEVAL

To provide a concrete description of the implementation and use of TopSig in ad-hoc retrieval we need to more precisely define document and query signatures, term weights, the ranking process, and how pseudo-relevance feedback is
used. We then describe the evaluation of document retrieval using the INEX Wikipedia collection and the TREC Wall Street Journal (WSJ) collection. We conclude this section with the description of document clustering experiments.

### 3.1 Document Signatures

So far we have not addressed the weighting of terms in the vector space representation of the term-by-document matrix. Weighting is all-important to improving precision and recall and it is the basis of the most successful ranking functions, such as BM25 and Language models, which compute term weights in many different ways. With TopSig, the most effective weighting function we have found is described in Equations (2), (3) and (4)

\[
P(t|D) = \frac{tf(D)}{|D|} \tag{2}
\]

\[
P(t|C) = \frac{tef(C)}{|C|} \tag{3}
\]

\[
W(t, D) = \log \left( \frac{P(t|D)}{P(t|C)} \right) \tag{4}
\]

where \(W(t, D)\) is the weight for term \(t\) in document \(D\). We define \(tf(D)\) to be the term frequency for term \(t\) in document \(D\), \(|D|\) as the total number of term occurrences in document \(D\), \(tef(C)\) as the collection frequency for term \(t\), and \(|C|\) as the total number of term occurrences in the collection. \(P(t|D)\) is an estimate of the probability of finding the term \(t\) given a document \(D\), and \(P(t|C)\) is an estimate of the probability of finding term \(t\) given the collection \(C\).

The weighting function \(W(t, D)\) produces a larger value if the frequency of a term in a document is higher than expected, and smaller if the frequency is lower than expected. The logarithm of the ratio of these expected values is taken, so as to dampen the effect of an inordinately large frequency of a term in a document.

The representation of a document is thus a bag of words, where the weight assigns an individual importance score to each term within a document. This effectively takes care of stop-words. We note that a term that occurs with approximately the expected frequency will have a weight close to zero. Negative weights that result from equation 4 are set to zero since that would indicate that the term occurs in the document with even lower frequency than expected. This weighting scheme ensures that stop words are naturally discounted without special treatment. Anecdotally, a document that consists of only the sentence “To be, or not to be, that is the question” will retain all terms with appreciable weights when generating this document’s signature, but most terms will have virtually negligible weights in much larger documents and thus the terms will be effectively stopped.

### 3.2 Alternative Term Weighting Functions

Surprisingly, TopSig performs quite respectably with no term weighting at all. The raw unweighted term frequencies and simply randomly indexed. One advantage of this approach is that is requires no global statistics at all – a document can be encoded purely by looking at the document in isolation.

When using the BM25 weighting function to create a vector space representation, we found the retrieval performance was relatively very poor. This is not surprising as BM25 was originally intended to be treated as a probabilistic model and we did not use it in that manner.

The TF-IDF representation produces retrieval quality that lies between raw term frequencies and the approach described in Section 3.1.

The detailed comparison of different weighting functions is outside the scope of this paper. What we provide here is anecdotal evidence to paint a clearer picture of the approach we have taken to developing TopSig.

### 3.3 Alternative Document Representations

While the representation we have described here for ad-hoc retrieval is a bag of words model for keyword search, there is no limitation of encoding other representations using the TopSig approach. For example, it is possible to create vector space representations of structured data such as XML and other textual features such as phrases. As with many popular machine learning approaches, most increases in quality with respect to human judges come from how the data is represented.

### 3.4 Choice of Sparsity Parameters

During our experiments we found that setting the sparsity of the random codes for each term to 1 in 12 set to +1 and 1 in 12 to -1 worked most effectively. As the density of the random codes or index vectors increases, the potential for cross talk between the codes increases. When the sparsity is decreased too far there is not enough information for the query to successfully match against. There is an optimal point for sparsity with respect to a given set of queries. Detailed analysis of the effect of sparsity, including automated methods to learn the optimal sparsity are outside the scope of this paper and will be investigated in further research.

### 3.5 Query Signatures

In order to search the collection with a given query, we need to generate a query signature. Query document vectors are generated using standard TF-IDF weighting. This real valued query vector is then converted to a signature using exactly the same process as used with document signatures.

All the weighted query term signatures are added to create a real valued randomly projected document. The sign-bits are then taken to form the binary signature. It is of course necessary to use exactly the same process and parameters in generating the query signature as when generating document signatures.

The use of term weights in generating the query ensures that query terms that are a-priori more significant (as determined through TF-IDF or some other weighting function) will tend to dominate the signature bits where there is a collision and a conflict. Of course there is no need for concern when the two terms agree on the sign when there is a collision. This is easily understood by looking at a case where we have two query terms, for instance, “space” and “shuttle”. If the term “shuttle” has a larger TF-IDF value then for any bit position where the two terms disagree on the sign, the term “shuttle” will dominate the sign in the signature. When there are multiple terms we effectively get a vote.

Document signatures are represented in binary form, where 1-bits correspond to +1, and 0-bits correspond to -1. Query signatures, before taking the sign bits, may contain a mix
of 3 classes of values: positive, negative, or zero. This depends on the signs of term signatures, and a value of zero is obtained when none of the query terms occupy some bit positions. As a matter of fact, with short queries and sparse term signatures this is almost invariably the case. These zero valued bit positions are those for which the query does not specify any preference. To account for this, a query mask is also generated to accompany the query signature. This mask has 1-bits in all positions other than those that are not covered by any term in the query. The set bits in the mask identify the subspace in signature space which the query terms cover. When comparing the query signature against document signatures, the similarity measure must not take account of differences in those bit positions. Conceptually, these are neither +1 nor −1.

3.6 Ranking

Ranking with TopSig is performed with the Hamming distance, calculating the similarity score for each document. The Hamming distance is rank equivalent to the Euclidean distance since all signatures have the same vector length – we note that the signatures correspond to +1 and −1 values, not 1 and 0 values, and hence the length of each signature of N bits is \( \sqrt{N} \). Since the mask is almost invariably different for each query, the Hamming distance for each query will generally be calculated in a different subspace. The distance metric is therefore a masked Hamming Distance.

If the document and query are identical in the query subspace then the Hamming distance will be zero. The Hamming distance between two signatures of N bits is restricted to the range \([0, 1, 2, \ldots N]\). For a signature file with 1024 bits per document there are at most 1025 possible distances between the query and a document, and many less if the query is short. This means that in a collection such as the Wikipedia, with millions of documents, if we rank all the documents by the Hamming distance from the query, we are bound to get numerous ties.

Although document signatures are not completely random – they are biased by the document contents, and similar documents have similar signatures – we still expect the vast majority of the documents to be centred at about a Hamming distance of \( N/2 \) from the query signature. Indeed this is always observed. The distribution of distances always resembles a binomial distribution, which we expect if the distribution of signatures was indeed random. It is not quite that, but we still observe strong resemblance to truly random distribution.

We are interested in early precision and so TopSig can still achieve granularity in ranking of documents. This is because a large number of documents fall much closer than \( N/2 \) to the query signature, and the number of ties diminishes rapidly as the distance becomes smaller. Some ties still remain nevertheless and these may be broken arbitrarily or by using simple heuristics or document features. For instance, page-rank can be used, or any one of hundreds of document features that are reportedly used in commercial search engines.

3.6.1 Partial Index Scanning

Given an index where each document signature is N bits wide it is possible scan only the first J bits of each signature. This allows for further decreases in time taken to rank. A multiple pass approach is possible where the documents are first ranked with relatively few bits such as 640. The top ten percent of the documents ranked using 640-bits can then be re-ranked using the full precision of the document signatures stored in the index.

3.7 Relevance Feedback

Pseudo relevance feedback is known to improve the performance of a retrieval system. TopSig can implement pseudo-relevance in the usual manner, through query expansion. This however is a generic approach and can be used with any search engine. There is however an additional opportunity to apply pseudo-relevance feedback, an opportunity that is unique and specific to TopSig. Explanation of pseudo-relevance feedback is required to completely describe the approach we have taken to ad-hoc retrieval with TopSig.

An initial TopSig search is first executed in the manner previously described in Section 3.6. This search is performed in the subspace of the query signature, the subspace spanned by the query terms. This is achieved by using the masked Hamming distance to rank all the documents in the collection. Now it is possible to proceed and apply pseudo relevance. The principle is the same as with all pseudo relevance approaches – use some of the top ranked results to inform a subsequent search.

We take the top- \( k \) ranked documents and create a new query signature by computing the arithmetic mean of the corresponding signatures by treating the signatures as integer valued vectors and then taking the sign-bits in the manner described in Section 2.3.2. Since this signature was is generated from full document signatures, this signature is now spanning the full signature space and takes into account information from highly ranked results, including in bit positions that were not informed directly by the query terms. Now the query signature is in fact based on the full content of the nearest \( k \) documents, through their signatures. The new query is constructed by inserting only the missing bits into the original signature. Therefore, the new signature consists of the original signature in all originally unmasked positions, and the feedback signature in all previously masked positions.

The ranking of documents in relation to the new query is then repeated, but it is not necessary to search the entire collection again. It is sufficient to re-rank a very small fraction of the nearest signatures – usually those that were retained in a shortened result list following the initial search. This step is consuming a negligible amount of additional computation – several orders of magnitude less than the initial search. The feedback leads to statistically significant improvement in performance.

The approach to pseudo-relevance feedback we have described exploits the binary representation used by TopSig. This is conceptually similar to standard pseudo-relevance feedback where the goal is to learn meaningful weights for relevant terms not in the query. However, the implementation of the approach with TopSig is drastically different as we work directly in the dimensionality reduced space of the binary document signatures, rather than with specific terms not in the original query.

4. Evaluation and Results

We have evaluated TopSig using the INEX Wikipedia 2009 collection, and the TREC Wall Street Journal (WSJ) Collection. INEX Wikipedia collection contains 2,666,192 doc-
uments with a vocabulary of 2,132,352 terms. The mean document length in the Wikipedia has 360 terms, the shortest has 1 term and the longest has 38,740 terms. We used all 68 queries from INEX 2009 for which there are relevance judgments. The Wall Street Journal Collection consists of 173,252 documents and a vocabulary of 113,288 terms. The mean WSJ document length is 475 terms, the shortest has 3 terms, and the longest has 12,811 terms. We used TREC WSJ queries 51-100.

To compare TopSig with state-of-the-art approaches, we have used the ATIRE search engine [18] which was formerly known as ANT. ATIRE is a highly efficient state-of-the-art system which implements several ranking functions, over an inverted file system. The ATIRE search engine has been thoroughly tested at INEX against other search engines, including several well known systems such as Zettair, Lucene, and Indri, and has been shown to produce accurate and reliable results.

The references given herein to the ranking functions that were compared with TopSig, are to the actual papers that were followed in implementing the methods, as documented in the ATIRE search engine manual. These are Jelinek-Mercer (LMJM) [22], DLH13 [12]. Divergence from Randomness [1], and Bose-Einstein [1]. The ranking functions were evaluated with relevance judgments from TREC and INEX, and the trec_eval program.

4.1 Recall-Precision

We first look at recall and precision over the full range of recall values. Figure 2 depicts the precision-recall curves for INEX 2009 topics, against a tuned BM25 system, using $k = 1.1$ and $b = 3$, and with Rocchio pseudo relevance feedback. This BM25 baseline curve is an optimistic over-fitted approach – it is tuned with the actual queries, and indeed performs better than any official run at INEX 2009. But we are concerned with evaluating TopSig and so this provides a very conservative yardstick by which to measure the performance. The figure shows several TopSig indexes, encoding the signature with 64, 128, 256, 512, 768, 1024, 2048, 3072, and 4096 bits per signature. Only one in 12 vector elements were set in the random term signatures, to either +1 or to −1, with the rest of the elements set to 0. It is interesting that even a 64 bit signature produced measurable early precision. As the number of bits in the signature increases, so does the recall. The performance of the file signatures is quite respectable once we allow for about 512 bits per signature – particularly at early precision.

All the other language model based ranking functions produce a recall-precision curve that falls below BM25, and just above the best TopSig curve, but are not shown on the plot so as to reduce the clutter. Note that the legends in the figures are ordered in decreasing order of area under the curve.

4.2 Early Recall

While Figure 2 may at first suggest that file signatures produce inferior retrieval quality, we must focus our attention on the early precision, and this requires some justification before we do that.

Moffat and Zobel [15] found that P@n correlates with user satisfaction. A user who is given 7 relevant documents in the top 10 is better off than one who is only given 2. They argue that recall does not have a similar use case that reflects user satisfaction. Even for a recall oriented task, a user is unlikely to look past the top 30 results. For most tasks, the first page or top 10 results are most useful to the user. Users achieve recall not through searching the entire ranked list but by reformulating queries. Recent work by Zobel and Moffat [24] suggests recall is not important except for a few recall oriented tasks such as retrieval in medical and legal domains. If a system provides 100% recall, it implies that the user can create a perfect query. Even in recall-based tasks, users tend to re-probe the collection with multiple queries to minimise the risk they have missed important documents.

The same argument is applied to discount the importance that is attributed the commonly used measure of Mean Average Precision (MAP) as it too depends on higher recall and a long tail of relevant results. Again, it is not clear what user satisfaction is correlated with MAP. Turpin and Scholer [19] performed retrieval experiments where users completed search tasks using search results with MAP scores between 55% and 95%. They were unable to find a correlation between MAP scores and a precision based task requiring the first relevant document to be found. For recall-based tasks, they only found a weak link between MAP and the number of relevant documents found in a given time period. They conclude that MAP does not correlate with user performance on simple information finding web search tasks.

4.3 Analysis of Early Recall

Recall is not likely to be important to users except in some specific domains. Therefore, we focus our attention on comparison of P@n results between TopSig and state of the art inverted file approaches. The results immediately make it obvious that TopSig is a viable option for common information finding tasks.

To assess TopSig at early precision we look at early precision in the P@n plots on Figures 3 and 4, for the 68 INEX 2009 ad-hoc queries and the TREC Wall Street Journal queries 51-100. It is immediately clear that TopSig performs similarly. The only system that consistently outperforms TopSig is the over-fitted BM25 baseline. The legends in the figures are ordered by the area under the curve, so that the best performing systems appear first in the legend.

In order to look more carefully at the differences, we focused on the P@10 performance differences on the INEX
Figure 3: INEX 2009 P@n

Figure 4: Wall Street Journal P@n

Figure 5: INEX 2009 P@10 by Topic

Wikipedia collection, between the best performing ranking function – BM25, and TopSig with 4096 bit signatures. The average P@10 for BM25 is 0.54, and for TopSig it is 0.51. We look at all 68 queries and performed two-tailed paired t-test. There is no statistically significant difference with p = 0.41. Figure 5 depicts the P@10 values for all 68 queries. The topics on the X-axis are ordered by increasing P@10 values for BM25. The TopSig P@10 values are plotted in the same order. It is obvious that the two approaches produce very different results on a per-topic comparison. The two systems do not agree on which topics are difficult and which are not, and both sometime fail (on different topics) to produce any relevant result in the top-10. It is a common and well understood phenomena that this should occur and it is true for all the ranking functions that we tested. However, there is a much stronger correlation between all the language models, and BM25, as to which topics are hard and which are easy. No such correlation is observed for TopSig which seems to behave quite differently despite producing similar overall precision. This leads us to conjecture that combining TopSig with BM25 (or any of the other models tested) may lead to better results than emerge from combining any other pair of more correlated ranking functions. Testing this conjecture is outside the scope of this paper.

By inspecting at Figures 2 through 4 one can observe that as the number of bits in a document signature increases, the quality of the results increases logarithmically. As more and more bits are added the increases in quality become smaller and smaller. This agrees with the Johnson-Lindenstrauss lemma [9] that states that the number of dimensions needed to embed a high dimensional Euclidean space into one of much lower dimension is logarithmic in the number of points.

4.4 Storage and Processing overheads

TopSig is efficient and compares well with the inverted file approach. On a standard PC, a 1024 bit signature index can be searched by brute force in about 175 milliseconds, with a collection of 2.7 million signatures of the English Wikipedia documents. The signatures file size for this collection is only 325 MB, less than 0.65% of the collection size, and so it easily
fits in memory. By contrast, the highly compressed inverted file of ATIRE that underlies all the other models, occupies 1.5GB, or about 3% of the uncompressed text collection size. ATIRE itself is highly efficient and for comparison, the Indri index for the same collection occupies about 11% of the space.

Searching with TopSig is also efficient. We have not implemented a parallel multi-processor search which offers linear speedup in the number of CPUs. Even so, all 68 queries for INEX collection were completed in 12 seconds for the Wikipedia collection and all 50 WSJ topics were completed in 4 seconds, on a basic Laptop. This is comparable to the performance that is obtained with the inverted file system.

There is potential to further compress the index by sorting the binary strings lexicographically. Huffman coding can be used after sorting to represent the differences between successive document signatures. This approach has been shown to reduce a similar index used for near duplicate detection by up to 50% when used with 64-bit codes [13]. Thorough testing of this style of approach is beyond the scope of this paper and is expected to be investigated in further research. It is also possible that document clustering can provide effective ways to further compress the index. Document signatures in a cluster fall within a small Hamming distance of each other. Therefore, only a few bits differ between the cluster representative and document signatures it represents.

5. CLUSTERING EVALUATION

The goal of clustering is to place documents into topical groups. To achieve this, clustering algorithms compare similarity between entire document vectors. Therefore, the space and time efficiency of the TopSig representation allows it to outperform current approaches using sparse vector representations. It is also competitive in terms of document cluster quality. We have modified the k-means algorithm to work with signatures. This approach is compared to the implementation of k-means in the CLUTO clustering toolkit [10] that is popular in the IR community. CLUTO uses full precision sparse vectors to represent documents.

The same approach is used to create document signatures as for ad-hoc retrieval as described in Section 3. The sparsity of the signatures does not have a large impact on the cluster quality but we found that index vectors with 1 in 6 bits set performed best. Index vectors with 1 in 3 and 1 in 12 bits set were also tested.

The k-means algorithm [11] was modified to work with the bit string representation of TopSig. Cluster centroids and documents are N bit strings. Each bit in a centroid is the median for all documents it represents. If more than half the documents contain a bit set to 1 then the centroid contains this value in the corresponding position. As the 1 and 0 values represent +1 and -1, this is equivalent to adding all the vectors and taking the sign of each position. The standard Hamming distance measure is used to compare all vectors. The algorithm is initialized by selecting k random documents as centroids. This modified version of k-means always converged when the maximum number of iterations was not limited. Whether this modified version has the same convergence guarantees as the original algorithm is unknown.

An implementation of the k-means clustering algorithm using bit-vectors is available from the K-tree project subversion repository. Note that this is an unoptimised Java implementation. It is expected further performance increases can be gained by implementation in a lower level language such as C.

5.1 Results

We have evaluated document clustering using the INEX 2010 XML Mining collection [5]. It is a 144,265 document subset of the INEX XML Wikipedia collection. Clusters are evaluated using two approaches. The standard approach of comparing clusters to a “ground truth” set of categories is measured via Micro Purity. Purity is the proportion of a cluster that is the majority category label. The final score is Micro averaged where the Purity for each cluster is weighted by its size. On this collection, Purity produces approximately the same relationships between different clustering approaches as F1, Normalized Mutual Information and Entropy. There are 36 categories for documents that are extracted from the Wikipedia category graph.

An alternative evaluation is performed that has a specific application in information retrieval. Ad-hoc retrieval relevance judgments are used to measure the spread of relevant documents over clusters. This is motivated by the cluster hypothesis [20], stating that documents relevant to the same information need tend to cluster together. If this hypothesis holds then most of the results will be in a small number of clusters. The Normalized Cumulative Cluster Gain measure represents how relevant documents are spread over clusters. It falls in the range [0, 1] where a score of 1 indicates all relevant documents were contained in 1 cluster and a score of 0 indicates all relevant documents were evenly spread across all clusters. Complete details of the evaluation are available in a track overview paper from INEX 2010 [5].

The sparse document vectors used to create the TopSig document signatures are used as input to the k-means implementation in CLUTO. Therefore, we are comparing the same algorithm on the same data except for the fact the TopSig representation is extremely compressed and has a different centroid representation and distance measure. Both implementations of k-means are initialized randomly and are allowed to run for a maximum of 10 iterations. 36, 100, 200, 500 and 1000 clusters were produced by each approach where 36 was chosen to match the number of categories. This allows the trend of the measures to be visualised as the number of clusters are varied.

5.2 Analysis of Results

Figures 6 and 7 represent the quality of the clustering approaches using the Micro Purity and NCCG measures respectively. The TopSig representation nears the quality of CLUTO at 1024 bits and matches it at 4096 bits according to both measures. The best NCCG scores are all greater than 0.84 for all numbers of clusters, strongly supporting the cluster hypothesis, even when splitting the collection into 1,000 clusters.

Figure 8 shows how many times faster the TopSig clustering is than the traditional sparse vector approach in CLUTO. For example, using 4096 bit signatures to create 500 clusters is completed 20 times faster than CLUTO and 80 times faster at 1024 bits. This is one to two orders of magnitude increase.
increase in efficiency while still achieving the same quality as traditional approaches.

Figures 6, 7 and 8 can not be significance tested as they are a single run of the algorithms. However, the graphs allow the general trends to be visualised. CLUTO k-means takes approximately 5 hours to produce 1,000 clusters on this relatively small collection. Therefore, the CLUTO and TopSig k-means algorithms were repeatedly run to produce 36 clusters given different starting conditions. Given each random initialisation, k-means converges to a different local minima. The k-means implementations were run 20 times to measure this variability. Table 1 contains the results of this experiment where TopSig approaches that are equivalent to the CLUTO approach are highlighted in boldface. Equivalence was tested using the $t$-test with $p > 0.05$ indicating no statistically significant difference.

The time to produce the document signatures from the sparse document vectors was not included in the evaluation. The time is negligible in comparison to the time it takes to cluster using sparse document representations. Furthermore, when the k-means algorithm is limited in the number of iterations it can run for, it’s complexity becomes linear. The complexity of the document signature generation is also linear in the number of non-zero ($nnz$) elements in the term-by-document matrix. As, $O(nnz) + O(nnz) = O(nnz)$, the complexity of the clustering system is not changed by the introduction of the generation of the document signatures.

Table 1: Detailed Evaluation of 36 clusters

<table>
<thead>
<tr>
<th>Method</th>
<th>Micro Purity</th>
<th>NCCG</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLUTO</td>
<td>0.543 ± 0.008</td>
<td>0.955 ± 0.003</td>
</tr>
<tr>
<td>TopSig 4096</td>
<td>0.540 ± 0.008</td>
<td>0.951 ± 0.007</td>
</tr>
<tr>
<td>TopSig 3072</td>
<td>0.528 ± 0.009</td>
<td>0.939 ± 0.005</td>
</tr>
<tr>
<td>TopSig 2048</td>
<td>0.520 ± 0.007</td>
<td>0.926 ± 0.007</td>
</tr>
<tr>
<td>TopSig 1024</td>
<td>0.480 ± 0.007</td>
<td>0.867 ± 0.012</td>
</tr>
</tbody>
</table>
6. DISCUSSION

We have described TopSig, an approach to the construction of file signatures that emerges from aggressive compression of the conventional term-by-document weight matrix that underlies the most common and most successful inverted file approaches. When focusing on early precision, using P@10 measures from P95 up to P630, TopSig is shown to be as effective as even the best models available, while requiring equal or less amounts of space for storing signatures. Significant reductions in signature index size can be achieved with TopSig as a trade-off, reducing the signature file size by orders of magnitude, while accepting reduced early precision. Remarkably, even a single double precision variable – a 64 bit signature – is found to achieve 10% P@10 over 2.7 million documents Wikipedia collection. Testing with standard clustering benchmark tasks demonstrates TopSig to be equally effective and as accurate as a state-of-the-art clustering solution such as CLUTO, with processing speedup of one to two orders of magnitude.

TopSig had been applied to documents of greatly varying lengths. Both the WSJ and the Wikipedia collections have very short to very long documents, varying in size by up to five orders of magnitude. It had been suggested that file signatures are susceptible to this situation because of the increased probability of collisions on terms, but TopSig still performs well on these collections. In particular, we have tested TopSig with WSJ – the same collection that was used by Zobel et al to demonstrate the superiority of conventional inverted files. TopSig clearly outperforms conventional file signatures that were previously discredited. In this paper we compare TopSig directly with inverted file approaches to demonstrate similar performance levels.

Unlike early approaches to searching with file signatures, TopSig does not necessitate the complicated and tedious removal of false matches, and supports ranked retrieval in a straightforward manner. All the performance evaluation results that are reported in this paper were performed without any attention being paid to false matches. Not only is TopSig producing comparable results, but with respect to false matches it is also virtually indistinguishable from a user perspective because false matches do not occur unless using far too aggressive compression is applied, for instance, compressing documents into 64 bit signatures.

There are certain differences between TopSig and inverted file based retrieval which may offer advantages in some application settings. TopSig performs the search in constant time and independently of query length. Comparing full documents to the collection in a filtering task, or processing long queries, take exactly the same time as comparing a single term query. This may be useful in applications where predictability and quality of service guarantees are critical. Shortening the signature length can reduce the index size, with smooth degradation in retrieval performance. Signatures may offer significant advantages where storage space is at a premium and a robust trade-off is sought.

Distributed search is an attractive setting for TopSig – distributed indexing and retrieval have to resolve the problems of collection splitting and result fusion. With TopSig these operations are trivial to implement since the Hamming Distance between signatures can be used as a universal metric across the system. Gathering of global statistics can be ignored by using the raw term frequencies from each document. This further simplifies use of TopSig in a distributed setting and the trade-off with quality may be acceptable depending on the particular use of the system. If each text object in an enterprise carries its own signature – perhaps generated independently as a matter of routine by the applications that maintain the objects – then crawling and indexing the enterprise collection is a simple as collecting the signatures. Alternatively, TopSig can support the implementation of massively parallel search simply by distributing the query signature to every participating sub-system that maintains its own set of signatures. It is also trivial to implement distributed filtering with TopSig by maintaining a “watch list” of signatures that can be compared with incoming text objects at run time. TopSig is trivial to distribute on multi-processor platforms for the very same reasons. The simplicity of the search process means that with shared memory processor architecture a linear speedup in the number of concurrent hardware threads available can be achieved.

TopSig is particularly efficient in indexing. It places virtually no memory requirements during indexing, processing an entire collection in a single pass (assuming term statistics are stable, which they are in very large collections). The most significant remaining drawback to TopSig is that it still requires a comparison with all signatures in the collection. Parallel processing offers a simple solution, but it is not entirely satisfactory. Parallel search does not reduce the amount of computation that is required, it only distributes it. There are many reports in the research literature about more efficient approaches to signature file searching, which operate in sub-linear time. Many tree based approaches have been described, and some solutions offer improvements. It is not a solved problem by any means it is the subject of ongoing research with TopSig too.

This paper introduces TopSig, a new file signature approach that represents a viable alternative to conventional search engines. Our results demonstrate that with a different approach to signature construction and searching file signatures performance is comparable to that of conventional language and probabilistic models at early precision. TopSig represents a principled approach to the construction of file signatures, placing it in the same conceptual framework as other models. This is very different from the conventional ad-hoc formulation of file signatures. Future work with TopSig will address multi-processor implementation, a tree structured approach to the search process, and evaluation in a massively parallel massively distributed setting. Early findings of experiments with longer documents indicate that even improved performance can be achieved with TopSig by splitting documents. This is the subject of ongoing research.

7. REFERENCES


Chapter 13

Document Clustering Evaluation: Divergence from a Random Baseline

I proposed and implemented all of the contributions in this paper. The co-authors provided valuable feedback on arrangement and style. I wrote the manuscript, wrote the software[^1], devised the experimental design, conducted experiments, and performed the data analysis.

13.1 Evaluation

This paper introduces an approach to the correction of ineffective clusterings called Divergence from a Random Baseline is introduced. It produces a baseline that looks identical to the clustering except that the documents are allocated to clusters uniformly randomly. The cluster size distribution of the baseline matches that of the clustering being evaluated. This is able to differentiate problematic clusterings using the NCCG evaluation measure. It also provides an optimum to distortion as measured by RMSE that is not apparent otherwise.

[^1]: http://mloss.org/software/view/468/
Document Clustering Evaluation: Divergence from a Random Baseline

Christopher M. De Vries ¹ and Shlomo Geva ¹ and Andrew Trotman ²

School of Electrical Engineering and Computer Science
Queensland University of Technology, Brisbane, Australia ¹

Department of Computer Science, University of Otago, Dunedin, New Zealand ²

chris@de-vries.id.au  s.geva@qut.edu.au  andrew@cs.otago.ac.nz

Abstract

Divergence from a random baseline is a technique for the evaluation of document clustering. It ensures cluster quality measures are performing work that prevents ineffective clusterings from giving high scores to clusterings that provide no useful result. These concepts are defined and analyzed using intrinsic and extrinsic approaches to the evaluation of document cluster quality. This includes the classical clusters to categories approach and a novel approach that uses ad hoc information retrieval. The divergence from a random baseline approach is able to differentiate ineffective clusterings encountered in the INEX XML Mining track. It also appears to perform a normalisation similar to the Normalised Mutual Information (NMI) measure but it can be applied to any measure of cluster quality. When it is applied to the intrinsic measure of distortion as measured by RMSE, subtraction from a random baseline provides a clear optimum that is not apparent otherwise. This approach can be applied to any clustering evaluation. This paper describes its use in the context of document clustering evaluation.

1 Introduction

This paper extends, motivates, and analyses a document clustering evaluation approach that compensates for ineffective document clusterings during evaluation. An ineffective clustering is one that achieves a high score according to a measure of document cluster quality but provides no value as a clustering solution. Divergence from a random baseline is introduced and formally defined to address ineffective clusterings in evaluation. A notion of work performed by a clustering is introduced where ineffective cases appear to perform no useful learning. The paper is concluded with a detailed analysis of the results from the INEX 2010 XML Mining track. This paper clearly defines and motivates this approach with theoretical and experimental analysis.

Ineffective document clusterings have been investigated using two extrinsic evaluations. The first is the standard clusters to categories approach where document clusters are compared to a ground truth set of category labels. The second approach evaluates document clustering using ad hoc information retrieval that has a use case for collection selection where a document collection is distributed across many machines. A broker needs to direct a search query to machines containing relevant documents. If the documents are allocated to machines by document cluster, it is expected that only a few topical clusters need to be searched. This is motivated by the cluster hypothesis [20] that states relevant documents tend to be more similar to each other than non-relevant documents. The Normalised Cumulative Cluster Gain (NCCG) measure evaluates document clustering with respect to this use case for ad hoc information retrieval.

The paper proceeds as follows. Section 2 introduces the collaborative XML document mining evaluation forum at INEX. Section 3 introduces document clustering in an information retrieval context and discusses different approaches. Evaluation of document clustering using the clusters to categories approach and ad hoc relevance judgments is discussed in Section 4. Sections 5, 6, and 7 introduce and define ineffective clusterings that perform no useful learning and can be adjusted for by applying divergence from a random baseline. Section 8 analyses the application of divergence from a random baseline using the INEX 2010 XML mining track. The paper is concluded in Section 9.

2 INEX XML Mining Track

The XML document mining track was run for six years at INEX, the Initiative for the Evaluation of XML Information Retrieval ¹ [12; 10; 11; 26; 8]. It explored the emerging field of classification and clustering of semi-structured documents.

Document clustering has been evaluated at INEX using the standard clusters to categories approach, where categories extracted from the Wikipedia were used as a ground truth. Clusterings produced by different systems were evaluated using measures such as Purity, Entropy, F1 and NMI, indicating how well the clusters match the categories.

A novel approach to document clustering evaluation was introduced at INEX in 2009 [26] and 2010 [8]. It used ad hoc information retrieval to evaluate document clustering by using relevance judgments from retrieval systems in the ad hoc track [34]. Ad hoc information retrieval evaluation is a system-based approach that evaluates how different systems rank relevant documents. For systems to be compared, the same set of information needs and documents have to be used. A test collection consists of documents, statements of information need, and relevance judgments [36]. Relevance judgments are often binary and any document is considered relevant if any of its contents can contribute to the satisfaction of the specified information need.

¹http://inex.otago.ac.nz/tracks/wiki-mine/wiki-mine.asp
However, the ad hoc track at INEX provides additional relevance information where assessors highlight the relevant text in the documents. Information needs are also referred to as topics and contain a textual description of the information need, including guidelines as to what may or may not be considered relevant. Typically, only the keyword based query of a topic is given to a retrieval system.

The ad hoc information retrieval based evaluation of document clustering is motivated by the cluster hypothesis that suggests relevant documents are more similar to each other than non-relevant documents; relevant documents tend to cluster together. The spread of relevant documents over a clustering solution was measured using the Normalised Cumulative Cluster Gain (NCCG) measure in the INEX XML mining track in 2009 and 2010 [26; 8]. This evaluation approach also has a specific use case in information retrieval. It evaluates clustering of a document collection for collection selection. Collection selection involves selecting a subset of a collection given a query. Typically, these subsets are distributed on different machines. The goal is to cluster documents such that only a small fraction of clusters, and therefore machines, need to be searched to find most of the relevant documents for a given query. This leads to improved run time performance as only a fraction of the collection needs to be searched. The total load over a distributed system is decreased as only a few machines need to be searched per query instead of every machine. It also provides a clear use case for document clustering evaluation. By contrast, comparing document clusters to predefined categories only evaluates clustering as a match against a particular classification.

This paper uses the INEX 2010 XML Mining track dataset [8]. It is a 146,225 document subset of the INEX XML Wikipedia collection determined by the reference run used for the ad hoc track [2]. The reference run contains the 1500 highest ranked documents for each of the queries in the ad hoc track. The queries were searched using an implementation of Okapi BM25 in the ATIRE [35] search engine.

Topical categories for documents are one of many views of extrinsic cluster quality. They are derived from what humans perceive as topics in a document collection. When categories are used for evaluation, a document clustering system is given a score indicating how well the clusters match the predefined categories. This is the most prevalent approach to evaluation of document clustering in the research literature.

The categories for the INEX 2010 XML Mining collection were extracted from the Wikipedia category graph which is noisy and nonsensical at times. Therefore, an approach using shortest paths in the graph was used to extract 36 categories [8].

3 Document Clustering

Document clustering is used in many different contexts, such as exploration of structure in a document collection for knowledge discovery [33], dimensionality reduction for other tasks such as classification [22], clustering of search results for an alternative presentation to the ranked list [19] and pseudo-relevance feedback in retrieval systems [23].

Recently there has been a trend towards exploiting semi-structured documents [27; 11]. This uses features such as the XML tree structure and hyper-link graphs to derive data from documents to improve the quality of clustering.

Document clustering groups documents into topics without any knowledge of the category structure that exists in a document collection. All semantic information is derived from the documents themselves. It is often referred to as unsupervised clustering. In contrast, document classification is concerned with the allocation of documents to predefined categories where there are labeled examples to learn from. Clustering for classification is referred to as supervised learning where a classifier is learned from labeled examples and used to predict the classes of unseen documents.

The goal of clustering is to find structure in data to form groups. As a result, there are many different models, learning algorithms, encoding of documents and similarity measures. Many of these choices lead to different induction principles [14] which result in discovery of different clusters. An induction principle is an intuitive notion as to what constitutes groups in data. For example, algorithms such as k-means [24] and Expectation Maximisation [9] use a representative based approach to clustering where a prototype is found for each cluster. These prototypes are referred to as means, centers, centroids, medians and medoids [14]. A similarity measure is used to compare the representatives to examples being clustered. These choices determine the clusters discovered by a particular approach.

A popular model for learning with documents is the Vector Space Model (VSM) [30]. Each dimension in the vector space is associated with one term in the collection. Term frequency statistics are collected by parsing the document collection and counting how many times each term appears in each document. This is supported by the distributional hypothesis [18] from linguistics that theorises that words that occur in the same context tend to have similar meanings. If two documents use a similar vocabulary and have similar term frequency statistics then they are likely to be topically related. The end result is a high dimensional, sparse document-by-term matrix who’s properties can be explained by Zipf distributions [41] in term occurrence. The matrix represents a document collection where each row is a document and each column is a term in the vocabulary. In the clustering process, document vectors are often compared using the cosine similarity measure. The cosine similarity measure has two properties that make it useful for comparing documents. Document vectors are normalised to unit length when they are compared. This normalisation is important since it accounts for the higher term frequencies that are expected in longer documents. The inner product that is used in computing the cosine similarity has non-zero contributions only from words that occur in both documents. Furthermore, sparse document representation allows for efficient computation.

Different approaches exist to weight the term frequency statistics contained in the document-by-term matrix. The goal of this weighting is to take into account the relative importance of different terms, and thereby facilitate improved performance in common tasks such as classification, clustering and ad hoc retrieval. Two popular approaches are TF-IDF [29] and BM25 [28; 38].

Clustering algorithms can be characterized by two properties. The first determines if cluster membership is discrete. Hard clustering algorithms only assign each document to one cluster. Soft clustering algorithms assign documents to one or more clusters in varying degree of membership. The second determines the structure of the clusters found as being either flat or hierarchical. Flat clustering
algorithms produce a fixed number of clusters with no relationships between the clusters. Hierarchical approaches produce a tree of clusters, starting with the broadest level clusters at the root and the narrowest at the leaves.

K-means [24] is one of the most popular learning algorithms for use with document clustering and other clustering problems. It has been reported as one of the top 10 algorithms in data mining [39]. Despite research into many other clustering algorithms it is often the primary choice for practitioners due to its simplicity [17] and quick convergence [1]. Other hierarchical clustering approaches such as repeated bisecting k-means [32], K-tree [7] and agglomerative hierarchical clustering [32] have also been used. Further methods such as graph partitioning algorithms [21], matrix factorisation [40], topic modeling [5] and Gaussian mixture models [9] have also been used.

The k-means algorithm [24] uses the vector space model by iteratively optimising \( k \) centroid vectors which represent clusters. These clusters are updated by taking the mean of the nearest neighbours of the centroid. The algorithm proceeds to iteratively optimise the sum of squared distances between the centroids and the set of vectors that they are nearest neighbours to (clusters). This is achieved by iteratively updating the centroids to the cluster means and reassigning nearest neighbours to form new clusters, until convergence. The centroids are initialized by selecting \( k \) vectors from the document collection uniformly at random. It is well known that k-means is a special case of Expectation Maximisation [9] with hard cluster membership and isotropic Gaussian distributions.

The k-means algorithm has been shown to converge in a finite amount of time [31] as each iteration of the algorithm visits a possible permutation without revisiting the same permutation twice, leading to a worst case analysis of exponential time. Arthur et. al. [1] have performed a smoothed analysis to explain the quick convergence of k-means theoretically. This is the same analysis that has been applied to the simplex algorithm, which has a \( n^2 \) worst case complexity but usually converges in linear time on real data. While there are point sets that can force k-means to visit every permutation, they rarely appear in practical data. Furthermore, most practitioners limit the number of iterations k-means can run for, which results in linear time complexity for the algorithm. While the original proof of convergence applies to k-means using squared Euclidean distance [31], newer results show that other similarity measures from the Bregman divergence class of measures can be used with the same complexity guarantees [3]. This includes similarity measures such as KL-divergence, log-likelihood, Mahalanobis distance and Itakura-Saito distance.

Ding and He [13] demonstrate the relationship between k-means and Principle Component Analysis. PCA is usually thought of as a matrix factorisation approach for dimensionality reduction where as k-means is considered a clustering algorithm. It is shown that PCA provides a solution to the relaxed k-means problem, thus formally creating a link between k-means and matrix factorisation methods.

4 Document Clustering Evaluation

Evaluating document clustering is a difficult task. Intrinsic or internal measures of quality such as distortion or log likelihood only indicate how well an algorithm optimised a particular representation. Intrinsic comparisons are inherently limited by the given representation and are not comparable between different representations. Extrinsic or external measures of quality compare a clustering to an external knowledge source such as a ground truth labeling of the collection or ad hoc relevance judgments. This allows comparison between different approaches. Extrinsic views of truth are created by humans and suffer from the tendency for humans to interpret document topics differently. Whether a document belongs to a particular topic or not can be subjective. To further complicate the problem there are many valid ways to cluster a document collection. It has been noted that clustering is ultimately in the eye of the beholder [14].

When comparing a cluster solution to a labeled ground truth, the standard measures of Purity, Entropy, NMI and F1 are often used to determine the quality of clusters with regard to the categories. Let \( \omega = \{w_1, w_2, \ldots, w_K\} \) be the set of clusters for the document collection \( D \) and \( \xi = \{c_1, c_2, \ldots, c_J\} \) be the set of categories. Each cluster and category is a subset of the document collection, \( \forall c \in \xi, \omega \in \omega : c, \omega \subset D \). Purity assigns a score based on the fraction of a cluster that is the majority category label,

\[
\arg\max_{c \in \xi} \frac{|c \cap w_k|}{|w_k|},
\]

in the interval \([0, 1]\) where 0 is absence of purity and 1 is total purity. Entropy defines a probability for each category and combines them to represent order within a cluster,

\[
-\frac{1}{\log J} \sum_{j=1}^{J} \frac{|c_j \cap w_k|}{|w_k|} \log \frac{|c_j \cap w_k|}{|w_k|},
\]

which falls in the interval \([0, 1]\) where 0 is total order and 1 is complete disorder. F1 identifies a true positive \((tp)\) as two documents of the same category in the same cluster, a true negative \((tn)\) as two documents of different categories in different clusters and a false negative \((fn)\) as two documents of the same category in different clusters where the score combines these classification judgements using the harmonic mean,

\[
\frac{2 \times tp}{2 \times tp + fn + fp}.
\]

The Purity, Entropy and F1 scores assign a score to each cluster which can be micro or macro averaged across all the clusters. The micro average weights each cluster by its size, giving each document in the collection equal importance in the final score. The macro average is simply the arithmetic mean, ignoring the size of the clusters. NMI makes a trade-off between the number of clusters and quality in an information theoretic sense. For a detailed explanation of these measures please consult Manning et. al. [25].

4.1 NCCG

The NCCG evaluation measure has been used for the evaluation of document clustering at INEX [26; 8]. It is motivated by van Rijsbergen’s cluster hypothesis [20]. If the hypothesis holds true, then relevant documents will appear in a small number of clusters. A document clustering solution can be evaluated by measuring the spread of relevant documents for the given set of queries.

NCCG is calculated using manual result assessments from ad hoc retrieval evaluation. Evaluations of ad hoc retrieval occur in forums such as INEX [2], CLEF [15] and TREC [6]. The manual query assessments are called the relevance judgments and have been used to evaluate ad hoc retrieval of documents. The process involves defining
a query based on the information need, a retrieval system returning results for the query and humans judging whether the results returned by a system are relevant to the information need.

The NCCG measure tests a clustering solution to determine the quality of clusters relative to the optimal collection selection. Collection selection involves splitting a collection into subsets and recommending which subsets need to be searched for a given query. This allows a retrieval system to search fewer documents, resulting in improved runtime performance over searching the entire collection. The NCCG measure has complete knowledge of which documents are relevant to queries and orders clusters in descending order by the number of relevant documents it contains. We call this measure an “oracle” because it has complete knowledge of relevant documents. A working retrieval system does not have this property, so this measure represents an upper bound on collection selection performance.

Better clustering solutions in this context will tend to group together relevant results for previously unseen ad hoc queries. Real ad hoc retrieval queries and their manual assessment results are utilised in this evaluation. This approach evaluates the clustering solutions relative to a very specific objective – clustering a large document collection in an optimal manner in order to satisfy queries while minimising the search space. The measure used for evaluating the collection selection is called Normalised Cumulative Cluster Gain (NCCG) [26].

The Cumulative Gain of a Cluster (CCG) is defined by the number of relevant documents in a cluster, \( CCG(c, t) = \sum_{i=1}^{n_t} Rel_i \). A sorted vector \( CG \) is created for a clustering solution, \( c \), and a topic, \( t \), where each element represents the CCG of a cluster. It is normalised by the ideal gain vector,

\[
\text{SplitScore}(t, c) = \frac{\text{cumsum}(CG)}{n_t^2},
\]

where \( n_t \) is total number of relevant documents for the topic, \( t \). The worst possible split places one relevant document in each cluster represented by the vector \( CG_1 \),

\[
\text{MinSplitScore}(t, c) = \frac{\text{cumsum}(CG_1)}{n_t^2}.
\]

NCCG is calculated using the previous functions,

\[
\text{NCCG}(t, c) = \frac{\text{SplitScore}(t, c) - \text{MinSplitScore}(t, c)}{1 - \text{MinSplitScore}(t, c)}.
\]

It is then averaged across all topics.

4.2 Single and Multi Label Evaluation

Both the clustering approaches and the ground truth can be single or multi label. Examples of algorithms that produce multi label clusterings are soft or fuzzy approaches such as fuzzy c-means [4], Latent Dirichlet Allocation [5] or Expectation Maximisation [9]. A ground truth is multi label if it allows more than one category label for each document. Any combination of single or multi label clusterings or ground truths are able to be used for evaluation.

However, it is only reasonable to compare approaches using the same combination of single or multi label clustering and ground truths. Multi label approaches are less restrictive than single label approaches as documents can exist in more than one category. There is redundancy in the data whether it is clustering or a ground truth. This redundancy has a real and physical costs when clustering is used for collection selection. More storage and compute resources are required with a multi label clustering as one document has to be stored and processed on more than one computer. A ground truth can be considered a clustering and compared to another ground truth to measure how well the ground truths fit each other. Furthermore, a ground truth can be used as a clustering solution and used for collection selection.

The evaluation of document clustering using ad hoc information retrieval can be viewed as being similar to an evaluation using a multi label category based ground truth. A document can be relevant to more than one query. However, unlike a category based approach, each query is evaluated separately and then averaged across all queries. In contrast, all categories are evaluated at once and the score is not averaged across categories.

5 Ineffective Clustering

In this paper we introduce the concept of an ineffective clustering. An ineffective clustering produces a high score according to an evaluation measure but does not represent any inherent value as a clustering solution.

The Purity evaluation measure has an obvious ineffective case. If each cluster contains one document then it is 100% pure with respect to the ground truth. A single document is the majority of the cluster. As the goal of clustering is to produce groups of documents or to summarise the collection, this is obviously flawed as it does neither. The same applies to the Entropy measure as the probability of a label for a cluster is 100%, resulting in the highest possible Entropy score.

The NCCG measure is ineffective when one cluster contains all the documents except for every other cluster containing one document. The NCCG measure orders clusters by the number of relevant documents they contain. A large cluster containing most documents will almost always be ranked first. Therefore, almost all relevant documents will exist in one cluster, achieving almost the highest score possible.

6 Work Performed by a Clustering

To overcome ineffective clusterings in the previous section, we introduce the concept of work performed by a clustering approach. Work is defined as an increase in quality of a clustering over a simple approach that ignores the documents being clustered. A useful clustering performs work beyond an approach that is purely random and ignores document content. If a random approach that performs no useful learning performs equally to an approach that attempts to learn from that data, it would appear that nothing has been achieved by analysing the data. We suggest that an ineffective clustering performs no useful learning. This is supported by a theoretical and experimental analysis in the following sections.

Figures 1 and 2 illustrate an approach using a clustering algorithm and a random approach that ignores document content. The difference in cluster quality between these two approaches represents work completed by a clustering algorithm.
7 Divergence from a Random Baseline

Many measures of cluster quality can give high quality scores for particular clustering solutions that are not of high quality by changing the number of clusters or number of documents in each cluster.

Measures that can be misled by creating an ineffective clustering can be adjusted by subtraction from a randomly generated clustering with the same number of clusters with the same number of documents in each cluster. Figures 1 and 2 highlight this example where the random baseline is the same number of documents in each cluster. Apart from the random assignment of documents to clusters, the random baseline appears the same as the real solution. Therefore, each clustering evaluated requires a random baseline that is specific to that clustering. The baseline is created by shuffling the documents uniformly randomly and splitting them into clusters the same size as the clustering being measured. The score for the random baseline clustering is subtracted from the matching clustering being measured.

The divergence from a random baseline approach can be applied to any measure of cluster quality whether it is intrinsic or extrinsic. However, it does require an existing measure of cluster quality. It is not a measure by itself but an approach to ensure a clustering is doing something sensible. Although we have highlighted its use for document clustering evaluation, it can be used for any clustering evaluation.

There are two issues at play here. Firstly, different distributions of cluster sizes can lead to arbitrarily high scores. The second issue is determining if the clustering algorithm is effectively learning with respect to a measure of quality. The divergence from a random baseline takes care of ineffective solutions in either case. If the internal ordering of clusters is no better than random noise then it achieves a score of zero. A negative score could be achieved as the random baseline scores a positive value using most measures on most data sets. It is possible for a clustering to have a worse score than the baseline. For example, a clustering approach could maximise dissimilarity of documents in clusters. This will create a solution where the most dissimilar documents are placed together, resulting in a worse score than random assignment. The random assignment does not bias the clustering towards or away from the measure of quality. If a clustering approach is in fact learning something with respect to the measure of quality, then it is expected that is will be biased towards it. Alternatively, if we reverse the optimisation process, it should be biased away from it.

Let \( \omega = \{w_1, w_2, \ldots, w_K\} \) be the set of clusters for the document collection \( D \) and \( \xi = \{c_1, c_2, \ldots, c_J\} \) be the set of categories. Each cluster and category is a subset of the document collection, \( \forall c \in \omega, w \in \xi : c, w \subseteq D \). We define the probability of a category in the baseline given a cluster as, \( P_b(c_j|w_k) = \frac{|c_j \cap w_k|}{|w_k|} \). The probability of a category given a cluster in the baseline only depends on the size of the categories. The baseline is a uniformly randomly shuffled list of documents that has been split into clusters that match the cluster size distribution in the solution being evaluated. Thus, within each cluster in the baseline is random uniform noise. It is not biased by the document representation. So, it is expected categories will occur at a rate proportional to the category’s size. For example, if there are three categories \( A, B, C \) containing 10, 20, 30 documents, each cluster in the baseline is expected to contain approximately \( \frac{1}{3} \) \( A \), \( \frac{1}{2} \) \( B \), and \( \frac{1}{3} \) \( C \). This only reflects the size distribution of the categories.

We let any measure of a cluster quality be interpreted as a probability. Although this is not formally the case for all measures, it serves as a reasonable explanation. We define the probability of a category in a cluster given the ground truth as, \( P_a(c_j|w_k) = \frac{|c_j \cap w_k|}{|w_k|} \). This is the proportion of the cluster that has the majority category label. It also represents the same process of using clustering for classification with labelled data where an unseen sample is labelled based on the majority category label of the cluster it is nearest neighbour to. We define \( d \) as a document in \( D \). The ground truth is restricted to being single label where a document, \( d \), only has only one label in one category in the ground truth, \( \forall d \in D, c_i \in \xi, c_j \in \xi : d \in c_i \wedge d \notin c_j \wedge c_i \neq c_j \).

The adjusted measure is the difference between the submission and the baseline. We define the adjusted probability of a category given a cluster as, \( P_a(c_j|w_k) = P_a(c_j|w_k) - P_b(c_j|w_k) \).

An alternative formal view of divergence from a ran-
domain baseline can be defined by a quality function, \( m : \mathbb{PP}(Z \times Z) \rightarrow \mathbb{R} \), that takes a set of clusters as a set of 
set of (document, category label) pairs, \( s \), and returns a real number indicating the quality of the clustering. Examples of these cluster quality functions are Entropy, F1, NCCG, Negentropy, NMI and Purity. There exists a function, \( r : \mathbb{PP}(Z \times Z) \rightarrow \mathbb{PP}(Z \times Z) \), that generates a random baseline, \( b \), given a clustering solution, \( s \). The baseline has the same number of clusters as the clustering solution, \( |b| = |s| \). For every cluster in each of the original clustering, \( s \), and the baseline, \( b \), the corresponding clusters contain the same number of documents, \( \forall k : |s_k| = |b_k| \). The adjusted measure, \( m_a : \mathbb{PP}(Z \times Z) \rightarrow \mathbb{R} \), becomes, 
\[ m_a(s) = m(s) - m(r(s)). \]

8 Application at the INEX 2010 XML Mining Track

Participants were asked to submit multiple clustering solutions containing approximately 50, 100, 200, 500 and 1000 clusters. The categories extracted contained 36 categories due to only using categories with greater than 3000 documents. This choice was arbitrary and the decision for cluster sizes was made based on the number of documents in the collection before the categories were extracted. The number of categories in a document collection is subjective. Therefore, a direct comparison of 36 clusters with 36 categories is not necessary. Measuring how the categories behave over multiple cluster sizes indicates the quality of clusters and the trend can be visualised.

A legend for Figures 4 to 9 can be found in Figure 3. The Structured Linked Vector Model (SLVM) [37] incorporates document structure, links and content. The k-star [34] is an iterative clustering method for grouping documents. The TopSig approach [16] produces binary strings that represent documents and a modified k-means algorithm that works directly with this representation.

Submissions using the k-star method at INEX 2010 [34] contained several large clusters and many other small clusters. This exposed weakness in the NCCG measure, which resulted in inappropriately high scores. When the scores are subtracted from a random baseline with the same properties they performed no better than a randomly generated solution. This can be clearly seen in Figures 6 and 7 where the k-star method changes drastically between the original score and the score when subtracted from a random baseline.

The NMI measure is almost unaffected by subtraction from a random baseline where as other measures have a larger difference. Figures 8 and 9 highlight this property on submissions from INEX 2010. This suggests that the normalisation we have proposed is similar to that of NMI but is applicable to any measure of cluster quality whether it is intrinsic or extrinsic. Figures 4 and 5 demonstrate how the difference between the adjusted and unadjusted measures is larger for measures that are not normalised. Each line represents a different document clustering system. The bottom most line in each graph is a randomly generated clustering submission where a category for a document is selected uniformly at random from the set of categories. Note that this random clustering in the figures differs from the random baseline. The cluster size distribution is also uniform. A random baseline has a cluster size distribution that is specific to the clustering being evaluated. When compared to the random baseline the expected results are achieved, with a score of zero for all cluster sizes. Note that without adjusting the cluster size distribution, it is not able to differentiate ineffective clusterings as per the NCCG metric in Figure 7. Subtracting the random submission with uniform cluster sizes from the NCCG submission does not reduce its score to zero as can be seen in Figure 6.

Figures 10 and 11 demonstrate the application of the divergence from random baseline approach on an intrinsic measure. RMSE is the Root Mean Squared Error of the clustering using the cosine similarity measure. The higher value the better the clustering. A cosine similarity of 1 indicates the document and the cluster centre are identical. A score of 0 indicates they are orthogonal and therefore have no overlap in vocabulary. This experiment was run on a 10,000 document randomly selected sample. The k-means algorithm was used to produce \( k \) clusters between 1 and 10,000. Subtraction from a random baseline assigns a score of zero to these ineffective cases. Furthermore, it provides a clear maximum for RMSE.
9 Conclusion

In this paper we introduced problems encountered in evaluation of document clustering. This is the concept of ineffective clustering and a notion of work. The divergence from random baseline approach deals with these corner cases and increases the confidence that a clustering approach is achieving meaningful learning with respect to any view of cluster quality. It is also applicable to any clustering evaluation but was only discussed in the context of document clustering in this paper.
Divergence from a random baseline was formally defined and analysed experimentally with both intrinsic and extrinsic measures of cluster quality. Furthermore, this approach appears to be performing a normalisation similar to that performed by NMI. It also provides a clear optimum for distortion as measured by RMSE.

References


Chapter 14

Pairwise Similarity of TopSig Document Signatures

I contributed almost all contributions in this paper. Shlomo aided in the experimental design. I wrote the manuscript, wrote the software, aided in experimental design, conducted experiments, and performed the data analysis.

14.1 Representations

This paper investigates the topology of TopSig document signatures by analyzing the distributions of the pairwise distances between signatures. It highlights that the indexing process of TopSig is influencing the random codes used to perform dimensionality reduction. It skews the binomial distribution for uniform random binary strings such that the feature space is no longer uniform and is clustered. It is this non-uniformity that allows the differentiation of meaning. Furthermore, it highlights that the curse of dimensionality still applies for document signatures with most signatures being equidistant to each other around the middle of the distribution. Only the signatures in the left tail of the distribution allow adequate separation of documents; i.e. only the local neighborhood of the signatures are interpretable. This also gives a reasonable explanation of why the TopSig model is only competitive at early precision for ad hoc information retrieval when compared to language and probabilistic models. There is only adequate separation between documents in the head of the ranked list. When proceeding down the ranked list, documents become equidistant at higher recall.
ABSTRACT
This paper analyses the pairwise distances of signatures produced by the TopSig retrieval model on two document collections. The distribution of the distances are compared to purely random signatures. It explains why TopSig is only competitive with state of the art retrieval models at early precision. Only the local neighbourhood of the signatures is interpretable. We suggest this is a common property of vector space models.

Categories and Subject Descriptors
H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval—Retrieval Models

Keywords
Signature Files, Topology, Vector Space IR, Random Indexing, Document Signatures, Search Engines, Document Clustering, Near Duplicate Detection, Relevance Feedback

1. INTRODUCTION
This paper investigates the properties of the pairwise similarities of document signatures produced by TopSig. TopSig is a retrieval model where documents are represented by d-bit binary strings that lie on a d-dimensional collection hypercube. The signatures are produced by a random process called random indexing [10] or random projection [1] which compresses the standard term-by-document matrix.

Pairwise similarity plays an important role in many information retrieval related tasks such as ad hoc retrieval, clustering, classification, filtering, near duplicate detection and relevance feedback.

The paper proceeds as follows. In Section 2, the TopSig retrieval model is introduced. Section 3 describes the document collections used in the experiments. The experimental setup is introduced in Section 4 and the results are presented in Section 5. The paper is concluded by a discussion of the implications of the results in Section 6.

2. TOPSIG
TopSig [7] offers a radically different approach to the construction of file signatures. Traditional file signatures [6] have been shown to be inferior to approaches using inverted indexes, both in terms of the time and space required to process and store the index [12, 13]. However, TopSig overcomes previous criticisms aimed at file signatures by taking a principled approach using the vector space model, dimensionality reduction and numeric quantisation. Previous approaches to file signatures were constructed in an ad hoc fashion by combining random binary signatures using a bitwise XOR which is a Bloom filter [2] for the terms contained in documents. In contrast, TopSig randomly indexes a weighted term-by-document matrix and then quantises it. TopSig is competitive with state of the art probabilistic and language retrieval models at early precision, and clustering approaches [7].

Let \( D = \{d_1, d_2, ..., d_n\} \) be a document collection of \( n \) documents signatures, \( D \subset \{+1, -1\}^d \), \( |D| = n \). Let \( F = \{f_1, f_2, ..., f_n\} \) be the same document collection as \( D \) where each document is represented by a \( v \)-dimensional real valued vector, \( F \subset \mathbb{R}^v \), \( |F| = n \), where \( v \) is the size of the vocabulary of the document collection. \( F \) is the term-by-document matrix in the full space of the collection vocabulary which underlies most modern retrieval systems.

TopSig indexes documents using a mapping function, \( m : \mathbb{R}^v \rightarrow \{+1, -1\}^d \), that maps a document from the original \( v \)-dimensional continuous real valued term space, to a \( d \)-dimensional discrete binary valued space. The index is constructed using a mapping function, \( D = \{f \in F : m(f)\} \). The mapping function creates a sparse random ternary index vector of \( d \)-dimensions for each term in the document with \(+1\) and \(-1\) values in random positions and the majority of positions containing 0 values. These randomly generated codes are almost orthogonal to each other and have been shown to provide comparable quality to orthogonal approaches such as principle component analysis [1]. The index vector is multiplied by the term weight and added to a \( d \)-dimensional real valued vector that represents the document. Once all the terms in a document have been processed, this reduced dimensionality document vector is then quantised to a \( d \)-dimensional binary vector by thresholding each value in each dimension to 1 if greater than 0 and 0 otherwise. The 1 and 0 values in the binary vector represent \(+1\) and \(-1\) values. This mapping function can be applied to each document independently, meaning that new documents can be indexed in isolation without having to update the existing index. This is a key advantage to random indexing [10] over other dimensionality reduction techniques such as latent semantic analysis [5] which requires global analysis of the term-by-document matrix using the singular value decomposition [8].

The indexing process of TopSig is similar to that of SimHash [3]. However, TopSig uses signatures an order of magnitude longer than SimHash and it uses much sparser random
codes. The search process for ad hoc retrieval also differs, where TopSig searches in the subspace of the query and applies relevance feedback.

The binary vectors in $D$ provide a faithful representation of the original document vectors in $F$. The topological relationships in the original space are preserved in the reduced dimensionality space. This is supported by the Johnson-Lindenstrauss lemma [9] that states if points in a high-dimensional space are projected into a randomly chosen subspace, of sufficiently high-dimensionality, then the distances between the points are approximately preserved. It also states that the number of dimensions required to reproduce the topology is asymptotically logarithmic in the number of points.

3. DOCUMENT COLLECTIONS

We have used the INEX Wikipedia 2009 collection and the TREC Wall Street Journal (WSJ) Collection to evaluate pairwise distances of TopSig signatures. The INEX Wikipedia collection contains 2,666,190 documents with a vocabulary of 2,132,352 terms. We have used 2 subsets of this collection during evaluation. The first is a 144,265 document subset used for the INEX 2010 XML Mining track [4]. This is the reference run for the ad hoc track in 2010 produced by an implementation of Okapi BM25 in the ATIRE search engine [11]. It is denoted by INEX-reference. The second is a randomly selected 144,265 document subset chosen to match the size of the XML Mining subset. It is denoted by INEX-random. Subsets of the INEX Wikipedia 2009 collection were used for this experiment because calculating pairwise distances has a time complexity of $O(n^2)$ and becomes intractable for millions of documents. The mean document length in the Wikipedia has 360 terms, the shortest has 1 term and the longest has 38,740 terms. The Wall Street Journal Collection consists of 173,252 documents and a vocabulary of 113,288 terms. The mean WSJ document length is 475 terms, the shortest has 3 terms, and the longest has 12,811 terms.

The INEX Wikipedia 2009 collection consists of 12GB of uncompressed text or 50GB of uncompressed XML which includes semantic markup. The 2,666,190 documents are split into 3,617,380 passages. 1024-bit TopSig signatures use a total of 27MB to index the collection. It splits documents into 222,238 passages. 1024-bit TopSig signatures use an estimated probability for finding a signature at Hamming distance, $h$, is the fraction of similarities at that distance over the total number of distance comparisons. The probability mass function, $p_m f_{h} : \mathbb{P} \times \mathbb{N} \rightarrow \mathbb{R}$, produces the estimated probability from the pairwise distances in $D$ where $n$ is the number of signatures in the collection $|D| = n$.

$$p_m f_{h}(D, h) = \frac{\left\{ (d_x, d_y) : d_x, d_y \in D \land d_x \neq d_y \wedge s(d_x, d_y) = h \right\}}{\left(\frac{1}{2}\right)^{n-2}}.$$ (1)

Note that $p_m f_{h}$ is the estimated probability for finding a signature at distance, $h$, when averaged across all documents in the collection, $D$.

The probability of finding a random binary code of length, $d$, at Hamming distance, $h$, is described by the Binomial probability mass function, $p_m f : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{R}$,

$$p_m f_{h}(d, h) = \binom{d}{h} p^h (1-p)^{d-h},$$ (2)

where $p$ is the probability of a bit being set, $p = 0.5$.

The cumulative distribution function for either the estimated, $cdf_{h} : \mathbb{P} \times \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{R}$, or Binomial, $cdf_b : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{R}$, probability distributions are the sum of the probability mass function from 0 to $h$,

$$cdf_{h}(D, h) = \sum_{0}^{h} p_m f_{h}(D, h),$$ (3)

$$cdf_b(d, h) = \sum_{0}^{h} p_m f_{h}(d, h).$$ (4)

An implementation of the TopSig search engine was used to index the document collections. It splits documents into passages on a sentence boundary between a minimum and maximum number of word tokens. If the maximum word token limit is reached before the end of a sentence, it is split at that point. Therefore, documents have multiple signatures. This has been found to be effective for retrieval of

http://topsig.googlecode.com
documents of varying length. The INEX collection was split on a minimum of 256 and maximum of 280 word tokens. The WSJ collection was split on a minimum of 256 and a maximum of 384 word tokens. All indexes use 1024-bit signatures, resulting in the number of signatures as listed in Table 1.

The resulting probability distributions have been multiplied by the number of signatures in a collection to produce the expected number of signatures at a given Hamming distance. In this case, the pmf gives the average number of signatures expected at a particular Hamming distance when comparing a signature to the entire collection. The cdf gives the average number of signatures expected to lie within a given Hamming distance when comparing a signature to the entire collection, i.e. the number of nearest neighbours to expect within a particular Hamming distance.

5. EXPERIMENTAL RESULTS

Figures 1 to 12 highlight the difference between the distributions estimated from the pairwise distances and the distributions expected from random binary signatures from the Binomial distribution. It can be seen that all the estimated distributions are left skewed towards a Hamming distance of 0. This indicates that the signatures produced by TopSig are biased in such a way that documents are more similar to each other. There are more documents expected at a more similar, lower Hamming distance, than expected at random.

The probability mass functions in Figures 1, 2 and 3 represent the expected number of signatures to be seen at a particular Hamming distance. The graphs have been centred around the middle of the distributions to allow better visualisation of the separation between the distributions. The
tails of the distributions tend towards 0 as expected. For example, the graph in Figure 3 has a y value for the estimated distribution of 1033.39 at a Hamming distance of 441. When comparing a signature to the entire collection, it would be expected on average to encounter 1033.39 signatures that are exactly at a Hamming distance of 441. However, the expected number of signatures at a Hamming distance of 441 for purely random signatures is only 0.29. This suggests that the signatures produced by TopSig are not uniformly distributed throughout the feature space. The number of nearest neighbours at a given Hamming distance, as described by the pmf, quickly increases when starting from a Hamming distance of 0 and proceeding to a Hamming distance of \( d \). This is the same order that TopSig ranks signatures in the ranked list, or, any other task that compares relative orderings of documents such as clustering. This is true for both the estimated and Binomial distributions. As the neighbourhood of analysis is increased, more and more documents become equidistant; i.e. they share the same Hamming distance. This is a property of vector space models known as the “curse of dimensionality”. However, the left skewness of estimated distributions indicates that the pairwise distances of the document collections allow better differentiation between documents than expected at random. It is this left skewness of the distributions that allows TopSig to compete with state of the art retrieval models at early precision. Documents are topically clustered and are not random bags of words. Neither of the document collections have signatures further apart than a Hamming distance of 617, meaning that the indexing process has moved the random signatures from the right side of the distribution to the left. This again indicates that similar signatures are being placed closer together and are therefore more topically related and clustered.

The cumulative distribution functions in Figures 4, 5 and 6 represent the area under the curve for each of the probability mass functions. The y value at a given Hamming distance indicates the average number of nearest neighbours expected within a given Hamming distance when comparing a signature to the entire collection. For example, the graph in Figure 6 has a y value for the estimated distribution of 25975.78 at a Hamming distance of 441. When comparing a signature to the entire collection, it would be expected on
average to encounter 25975.78 signatures that are nearest neighbours at a Hamming distance of 441. However, the expected number of signatures at a Hamming distance of 441 for purely random signatures is only 1.13. Again, the separation between the curves indicates that TopSig is placing semantically related documents close together and preserving the topological relationships of the original document vectors.

Figures 7, 8 and 9 zoom in on the \( cdf \) where the first 100 nearest neighbours are expected for the distribution estimated from the pairwise distances of the collections. In all cases almost zero signatures are expected at random where there are TopSig signatures expected in the range \([1, 100]\). This indicates that the signatures produced by TopSig return nearest neighbours at a Hamming distance much earlier than expected by purely random signatures. The start and end points of these curves are listed in Table 2.

Figures 10, 11 and 12 zoom in on the \( cdf \) where the first 100 nearest neighbours are expected for random binary signatures as described by the Binomial distribution. The start

---

**Table 2: Nearest Neighbours Expected from \( cdf_b \)**

<table>
<thead>
<tr>
<th>Collection</th>
<th>( cdf_{b@cdf_b} = 1 )</th>
<th>( cdf_{b@cdf_b} = 100 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>INEX reference</td>
<td>( 1.66 \times 10^{-168} )</td>
<td>( 1.104 \times 10^{-15} )</td>
</tr>
<tr>
<td>INEX random</td>
<td>( 1.80 \times 10^{-158} )</td>
<td>( 2.06 \times 10^{-34} )</td>
</tr>
<tr>
<td>WSJ</td>
<td>0</td>
<td>( 1.59 \times 10^{-34} )</td>
</tr>
</tbody>
</table>

**Table 3: Nearest Neighbours Expected from \( cdf_e \)**

<table>
<thead>
<tr>
<th>Collection</th>
<th>( cdf_{e@cdf_e} = 1 )</th>
<th>( cdf_{e@cdf_e} = 100 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>INEX reference</td>
<td>567.86</td>
<td>2129.02</td>
</tr>
<tr>
<td>INEX random</td>
<td>1793.26</td>
<td>3673.78</td>
</tr>
<tr>
<td>WSJ</td>
<td>25495.67</td>
<td>50709.94</td>
</tr>
</tbody>
</table>
Table 4: Signatures Expected within $d^2$

<table>
<thead>
<tr>
<th>Collection</th>
<th>$cd_f, d^2=512$</th>
<th>$cd_f, d^2=512$</th>
</tr>
</thead>
<tbody>
<tr>
<td>INEX reference</td>
<td>0.51 0.61</td>
<td>0.51 0.63</td>
</tr>
<tr>
<td>INEX random</td>
<td>0.51 0.63</td>
<td>0.51 0.89</td>
</tr>
<tr>
<td>WSJ</td>
<td>0.51 0.89</td>
<td>0.51 0.89</td>
</tr>
</tbody>
</table>

and end points of these curves are listed in Table 3. There are many more signatures expected to be nearest neighbours when using TopSig signatures. However, both the estimated and Binomial distributions have many equidistant documents around the middle of their distributions. This suggests that only the local neighbourhood of the signatures has semantic meaning. This can also be seen in the pmf distributions where most of the signatures exist around the middle of the distribution. Another perspective is that there are too many ties at these distances for the feature space to differentiate signatures.

The skewness of the estimated distributions suggests that the feature space is not uniform and is clustered. Some areas of the space are more dense than others. This is vital for any document representation because it is this non-uniformity that allows differentiation of meaning.

Table 3 lists the number of nearest neighbours expected from the distribution estimated from pairwise distances when the Binomial distribution expects 1 and 100 nearest neighbours, as listed in columns 2 and 3 respectively. For example, the INEX random collection expects on average 1793.26 signatures to be nearest neighbours to other signatures when purely random signatures would expect 1. When purely random signatures expect on average 100 nearest neighbours, the INEX random collection expects 3673.78 nearest neighbours. These values are linearly interpolated as they exist in between two Hamming distances under the cdf. These values are the start and end points for the curves in Figures 10, 11 and 12. Table 2 lists the opposite, i.e., the number of nearest neighbours expected from the Binomial distribution when the distribution estimated from pairwise distances expects at 1 and 100 nearest neighbours.

Table 4 lists the number of signatures expected within a Hamming distance of $d$. This summarises the distributions in a single number, where the difference between the distributions indicates the fraction of the signatures shifted from the left hand side of the Binomial distribution to the right by the indexing process. It is also the value under the pmf at $d$ which is also the y value of the cdf at $d$.

The difference in distributions between the INEX reference and random subsets indicates that the reference run is not suitable for estimating properties of the entire collection. This is to be expected as the reference run has been biased by the queries used for ad hoc retrieval. Table 3 shows that INEX reference expects 567.86 nearest neighbours where as INEX random expects 1793.26 nearest neighbours when purely random signatures expect 1 nearest neighbour. This indicates that the reference run is less clustered than than a random sample from the INEX Wikipedia collection. This can explained because the documents returned by the reference run are more diverse than a random sample from the collection. As the diverse topics are further apart, i.e. more dissimilar, there are more inter-topic distances than
6. DISCUSSION

The results presented indicate why TopSig is only competitive at early precision in comparison to probabilistic and language models for ad hoc retrieval. As the Hamming distance increases when proceeding down the ranked list more and more documents become equidistant. This can be seen in Figures 1, 2 and 3 containing plots of probability mass functions indicating the expected number of documents at a given Hamming distance. The curves quickly increase to the point where thousands of documents are equidistant. This is likely to be a property of any vector space model due to the “curse of dimensionality”. Only the tails of the distribution of distances are useful for differentiation of relevant and non-relevant documents.

Approaches to near duplicate detection such as SimHash [3] use short signatures that are 64-bits in length. This only allows the few nearest neighbours to be differentiated which is adequate for near duplicate detection. This can be explained by the probability mass functions in Figures 1, 2 and 3. The x axis for 64-bit signatures will only contain 65 positions for the Hamming distances 0 to 64. As the number of equidistant documents is a function of the x value, or, Hamming distance, many documents will appear equidistant much sooner than with signatures of 1024-bits in length. The same curve has to be squeezed into 65 positions instead of 1025 positions. A duplicate is expected to be very similar to other documents it is a duplicate of, so these short signatures will suffice. In contrast, TopSig uses much longer signatures that allow for better separation for tasks such as ad hoc retrieval and clustering.

Document clustering places similar documents into groups of topically related documents. The results presented in this paper suggest that only document clusters that exist within the local neighbourhood of a vector space are interpretable. As the documents within a cluster become more dissimilar, the grouping of these documents loses its meaning for the same reason precision at higher recall suffers in ad hoc retrieval, there are many equidistant documents that are unable to be differentiated from one another. This suggests that only a large number of smaller document clusters are meaningful. The maximum interpretable radius for a document cluster can be estimated heuristically from the distributions of estimated from the pairwise data. This heuristic is to stop at the point where the distribution starts to sharply increase. In Figure 1 this would be approximately a Hamming distance of 450, or, the point before the elbow in the left hand side of the distribution occurs.

Furthermore, TopSig is likely to be useful for increased computational efficiency of document-to-document comparisons. Examples of this include clustering, classification, filtering, relevance feedback, near duplicate detection and explicit semantic analysis. All of these tasks can exploit the left tails of the probability mass function distributions depicted in Figures 1, 2 and 3. In fact, TopSig has been shown to provide a 1 to 2 magnitude increase in processing speed for document clustering [7] over traditional sparse vector representations.

The analysis presented in this paper is expected to be useful for any vector space model. It would be expected that similar behaviour would be exhibited whether comparing entire documents in the full vocabulary space of the term-by-document matrix or comparing dimensionality reduced documents in a continuous space such as those produced by latent semantic analysis, principal component analysis or random indexing.

7. REFERENCES

Chapter 15

Distributed Information Retrieval: Collection Distribution, Selection and the Cluster Hypothesis for Evaluation of Document Clustering

I contributed almost all of this paper. Shlomo and Andrew provided much needed feedback and aided in experimental design. I wrote the manuscript, wrote software, aided in experimental design, conducted experiments, and performed the data analysis. I proposed, implemented and evaluated all of the contributions in this paper such as the TopSig K-tree and the cluster ranking process, CBM625.

15.1 Algorithms

TopSig K-tree is introduced to scale clustering to 50 million documents while producing approximately 140,000 clusters in 10 hours in a single thread of execution. To the best of my knowledge, clustering of a document collection this large into this many clusters has not been reported in the literature. A delayed update mechanism is introduced as the K-tree algorithm regularly updates means along the insertion path of vectors inserted into the tree. When using binary vectors the cost of updating a mean is high as all the bits in vectors associated
with the mean must be unpacked and repacked.

15.2 Information Retrieval

This paper combines all of the previous sections of representations, algorithms and evaluation to implement the ideas as a cluster-based search engine. The final result is that only 0.1% of the 50 million document ClueWeb 2009 Category B collection has to be searched. This is a 13 fold improvement over the previously best known result. It also addresses issues surrounding the evaluation of document clustering and demonstrates the use case presented in the evaluation at INEX.
Distributed Information Retrieval: Collection Distribution, Selection and the Cluster Hypothesis for Evaluation of Document Clustering

CHRISTOPHER M. DE VRIES, Queensland University of Technology
SHLOMO GEVA, Queensland University of Technology
ANDREW TROTMAN, University of Otago

This paper addresses the problem of using document clusters for ad hoc information retrieval in large scale distributed and parallel search engines. We present a novel approach to collection distribution and selection via the use of the TopSig K-tree for document clustering for collection distribution, and the CBM625 method for ranking clusters for collection selection. This approach allows a 13 fold decrease in the number of documents that have to be retrieved over the previous best known approach. We suggest that the major contributor to this improvement is the TopSig K-tree. It allows the creation of fine grained clusters of large scale document collections without resorting to sampling. The TopSig K-tree can be updated efficiently in a dynamic and online fashion. TopSig also allows the creation of document signatures in isolation without performing any global analysis. These properties are particularly useful for operational retrieval systems, where new documents must be regularly indexed. The whole world does not need to be stopped, and the indexes can be updated incrementally and efficiently. This research started with the evaluation of document clustering at the INEX XML Mining track in 2009. This paper also presents the use of ad hoc relevance judgments as an alternative approach to the evaluation of document clustering. Furthermore, this evaluation is motivated by a real use case that is demonstrated in this paper; the reduction in the number of documents required to be searched in a distributed retrieval system. These results further confirm the cluster hypothesis.

Categories and Subject Descriptors: H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval

General Terms: Algorithms, Design, Experimentation, Performance, Theory

ACM Reference Format:

This work is supported by the Queensland University of Technology Deputy Vice Chancellor's Scholarship, the Queensland University of Technology High Performance Computing Centre and the Queensland University of Technology Big Data Lab.

Author's addresses: C.M. De Vries and Shlomo Geva, School of Electrical Engineering and Computer Science, Queensland University of Technology, Brisbane, Australia; Andrew Trotman, Department of Computer Science, University of Otago, Dunedin, New Zealand.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.

© YYYY ACM 1046-8188/YYYY/01-ARTA $15.00
DOI: http://dx.doi.org/10.1145/0000000.0000000

1. INTRODUCTION

Category based document clustering evaluation does not have a specific use case. Clusters are compared to a known solution referred to as the ground truth. The ground truth is determined by humans who assign each document to one or more classes based on the topics they believe exist in a document. In this paper, we offer a novel and alternative evaluation using ad hoc relevance judgements from information retrieval evaluation forums. This approach is motivated by the cluster hypothesis, which states that documents relevant to information needs tend to be more similar to each other than non-relevant documents – the relevant documents tend to cluster together. Therefore, if the cluster hypothesis holds, relevant documents for a given query will tend to appear in relatively few document clusters. This evaluation tests document clustering for a specific use case. It evaluated how effective a cluster is at increasing the throughput of an information retrieval system by only searching a subset of a document collection.

The cluster hypothesis can be exploited for searching a fraction of a document collection while still covering and ranking most of the relevant documents. This use case is tested with several experiments and evaluation strategies, including tests with working retrieval systems and reusable information retrieval evaluation resources. The process of selecting clusters to search in an information retrieval system is referred to as collection selection. It is also discussed in the literature under the names of sharding or index partitioning. Each cluster is treated as a sub-collection so that each cluster can be distributed to a different machine. Each sub-collection or cluster is processed by a collection selection algorithm where clusters are ranked by their estimated relevance to a query. Only a small fraction of the most relevant clusters is searched. If the clusters hypothesis holds, then the most relevant clusters will contain most of the relevant documents. Whether these clusters are distributed over many machines or only exist on a single machine, they can be exploited so that only a fraction of a collection has to be searched. The clustering or partitioning of a collection does not strictly have to be produced by a clustering algorithm. It can be created by random partitioning or source based methods. The collection selection process simply takes a grouping of documents and recommends which groupings to search given a query. The collection selection process can be seen as a broker or recommender where it recommends the collections or clusters to search given a query.

The study of collection distribution and selection is particularly relevant to distributed information retrieval where many machines are cooperating to provide a search service. It is expected that distributing a collection via document clustering and search via collection selection can increase throughput. In this paper, we only consider the case of a cooperative distributed information retrieval system where full access to the indexes of each system is possible. This enables use of global statistics and any other approaches that require more than the use of the standard query interface. Uncooperative systems only allow use of the standard query interface. All necessary information about a system that is acting as part of the distributed system has to be gained via this interface. The collection selection approach has to build an appropriate model by querying the systems individually.

Document clustering is not typically thought of as an efficient and scalable process. Many popular clustering algorithms have a time complexity of $O(n^2)$ making them impractical for collections containing millions to billions of documents. However, we present document clustering approaches that can scale to large document collections and we demonstrate their effectiveness for use with collection selection so that only a fraction of a document collection has to be searched.
The material presented here builds upon and extends the evaluation of document clustering in the XML Mining track at the Initiative for the Evaluation of XML Information Retrieval (INEX) in 2009 [Nayak et al. 2010] and 2010 [De Vries et al. 2011]. It also builds upon the divergence from random baseline approach to the evaluation of document clustering published in 2012 [De Vries et al. 2012].

For a comprehensive review of clustering algorithms see Duda et al. [2001] and for a comprehensive review of information retrieval see Manning et al. [2008].

The paper proceeds as follows. Related publications are discussed in Section 2. Section 3 introduces the INEX XML Mining track where this line of research originated with a collaborative evaluation. Ad hoc information retrieval and its associated evaluation are briefly introduced in Section 6. The literature surrounding document clustering is discussed in Section 4. The cluster hypothesis that links document clustering and ad hoc information retrieval are discussed in Section 7. Section 6.1 discusses collection distribution and selection, and relevant publications in this area. The issues with current approaches to evaluation of document clustering are discussed in Section 4.1. It is explained how this evaluation overcomes these issues in Section 8. The Normalised Cumulative Cluster Gain evaluation approach is introduced in Section 9. It provides an upper bound on retrieval performance by using an “oracle” collection selection approach. Section 10 discusses the Divergence from Random Baseline approach to the evaluation of document clustering that adjusts for problematic clusterings in evaluation. The TopSig K-tree is presented for the first time in Section 11. A new cluster ranking approach for collection selection named CBM625 is defined in section 12. The prior theoretical results from INEX are tested on a cluster based retrieval system in Section 13. Finally, the implications of all the results are discussed in Section 14.

2. RELATED WORK

Other researchers have investigated splitting document collections into shards. The study that is most similar to this study is that of Kulkarni and Callan [2010]. It investigates the use of topics learned in an unsupervised manner to search a small fraction of a large scale document collection. Documents are clustered using k-means with the KL-divergence similarity measure using a small sample of uniformly randomly sampled documents. The remaining documents are then assigned to these centroids found using the sample. The drawback of this approach is out of vocabulary terms when ranking clusters. The approach outlined in this paper does not suffer from this drawback as it clusters all documents. Xu and Croft [1999] also perform a similar study finding similar results on smaller scale collections. All of these studies, including this study, have found the same result, that topic based allocation of documents via document clustering outperforms uniform random assignment of documents.

Other studies have investigated alternative collection distribution strategies. Larkey et al. [2000] investigate the use of manual topical classification of patent documents where it outperformed chronological allocation. Puppin et al. [2006] investigate the use of query logs to partition a document collection with the drawback that not all documents can be clustered.

Ding et al. [2011] describe a histogram approach for collection selection. This is similar to the approach we tried in Section 12. However, we found that squaring the weights to produce document scores to be more effective than a summation as in the histogram approach. The authors propose that random allocation and random selection of documents and servers are a good candidate due to its simplicity. Fine grained clusters such as those produced by TopSig K-tree in Section 11 represent something much closer to a document than much larger clusters such as those used by Kulkarni and Callan [2010]. Therefore, randomly allocating fine grained clusters can still have the advantages of this approach and topical clustering approaches.
Fuhr et al. [2011] introduce the “The optimum clustering framework”. It augments document clustering with queries and relevance judgements so that relevant documents can be optimally placed into clusters to maximise the cluster hypothesis. This is no longer unsupervised learning and has been transformed into semi-supervised learning. Therefore, this study is essentially different from studies of Kulkarni and Callan [2010], Xu and Croft [1999] and this study. However, we suggest that the optimum clustering framework may also be more effective when using scalable clustering approaches such as TopSig K-tree.

They take a different approach to dealing with the issues surrounding evaluation of clustering using a ground truth. They suggest that using multiple different ground truths as different views of clustering can overcome issues with clustering evaluation. However, this does not deal with the problem of labelling large data sets. It only increases assessor load as each example has to be labelled multiple times.

3. INEX XML MINING TRACK

The XML document mining track was run for six years at INEX, the Initiative for the Evaluation of XML Information Retrieval [Denoyer et al. 2007; Denoyer and Gallinari 2008; Denoyer and Gallinari 2009; Nayak et al. 2010; De Vries et al. 2011]. It explored the emerging field of classification and clustering of semi-structured documents.

Document clustering has been evaluated at INEX using the standard clusters to categories approach. Categories were extracted from the Wikipedia and used as a ground truth. Clusterings produced by different systems were evaluated using measures such as Purity, Entropy, F1 and NMI, indicating how well the clusters match the categories.

A novel approach to document clustering was introduced at INEX in 2009 [Nayak et al. 2010] and 2010 [De Vries et al. 2011]. It used ad hoc information retrieval to evaluate document clustering by using relevance judgments from retrieval systems in the ad hoc track [7]. Ad hoc information retrieval evaluation is a system based approach that evaluates how different systems rank relevant documents. For systems to be compared, the same set of information needs and documents have to be used. A test collection consists of documents, statements of information need, and relevance judgments [Voorhees 2002]. Relevance judgments are often binary, and any document is considered relevant if any of its contents can contribute to the satisfaction of the specified information need. However, the ad hoc track at INEX provides additional relevance information where assessors highlight the relevant text in the documents. Information needs are also referred to as topics and contain a textual description of the information need, including guidelines as to what may or may not be considered relevant. Typically, only the keyword based query of a topic is given to a retrieval system.

The ad hoc information retrieval based evaluation of document clustering is motivated by the cluster hypothesis that suggests relevant documents are more similar to each other than non-relevant documents; relevant documents tend to cluster together. The spread of relevant documents over a clustering solution was measured using the Normalised Cumulative Cluster Gain (NCCG) measure in the INEX XML mining track in 2009 and 2010 [Nayak et al. 2010; De Vries et al. 2011]. This evaluation approach also has a specific use case in information retrieval. It evaluates clustering of a document collection for collection selection. Collection selection involves selecting a subset of a collection given a query. Typically, these subsets are distributed on different machines. The goal is to cluster documents such that only a small fraction of clusters, and therefore machines need to be searched to find most of the relevant documents.
documents for a given query. This leads to improved run time performance as only a fraction of the collection needs to be searched. The total load over a distributed system is decreased as only a few machines need to be searched per query instead of every machine. It also provides a clear use case for document clustering evaluation. By contrast, comparing document clusters to predefined categories only evaluates clustering as a match against a particular classification.

Topical categories for documents are one of many views of extrinsic cluster quality. They are derived from what humans perceive as topics in a document collection. When categories are used for evaluation, a document clustering system is given a score indicating how well the clusters match the predefined categories. This is the most prevalent approach to evaluation of document clustering in the research literature.

The categories for the INEX 2010 XML Mining collection were extracted from the Wikipedia category graph which is noisy and nonsensical at times. Therefore, an approach using shortest paths in the graph was used to extract 36 categories [De Vries et al. 2011].

4. DOCUMENT CLUSTERING

Document clustering is used in many different contexts, such as exploration of structure in a document collection for knowledge discovery [Tan 1999], dimensionality reduction for other tasks such as classification [Kyriakopoulou and Kalamboukis 2007], clustering of search results for an alternative presentation to the ranked list [Hearst and Pedersen 1996] and pseudo-relevance feedback in retrieval systems [Lee et al. 2008].

Recently there has been a trend towards exploiting semi-structured documents [Nayak et al. 2002; Denoyer and Gallinari 2009]. This uses features such as the XML tree structure and hyper-link graphs to derive data from documents to improve the quality of clustering.

Document clustering groups documents into topics without any knowledge of the category structure that exists in a document collection. All semantic information is derived from the documents themselves. It is often referred to as unsupervised clustering. In contrast, document classification is concerned with the allocation of documents to predefined categories where there are labeled examples. Clustering for classification is referred to as supervised learning where a classifier is learned from labeled examples and used to predict the classes of unseen documents.

The goal of clustering is to find structure in data to form groups. As a result, there are many different models, learning algorithms, encoding of documents and similarity measures. Many of these choices lead to different induction principles [Estivill-Castro 2002] which result in discovery of different clusters. An induction principle is an intuitive notion as to what constitutes groups in data. For example, algorithms such as k-means [Lloyd 1982] and Expectation Maximization [Dempster et al. 1977] use a representative based approach to clustering where a prototype is found for each cluster. These prototypes are referred to as means, centers, centroids, medians and medoids [Estivill-Castro 2002]. A similarity measure is used to compare the representatives to examples being clustered. These choices determine the clusters discovered by a particular approach.

A popular model for learning with documents is the Vector Space Model (VSM) [Salton et al. 1975]. Each dimension in the vector space is associated with one term in the collection. Term frequency statistics are collected by parsing the document collection and counting how many times each term appears in each document. This is supported by the distributional hypothesis [Harris 1954] from linguistics that theorizes that words that occur in the same context tend to have similar meanings. If two documents use a similar vocabulary and have similar term frequency statistics,
then they are likely to be topically related. The result is a high dimensional, sparse document-by-term matrix with properties that can be explained by Zipf distributions [Zipf 1949] in term occurrence. The matrix represents a document collection where each row is a document and each column is a term in the vocabulary. In the clustering process, document vectors are often compared using the cosine similarity measure. The cosine similarity measure has two properties that make it useful for comparing documents. Document vectors are normalized to unit length when they are compared. This normalization is important since it accounts for the higher term frequencies that are expected in longer documents. The inner product that is used in computing the cosine similarity has non-zero contributions only from words that occur in both documents. Furthermore, sparse document representation allows for efficient computation.

Different approaches exist to weight the term frequency statistics contained in the document-by-term matrix. The goal of this weighting is to take into account the relative importance of different terms, and thereby facilitate improved performance in common tasks such as classification, clustering and ad hoc retrieval. Two popular approaches are TF-IDF [Salton and Buckley 1988] and BM25 [Robertson et al. 1995; Whissell and Clarke 2011].

Clustering algorithms can be characterized by two properties. The first determines if cluster membership is discrete. Hard clustering algorithms only assign each document to one cluster. Soft clustering algorithms assign documents to one or more clusters in varying degree of membership. The second determines the structure of the clusters found as being either flat or hierarchical. Flat clustering algorithms produce a fixed number of clusters with no relationships between the clusters. Hierarchical approaches produce a tree of clusters, starting with the broadest level clusters at the root and the narrowest at the leaves.

K-means [Lloyd 1982] is one of the most popular learning algorithms for use with document clustering and other clustering problems. It has been reported as one of the top 10 algorithms in data mining [Wu et al. 2008]. Despite research into many other clustering algorithms, it is often the primary choice for practitioners due to its simplicity [Guyon et al. 2009] and quick convergence [Arthur et al. 2009]. Other hierarchical clustering approaches such as repeated bisecting k-means [Steinbach et al. 2000], K-tree [De Vries and Geva 2009] and agglomerative hierarchical clustering [Steinbach et al. 2000] have also been used. Note that the K-tree paper is by the author of this paper and other related content can be found in the associated Masters thesis [De Vries 2010b]. Further methods such as graph partitioning algorithms [Karypis et al. 1999], matrix factorization [Xu et al. 2003], topic modeling [Blei et al. 2003] and Gaussian mixture models [Dempster et al. 1977] have also been used.

The k-means algorithm [Lloyd 1982] uses the vector space model by iteratively optimizing k centroid vectors, which represent clusters. These clusters are updated by taking the mean of the nearest neighbors of the centroid. The algorithm proceeds to iteratively optimize the sum of squared distances between the centroids and the nearest neighbor set of vectors (clusters). This is achieved by iteratively updating the centroids to the cluster means and reassigning nearest neighbors to form new clusters, until convergence. The centroids are initialized by selecting k vectors from the document collection uniformly at random. It is well known that k-means is a special case of Expectation Maximization with hard cluster membership and isotropic Gaussian distributions [Press 2007].

The k-means algorithm has been shown to converge in a finite amount of time [Selim and Ismail 1984] as each iteration of the algorithm visits a possible permutation without revisiting the same permutation twice, leading to the worst case analysis of exponential time. Arthur et al. [2009] have performed a smoothed analysis to explain the quick convergence of k-means theoretically. This is the same analysis that has
been applied to the simplex algorithm, which has an $n^2$ worst case complexity but usually converges in linear time on real data. While there are point sets that can force k-means to visit every permutation, they rarely appear in practical data. Furthermore, most practitioners limit the number of iterations k-means can run for, which results in linear time complexity for the algorithm. While the original proof of convergence applies to k-means using squared Euclidean distance [Selim and Ismail 1984], newer results show that other similarity measures from the Bregman divergence class of measures can be used with the same complexity guarantees [Banerjee et al. 2005]. This includes similarity measures such as KL-divergence, logistic loss, Mahalanobis distance and Itakura-Saito distance. Ding and He [2004] demonstrate the relationship between k-means and Principle Component Analysis. PCA is usually thought of as a matrix factorization approach for dimensionality reduction whereas k-means is considered a clustering algorithm. It is shown that PCA provides a solution to the relaxed k-means problem by ignoring some constants, thus formally creating a link between k-means and matrix factorization methods.

4.1. Evaluation

Evaluating document clustering is a difficult task. Intrinsic or internal measures of quality such as distortion [Pelleg et al. 2000] or log likelihood [Biernacki et al. 2000] only indicate how well an algorithm optimized a particular representation. Intrinsic comparisons are inherently limited by the given representation and are not comparable between different representations. Extrinsic or external measures of quality compare a clustering to an external knowledge source such as a ground truth labeling of the collection or ad hoc relevance judgments. This allows comparison between different approaches. Extrinsic views of truth are created by humans and suffer from the tendency for humans to interpret document topics differently. Whether a document belongs to a particular topic or not can be subjective. To further complicate the problem, there are many valid ways to cluster a document collection. It has been noted that clustering is ultimately in the eye of the beholder [Estivill-Castro 2002].

Most of the current literature on clustering evaluation utilizes the classes-to-clusters evaluation which assumes that the classes of the documents are known. Each document has known category labels. Any clustering of these documents can be evaluated with respect to this predefined classification. It is important to note that the class labels are not used in the process of clustering, but only for the purpose of evaluation of the clustering results.

4.1.1. Evaluation Metrics. When comparing a cluster solution to a labeled ground truth, the standard measures of Purity, Entropy, NMI and F1 are often used to determine the quality of clusters with regard to the categories. Let $\omega = \{ w_1, w_2, \ldots, w_K \}$ be the set of clusters for the document collection $D$ and $\xi = \{ c_1, c_2, \ldots, c_J \}$ be the set of categories. Each cluster and category are a subset of the document collection, $\forall c \in \xi, w \in \omega : c, w \subseteq D$. Purity assigns a score based on the fraction of a cluster that is the majority category label,

$$\arg\max_{c \in \xi} \frac{|c \cap w_k|}{|w_k|},$$

(1)

in the interval $[0, 1]$ where 0 is the absence of purity and 1 is total purity. Entropy defines a probability for each category and combines them to represent order within a clustering,

$$-\frac{1}{\log J} \sum_{j=1}^{J} \frac{|c_j \cap w_k|}{|w_k|} \log \frac{|c_j \cap w_k|}{|w_k|},$$

(2)
which falls in the interval $[0, 1]$ where 0 is total order and 1 is complete disorder. F1 identifies a true positive ($tp$) as two documents of the same category in the same cluster, a true negative ($tn$) as two documents of different categories in different clusters and a false negative ($fn$) as two documents of the same category in different clusters where the score combines these classification judgments using the harmonic mean,

$$\frac{2 \times tp}{2 \times tp + fn + fp}.$$  

(3)

The Purity, Entropy and F1 scores assign a score to each cluster which can be micro or macro averaged across all the clusters. The micro average weights each cluster by its size, giving each document in the collection equal importance, in the final score. The macro average is simply the arithmetic mean, ignoring the size of the clusters. NMI makes a trade-off between the number of clusters and quality in an information theoretic sense. For a detailed explanation of these measures please consult [Manning et al. 2008].

4.1.2. Single and Multi Label Evaluation. Both the clustering approaches and the ground truth can be single or multi label. Examples of algorithms that produce multi label clusterings are soft or fuzzy approaches such as fuzzy c-means [Bezdek et al. 1984], Latent Dirichlet Allocation [Blei et al. 2003] or Expectation Maximisation [Dempster et al. 1977]. A ground truth is multi label if it allows more than one category label for each document. Any combination of single or multi label clusterings or ground truths can be used for evaluation. However, it is only reasonable to compare approaches using the same combination of single or multi label clustering and ground truths. Multi label approaches are less restrictive than single label approaches as documents can exist in more than one category. There is redundancy in the data whether it is clustering or a ground truth. A ground truth can be viewed as a clustering and compared to another ground truth to measure how well the ground truths fit each other.

5. DOCUMENT REPRESENTATIONS

There are many different techniques that effect the generation of representations for clustering. By pre-processing documents, using new representations for different sources of information, and improving the representation of knowledge for learning, both the quality of clusters discovered and the run-time cost of producing clusters can be improved.

When representing natural language, different components of the language lead to different representations. Popular models include the bag-of-words approach [Steinbach et al. 2000], n-grams [Miao et al. 2005], part of speech tagging [Allan and Raghavan 2002], and, semantic spaces [Lund and Burgess 1996].

Before any term based representation can be built, what constitutes a term must be determined via a number of factors such as term tokenization, conversion of words to their stems [Porter 2006], removal of stop words [Fox 1989], and, conversion of characters such as removing accents. Additionally, automated techniques may remove irrelevant boilerplate content from documents such as navigation structure on a web page are used [Yi et al. 2003].


Links between documents can be represented for use in document clustering. There are algorithms that work directly with the graph structure in an iterative fashion to produce clusters [Van Dongen 2008]. Other approaches embed the graph structure in a
document representation such as vector space representations [Riesen et al. 2007; Luo et al. 2003]. The representations are then used with general clustering algorithms. The links can also be weighted by popular algorithms from ad hoc information retrieval such as PageRank [Page et al. 1999] and HITS [Kleinberg 1999].

Other internal structural information about documents can be represented implicitly or explicitly. One effective implicit approach is to double the weight of title words in a bag-of-words representation [Cohen and Singer 1999]. Other approaches explicitly represent structure, such as those that represent the XML structure of documents. For example, Kutty et al. [2011] represent both content and structure explicitly using a Tensor Space Model. Tagarelli and Greco [2006] combine structure and content together using tree mining and modified similarity measures.

There are many dimensionality reduction techniques aimed at reducing the number of dimensions required to represent a document in a vector space model. They all aim to preserve the information available in documents. However, some particular approaches such as Latent Semantic Analysis [Deerwester et al. 1990] based upon the Singular Value Decomposition have been suggested to improve document representations by finding “concepts” in text. Popular approaches include Principle Component Analysis [Bingham and Mannila 2001], Singular Value Decomposition [Golub and Kahan 1965], Non-negative Matrix Factorization [Xu et al. 2003], Wavelet transform [Murtagh et al. 2000], Discrete Cosine Transform [Fox 2005], Semantic Hashing [Salakhutdinov and Hinton 2009], Locality Sematnic Hashing [Indyk and Motwani 1998], Random Indexing [Sahlgren 2005], and, Random Manhattan Indexing [Qasemizadeh and Handschuh 2014].

With the rise of the social web, many web pages now have social annotation such as tags. These can be introduced into a document representation to further disambiguate the topic of a document [Ramage et al. 2009].

Various other multimedia sources of information can be represented in documents such as audio [Lu et al. 2002], video [Yeung et al. 1998] and images [Liew et al. 2000]. The processing of these signals constitutes many research fields to themselves. But there have been approaches incorporating this information into the document clustering process [Barnard and Forsyth 2001].

Documents can be mapped onto ontologies to provide a conceptual framework for the representation of meaning in documents. This has been used to improve the quality of document clusters [Hotho et al. 2003].

Additionally, incorporating information from other category systems such as the Wikipedia has shown to improve the quality of clusters [Hu et al. 2009].

There are many approaches trying to use a single additional approach to improve document clustering via improved representation. However, there seems to be few approaches combining multiple sources and comparing how they contribute to the identification of clusters.

6. AD HOC INFORMATION RETRIEVAL

Ad hoc information retrieval is concerned with the retrieval of documents given a user query. The phrase ad hoc is of Latin origin and literally means “for this”. Users create a query specific to their information need and retrieval systems return relevant documents. Much of modern IR research has been concerned with ranked retrieval of documents. Each document is assigned an estimate of relevance for a given query and the results are displayed in descending order of relevance.

Retrieval systems or search engines provide automated ad hoc retrieval. The systems are typically implemented in software, but there have been hardware based retrieval systems [Weeks et al. 2002]. The inverted file is the data structure that underpins most modern retrieval systems [Manning et al. 2008]. Like an index at the back of
A textbook, an inverted file maps words to instances where they appear. The inverted file maps each term in the collection vocabulary to a postings list that identifies documents containing the term and the frequency with which it occurs. Various ranking functions re-weight query and document scores using the inverted file to improve ad hoc retrieval effectiveness. Popular approaches include TF-IDF [Salton and Buckley 1988], Language Models [Zhai and Lafferty 2004; Macdonald and Ounis 2006], Divergence from Randomness [Amati and van Rijsbergen 2002] and Okapi BM25 [Robertson and Jones 1997]. However, signature file based approaches have been popular in the past [Faloutsos and Christodoulakis 1984] and have recently reappeared in the mainstream IR literature using random projections or random indexing [Geva and De Vries 2011], which is a contribution of this paper. Signature files use an n bit binary string to represent documents and queries which are compared using the Hamming distance. The motivation of this approach is to exploit efficient bit-wise instructions on modern CPUs.

The study of retrieval models most commonly as different similarity functions focuses on the quality aspects of information retrieval systems. This studies how can the quality of results with respect to human judgement of a search engine be improved. In contrast, indexing implementations focus on the efficiency and approximate indexing that make a trade-off between quality of search results and efficiency.

Document signatures have been missing from mainstream publications about search engines and the Information Retrieval field in general for many years. Zobel et al. clearly demonstrate the inferior performance of traditional document signatures in their paper, “Inverted Files Versus Signature Files for Text Indexing” [Zobel 1998]. They conclude that signature files or document signatures are slower, offer less functionality, and require more storage. Witten et al. [1999] come to the same conclusion. Traditional signature based indexes are not competitive with inverted files. They offer no distinct advantages over inverted indexes.

Traditional bit slice signature files [Faloutsos and Christodoulakis 1984] use efficient bit-wise operators. This is presented in an ad hoc manner and is motivated by efficient bit-wise processing without respect to Information Retrieval theory. Each document is allocated a signature of \( N \) bits. Each term is assigned a random signature where only \( n \ll N \) bits are set. These are combined using the bit-wise XOR operation. This is a Bloom filter [Bloom 1970] for the set of terms in a document. To avoid collisions resulting in errors, the difference between \( n \) and \( N \) must be very large. In contrast, the TopSig signatures [Geva and De Vries 2011] presented in this paper compress the same representation that underlies the inverted index, the document-by-term matrix. It first begins with full precision vectors and compresses them via random projections and numeric quantization. This is founded in the theory of preservation of the topological relationships of low dimensional embeddings of vectors presented in the Johnson-Lindenstrauss lemma [Johnson and Lindenstrauss 1984].

Other recent approaches to similarity search [Zhang et al. 2010] have focussed on mapping documents to \( N \) bit strings for comparison using the Hamming distance. However, they have only focussed on short strings for document-to-document search for tasks such as near duplicate detection or the “find other documents like these” queries. They do not investigate use of such structures for ad hoc information retrieval or clustering. Additionally they use machine learning approaches to find a better mapping for signatures, which may be computationally prohibitive for web scale document collections containing billions of documents.
6.1. Collection Distribution and Selection
A distributed information retrieval system consists of many different machines connected via a data communications network working together as one to provide search services for documents stored on each of the machines.

In a distributed retrieval setting, several different information resources can be stored in different retrieval systems and indexes. These resources are combined to provide a single query interface. This often results in an uncooperative distributed system. Uncooperative systems only allow use of the standard query interface provided by each system. All necessary information about a system that is acting as part of the distributed system has to be gained via these interfaces. The collection selection approach has to build an appropriate model by querying the systems individually.

Alternatively, a large collection such as the World Wide Web must be stored on more than one machine so it can be stored and processed. A large collection such as this is often processed in the setting of a cooperative distributed information retrieval system. Full access to the indexes of each system is possible. This enables use of global statistics and any other approaches that require more than the use of the standard query interface.

Collection distribution is the strategy that determines which documents are assigned to which machine. Common approaches include random [Kulkarni and Callan 2010; Ding et al. 2011], source [Xu and Croft 1999] and topic based allocation [Kulkarni and Callan 2010; Xu and Croft 1999]. A common criticism of topic based allocation is that automated methods for document clustering are not scalable for large collections.

Collection selection is the process that determines which machines to search in a distributed system given a query. There are many approaches to collection selection such as SHiRE [Kulkarni et al. 2012], gGlOSS [Gravano and Garcia-Molina 1999] and Cue-Validity Variance [Yuwono and Lee 1997] that use collection statistics to summarize the collection stored on a particular machine.

6.2. Evaluation
Information retrieval evaluation allows comparative evaluation of retrieval systems. System effectiveness is quantified by one of two approaches. User based evaluations measure user satisfaction with the system whereas system based evaluation determines how well a system can rank documents with respect to relevance judgments of users. The system based approach has become most popular due to its repeatability in a laboratory setting and the difficulty and cost of performing user based evaluations [Voorhees 2002].

The Cranfield paradigm [Cleverdon 1967] is an experimental approach to the evaluation of retrieval systems. It has been adapted over the years and provides the theory that supports modern information retrieval evaluation efforts such as CLEF [Fornar et al. 2011], NTCIR [Kando et al. 1999], INEX [Arvola et al. 2011] and TREC [Clarke et al. 2010]. It is a system based approach that evaluates how different systems rank relevant documents. For systems to be compared, the same set of information needs and documents have to be used. A test collection consists of documents, statements of information need, and relevance judgments [Voorhees 2002]. Relevance judgments are often binary. A document is considered relevant if any of its contents can contribute to the satisfaction of the specified information need. Information needs are also referred to as topics and contain a textual description of the information need, including guidelines as to what may or may not be considered relevant. Typically, only the keyword based query of a topic is given to a retrieval system.

Many different measures exist for evaluating the distribution of relevant documents in ranked lists. The simplest measure is precision at \( n \) (P@n), defined as the fraction of...
the documents that are relevant in the first \( n \) documents in the ranked list. A strong justification for the use of \( \text{P}@[n] \) is that it reflects a real use case [Zobel et al. 2009]. Most users only view the first few pages of results presented by a retrieval system. On the web, users may not even view the entire first page of results. Users reformulate their queries rather than continue searching the result list. Zobel et al. [2009] argue that recall does not reflect a user’s need except in some specific circumstances such as medical and legal retrieval that require exhaustive searching. Furthermore, achieving total recall requires complete knowledge of the collection. The mean average precision measure or MAP takes into account both precision and recall. It is the area under the precision versus recall curve. Precision is the fraction of relevant documents at a given recall level, and a recall level is the fraction of returned relevant documents in the set of all relevant documents. As recall increases, precision tends to drop since systems have a higher density of relevant results early in the ranked list. This naturally is the goal of a search engine.

The original Cranfield experiments required complete relevance judgments for the entire collection. This approach rules out large collections due to the necessity of exhaustive evaluation. On large scale collections with millions to billions of documents it is not possible to evaluate the relevance of every single document to every query. Therefore, an approach called pooling was developed [Sparck Jones and van Rijsbergen 1975]. Each system in the experiment returns a ranked list of the first 1,000 to 2,000 documents in decreasing order of relevance. A proportion of the most relevant documents, often around 100 documents, are taken from each system and pooled together, and duplicates are removed since often there is a large overlap between the results lists of different systems. These pooled results are presented to the assessors in an unranked order. The assessors then determine relevance on this relatively small pool.

There has been much debate and experimentation to determine the effect of pooling. As test collections are designed to be reused, new approaches may lead to the return of relevant documents that were not represented in the pool that was assessed. These are un-judged relevant documents. Zobel [1998] found this is true for un-judged runs, but it does not unfairly affect a system’s performance on TREC collections. However, thought is required when designing an information retrieval evaluation experiment. The pool depth and diversity may need to change based on the type and size of the collection. Furthermore, tests such as those suggested by Zobel [1998] can be carried out to determine the reliability and reusability of a test collection.

Saracevic [2008] performed a meta-analysis of judge consistency. They concluded that consistency changes dramatically depending on the expertise of the assessors. Assessors with higher expertise agreed more often on the relevance of a document. Consistency also changed from experiment to experiment using the same assessors. Cranfield based evaluation of information retrieval has been scrutinized heavily over the years. However, assessor disagreement does not affect this type of evaluation because the rank order of each retrieval system does not often change when using different assessments from different judges. The stability only applies when averaging retrieval system performance over a set of queries. The retrieval quality can change drastically from query to query given a different assessor.

7. CLUSTER HYPOTHESIS

The cluster hypothesis connects ad hoc information retrieval and document clustering. Manning et al. [2008] states that “documents in the same cluster behave similarly with respect to relevance to information needs”. If a document from a cluster is relevant to a query, then it is likely other documents in the same cluster are also relevant to the same query. This is due to the clustering algorithm grouping documents with
similar terms together. There may also be some higher order correlation due to effects
caused by the distributional hypothesis [Harris 1954] and limitation of analysis to
the size of a document; i.e. words are defined by the company they keep. It should
be noted that the relevance of a document is assumed in an unsupervised manner
in an operational retrieval system. While our evaluation of document clustering uses
relevance judgments from humans, these are not always available or are noisy as they
are derived from click through data.

The cluster hypothesis has been analyzed in three different forms. Originally the
cluster hypothesis was tested at the collection level by clustering all documents in the
Collection. It was then extended to cluster only search results to improve display and
ranking of documents. Finally, it has been extended to the sub-document level to deal
with the multi-topic nature of longer documents.

7.1. Measuring the Cluster Hypothesis

There have been direct and indirect approaches to measuring the cluster hypothesis.
Direct approaches measure similarity between documents to determine if relevant doc-
uments are more similar than non-relevant documents. Indirect measures use other
IR related tasks such as ad hoc retrieval or document clustering to evaluate the hy-
pothesis. If the cluster hypothesis is true for ad hoc search, then only a small fraction
of the collection needs to be searched while the quality of search results will not be
impacted or potentially improved. The same idea applies to the spread of relevant doc-
uments over a clustering of a collection. If the hypothesis is true, then the documents
will only appear in a few clusters.

Voorhees [1985] proposed the nearest neighbor (NN) test. It analyzes the five nearest
neighbors of relevant documents according to a similarity measure. It determines the
fraction of relevant documents located nearby other relevant documents. The NN test
is comparable across different test collections and similarity measures making it use-
ful for comparative evaluations. Voorhees [1985] states that the test chose five nearest
neighbors arbitrarily. If the number of neighbors is increased, then the test becomes
less local in nature. The NN test overcame problems with the original cluster hypothe-
sis test suggested by [Jardine and van Rijsbergen 1971]. The original test plots the sim-
ilarity between all pairs of relevant–relevant and relevant–non-relevant documents
separately using a histogram approach. It does so for each query and then the results
are averaged over all queries. A separation between the distributions indicates that
the cluster hypothesis holds. The criticism of this approach is that there will always be
more relevant–non-relevant pairs than relevant–relevant pairs. Voorhees [1985] found
that these two different measures give very different pictures of the cluster hypothesis.
However, van Rijsbergen and Jones [1973] found that the original measure was useful
for explaining the different performance of retrieval systems on different collections.

Smucker and Allan [2009] introduce a global measure of the cluster hypothesis based
on the relevant document networks and measures from network analysis. The NN test
is a local measure of clustering as it only inspects the five nearest neighbors of each
relevant document. The measure proposed builds a directed, weighted, fully connected
graph between all relevant documents where vertices are documents and edges rep-
resent the rank of the destination document when the source document is used as a
query. This represents all relationships between relevant documents and not just those
that fall within the five nearest as in the NN test. The normalized, mean reciprocal dis-

tance (nMRD) from small world network analysis [Latora and Marchiori 2001] is used
to measure the global efficiency based upon all shortest path distances between pairs
of documents in the network. Each measure of the cluster hypothesis allows for a dif-
ferent view, allowing for multiple comparisons across multiple approaches to ranking.
In the following sections, we review the measurement of the cluster hypothesis via indirect measures.

7.2. Cluster Hypothesis in Ranked Retrieval

Many retrieval systems return a ranked list of documents in descending order of estimated relevance. Clustering provides an alternative approach to the organization of retrieved results that aims to reduce the cognitive load of analyzing retrieved documents. It achieves this by clustering documents into topical groups. The cluster hypothesis implies that most relevant documents will appear in a small number of clusters. A user can easily identify these clusters and find most of the relevant results. Alternatively, clusters are used to improve ranked results such as pseudo-relevance feedback and re-ranking using clusters.

Early studies in clustering ad hoc retrieval results by Croft [1980] and Voorhees [1986] performed static clustering of the entire collection. More recent studies found that clustering results for each query is more effective for improving the quality of results [Hearst and Pedersen 1996; Leuski 2001; Tombros et al. 2002; Lee et al. 2008]. However, clustering of an entire document collection allows for increased efficiency and its discussion has reappeared recently in the literature [Nayak et al. 2010; De Vries et al. 2011; Kulkarni and Callan 2010].

Evaluation of the cluster hypothesis changed significantly after the result of Hearst and Pedersen [1996] that confirmed that the cluster hypothesis can improve result ranking significantly. It did this by clustering results for each query rather than entire collections. Hearst and Pedersen [1996] revise the cluster hypothesis. If two documents are similar to one query, they are not necessarily similar to another query. As documents have very high dimensionality, the definition of nearest neighbors changes depending on the query. Their system dynamically clusters retrieval results for each query using the interactive Scatter/Gather approach. The Scatter/Gather approach displays five clusters of the top \( n \) documents retrieved by a system. A user selects one or more clusters indicating that only those documents are to be viewed. This process can be applied recursively, performing the Scatter/Gather process again on the selected subset. The experiments using this system were performed on the TREC/Tipster collection using the TREC-4 queries. The top ranked clusters always contained at least 50 percent of the relevant documents retrieved. The results retrieved by the Scatter/Gather system were shown to be statistically significantly better than not using the system according to the \( t \)-test.

Crestani and Wu [2006] investigate the cluster hypothesis in a distributed information retrieval environment. Even though there are many issues that introduce noise when presenting results from heterogeneous systems in a distributed environment, they demonstrate that clustering is still an effective approach for displaying search results to users.

Lee et al. [2008] present a cluster-based resampling method for pseudo-relevance feedback. Previous attempts to use clusters for pseudo-relevance feedback have not resulted in statistically significant increases in retrieval efficiency. This is achieved by repeatedly selecting dominant documents, which is motivated by boosting from machine learning and statistics that repeatedly selects hard examples to move the decision boundary toward hard examples. It uses overlapping or soft clusters where documents can exist in more than one cluster. A dominant document is one that participates in many clusters. Repeatedly sampling dominant documents can emphasize the topics in a query and thus this approach clusters the top 100 documents in the ranked list. The most relevant terms are selected from the most relevant clusters and are used to expand the original query. It was most effective for large scale document collections where there are likely to be many topics present in the results for a query.
Re-ranking of the result list using clusters has found to be effective using both vector space approaches [Lee et al. 2001; Lee et al. 2004], language models [Liu and Croft 2004] and local score regularization [Diaz 2005]. In these approaches, the original ranked list is combined with the rank of clusters, rather than just the original query-to-document score.

Kulkarni and Callan [2010] investigate the use of topical clusters for efficient and effective retrieval of documents. This work finds similar results to those discovered at the INEX XML Mining track [Nayak et al. 2010; De Vries et al. 2011] when clustering large document collections, that relevant results for a given query cluster tightly and only fall in a few topical document clusters. Kulkarni and Callan solve the scalability problem of clustering by using sampling using k-means with the Kullback-Leibler divergence similarity measure. This work also implements and evaluates a collection selection approach called ReDDE and finds that topical clusters outperform random partitioning and that as little as one percent of a collection needs to be searched to maintain early precision.

7.3. Sub-document Cluster Hypothesis in Ranked Retrieval

Lamprier et al. [2008] investigate the use of sub-document segments or passages to cluster documents in information retrieval systems. They extend the notion of the cluster hypothesis to the sub-document level where different segments of a document can have different topics that are relevant to different queries. If the cluster hypothesis holds at the sub-document level, then relevant passages will tend to cluster together. The approach proposed by Lamprier et al. [2008] returns whole documents but uses passages to cluster multi-topic documents more effectively.

Lamprier et al. [2008] suggest several different approaches to the segmentation of documents into passages. The authors chose to select an approach based on an evolutionary search algorithm that minimizes similarity between segments called SegGen [Lamprier et al. 2007]. They dismissed using arbitrary regions of overlapping text even though this was shown to be effective for passage retrieval by Kaszkiel and Zobel [2001]. The reasons given are added complexity and that only disjoint subsets of the documents are appropriate in this case.

Lamprier et al. [2008] performed experiments using sub-document clustering on ad hoc retrieval results with the aim of improving the quality. The authors did not consider sub-document clustering of the entire collection because earlier studies suggested this was not effective for whole document clustering. They found that sub-document clustering allowed better thematic clustering that placed more relevant results together than whole document clustering.

8. USING AD HOC RELEVANCE JUDGMENTS TO EVALUATE DOCUMENT CLUSTERING

In Section 4.1 we introduced some of the issues with determining a ground truth set of category labels for large document collections. As described in Section 6.2, the information retrieval evaluation community has already dealt with the problem of assessor load in ad hoc relevance evaluation via the use of pooling. Pooling reduces assessor load and topics evaluated are specific and well defined. By using ad hoc relevance to assess document clustering, instead of category labels for every document in a collection, the issues of assessor load is alleviated. Many categories that exist in ground truths for document collections are lofty and not well defined ideas such as “Arts”. What Arts is and is not could be debated ad infinitum.

The cluster hypothesis suggests that documents that cluster together tend to have relevance to similar queries. The evaluation strategy we are proposing aims to evaluate the utility of clustering in collection selection. The goal of clustering is to minimise the spread of relevant results of ad hoc queries over a clustering solution. The purpose
of clustering in this context is to determine the distribution of a collection over multiple machines. We have a dual optimisation problem – it is desirable to maximise the number of clusters while minimising the spread of relevant results of ad hoc queries over the clusters. Search efficiency can be increased with the distribution of clusters on more machines. However, since it is not possible to produce clusters that split the collection to satisfy all conceivable ad hoc queries. A good clustering solution is expected to optimise the distribution such that for most ad hoc queries most of the results can be found in a small set of clusters. The goal of collection selection is then to rank the clusters to identify the order in which they should be searched to satisfy any given query.

Guyon et al. [2009] argue that the context of clustering needs to be taken into account during evaluation. The evaluation we are proposing tests the clustering hypothesis in the information retrieval specific. Clustering is intended to facilitate document distribution and collection selection for ad hoc retrieval, and it is tested in that setting. This differs greatly from evaluation where authors assign categories to documents and the categories are then used as the ground truth for the evaluation of clustering. Guyon et. al. [Guyon et al. 2009] argue, and we agree, that ground truth based evaluations are unsound. This is particularly true when it comes to an information retrieval setting where the number of potential topics are virtually unconstrained. It is a near impossible task to compare alternative clustering possibilities by inspecting large numbers of documents in clusters. In contrast, the evaluation of topics represented as queries in an ad hoc retrieval system has been shown to be stable across different assessors [Saracevic 2008]. These relevance judgments have been the backbone of ad hoc information retrieval system evaluations for many years. They have also been exposed to criticism and review by many of the top researchers in the field. By exploiting this high quality, human generated information, we can have great confidence that we are testing clustering in the context of its use. The context is specifically clustering of documents in an information retrieval setting.

A ground truth based evaluation treats clustering as if it were a classification problem. If you want to predict known categories for a document, then it is highly likely that supervised learning approaches will give higher classification accuracy. In contrast, the ad hoc retrieval evaluation we are proposing is treating clustering as a clustering problem. It is directly motivated by the cluster hypothesis from information retrieval.

As each query represents a topic, albeit very focussed, we argue that this evaluation is also sound for a general evaluation of document clustering. If relevant documents are not coherent with respect to the cluster hypothesis, then they are probably not of use for any situation where document clustering is required. The cluster hypothesis refers to the, $\sum_{i=1}^{n} C_{vi}$, number of possible queries whose ranked results are clusters themselves, where $v$ is the size of the vocabulary. The cluster ends where the relevant results end in the ranked list. It supports the notion that ad hoc retrieval is ad hoc clustering of results given a query.

The evaluation of document clustering using ad hoc information retrieval can be viewed as being similar to an evaluation using a multi label category based ground truth. A document can be relevant to more than one query. The class imbalance is much greater when using relevance judgments. There are many more non-relevant than relevant documents for any given query. Furthermore, if each query was treated as a topic and relevant documents were assigned a class label if and only if they are relevant to a query, the entire collection would not be covered. For example, the 52 queries from the INEX 2010 ad hoc track only contain 5471 relevant documents for the entire 2,660,190 document INEX XML Wikipedia collection.
9. NORMALISED CUMULATIVE CLUSTER GAIN

The NCCG evaluation measure has been used for the evaluation of document clustering at INEX [Nayak et al. 2010; De Vries et al. 2011]. It is motivated by van Rijsbergen’s cluster hypothesis [Jardine and van Rijsbergen 1971]. If the hypothesis holds true, then relevant documents will appear in a small number of clusters. A document clustering solution can be evaluated by measuring the spread of relevant documents for the given set of queries.

NCCG is calculated using manual result assessments from ad hoc retrieval evaluation. Evaluations of ad hoc retrieval occur in forums such as INEX [Arvola et al. 2011], CLEF [Ponter et al. 2011] and TREC [Clarke et al. 2010]. The manual query assessments are called the relevance judgments and have been used to evaluate ad hoc retrieval of documents. The process involves defining a query based on the information need, a retrieval system returning results for the query and humans judging whether the results returned by a system are relevant to the information need.

The NCCG measure tests a clustering solution to determine the quality of clusters relative to the optimal collection selection. Collection selection involves splitting a collection into subsets and recommending which subsets need to be searched for a given query. This allows a retrieval system to search fewer documents, resulting in improved runtime performance over searching the entire collection. The NCCG measure has complete knowledge of which documents are relevant to queries and orders clusters in descending order by the number of relevant documents it contains. We call this measure an “oracle” because it has complete knowledge of relevant documents. A working retrieval system does not have this property, so this measure represents an upper bound on collection selection performance.

Better clustering solutions in this context will tend to group together relevant results for previously unseen ad hoc queries. Real ad hoc retrieval queries and their manual assessment results are utilised in this evaluation. This approach evaluates the clustering solutions relative to a very specific objective – clustering a large document collection in an optimal manner in order to satisfy queries while minimising the search space. The measure used for evaluating the collection selection is called Normalised Cumulative Cluster Gain (NCCG) [Nayak et al. 2010].

The Cumulative Gain of a Cluster (CCG) is defined by the number of relevant documents in a cluster, \( \text{CCG}(c,t) = \sum_{i=1}^{n_i} \text{Rel}_i \). A sorted vector CG is created for a clustering solution, \( c \), and a topic, \( t \), where each element represents the CCG of a cluster. It is normalised by the ideal gain vector,

\[
\text{SplitScore}(t,c) = \sum_{\text{CG}} \frac{\text{cumsum}(\text{CG})}{n_r^2},
\]

where \( n_r \) is total number of relevant documents for the topic, \( t \). The worst possible split places one relevant document in each cluster represented by the vector CG1,

\[
\text{MinSplitScore}(t,c) = \sum_{\text{CG1}} \frac{\text{cumsum}(\text{CG1})}{n_r^2}.
\]

NCCG is calculated using the previous functions,

\[
\text{NCCG}(t,c) = \frac{\text{SplitScore}(t,c) - \text{MinSplitScore}(t,c)}{1 - \text{MinSplitScore}(t,c)}.
\]

It is then averaged across all topics.
10. DIVERGENCE FROM A RANDOM BASELINE

Divergence from a random baseline is a technique for the evaluation of document clustering. It ensures cluster quality measures are performing work that prevents ineffective clusterings from giving high scores to clusterings that provide no useful result. These concepts have been defined and analyzed in previous work by De Vries et al. [2012]. It was defined and analyzed using intrinsic and extrinsic evaluations of clustering, including use of document categories and ad hoc relevance judgments.

An ineffective clustering is one that achieves a high score according to a measure of document cluster quality but provides no value as a clustering solution. Divergence from a random baseline addresses ineffective clusterings in evaluation. It introduces a notion of work performed by a clustering where ineffective cases appear to perform no useful learning.

10.1. Ineffective Clustering

An ineffective clustering produces a high score according to an evaluation measure but does not represent any inherent value as a clustering solution.

The Purity evaluation measure has an obvious ineffective case. If each cluster contains one document, then it is 100% pure with respect to the ground truth. A single document is the majority of the cluster. As the goal of clustering is to produce groups of documents or to summarize the collection, this is obviously flawed as it does neither. The same applies to the Entropy measure as the probability of a label for a cluster is 100%, resulting in the highest possible Entropy score.

The NCCG measure is ineffective when one cluster contains all the documents except for every other cluster containing one document. The NCCG measure orders clusters by the number of relevant documents they contain. A large cluster containing most documents will almost always be ranked first. Therefore, almost all relevant documents will exist in one cluster, achieving almost the highest score possible.

10.2. Work Performed by a Clustering

To overcome ineffective clusterings in the previous section, we introduce the concept of work performed by a clustering approach. Work is defined as an increase in quality of a clustering over a simple approach that ignores the documents being clustered. A useful clustering performs work beyond an approach that is purely random and ignores document content. If a random approach that performs no useful learning performs equally to an approach that attempts to learn from that data, it would appear that nothing has been achieved by analyzing the data. We suggest that an ineffective clustering performs no useful learning.

10.3. Divergence from a Random Baseline

Many measures of cluster quality can give high quality scores for particular clustering solutions that are not of high quality by changing the number of clusters or number of documents in each cluster.

Measures that can be misled by creating an ineffective clustering can be adjusted by subtraction from a randomly generated clustering with the same number of clusters with the same number of documents in each cluster. The random baseline distributes documents into buckets the same size as the clusters found by the clustering algorithm. Apart from the random assignment of documents to clusters, the random baseline appears the same as the real solution. Therefore, each clustering evaluated requires a random baseline that is specific to that clustering. The baseline is created by shuffling the documents uniformly randomly and splitting them into clusters the same...
size as the clustering being measured. The score for the random baseline clustering is subtracted from the matching clustering being measured.

The divergence from a random baseline approach can be applied to any measure of cluster quality whether it is intrinsic or extrinsic. However, it does require an existing measure of cluster quality. It is not a measure by itself, but an approach to ensure a clustering is doing something sensible. Although we have highlighted its use for document clustering evaluation, it can be used for any clustering evaluation.

There are two issues at play here. Firstly, different distributions of cluster sizes can lead to arbitrarily high scores. The second issue is determining if the clustering algorithm is effectively learning with respect to a measure of quality. The divergence from a random baseline takes care of ineffective solutions in either case. If the internal ordering of clusters is no better than random noise, then it achieves a score of zero. A negative score could be achieved as the random baseline scores a positive value using most measures on most data sets. It is possible for a clustering to have a worse score than the baseline. For example, a clustering approach could maximize dissimilarity of documents in clusters. This will create a solution where the most dissimilar documents are placed together, resulting in a worse score than random assignment. The random assignment does not bias the clustering towards or away from the measure of quality. If a clustering approach is learning something with respect to the measure of quality, then it is expected that is will be biased towards it. Alternatively, if we reverse the optimization process, it should be biased away from it.

Let \( \omega = \{w_1, w_2, \ldots, w_K\} \) be the set of clusters for the document collection \( D \) and \( \xi = \{c_1, c_2, \ldots, c_J\} \) be the set of categories. Each cluster and category are a subset of the document collection, \( \forall c \in \xi, w \in \omega : c, w \subset D \). We define the probability of a category in the baseline given a cluster as, \( P_b(c_j|w_k) = \frac{|c_j|}{\sum_i|c_i|} \). The probability of a category given a cluster in the baseline only depends on the size of the categories. The baseline is a uniformly randomly shuffled list of documents that have been split into clusters that match the cluster size distribution in the solution being evaluated. Thus, within each cluster in the baseline is random uniform noise. It is not biased by the document representation. Therefore, it is expected categories will occur at a rate proportional to the category’s size. For example, if there are three categories \( A, B, C \) containing 10, 20, 30 documents, each cluster in the baseline is expected to contain approximately \( 10^\frac{1}{3}, 20^\frac{1}{3}, 30^\frac{1}{3} \). This only reflects the size distribution of the categories.

We let any measure of a cluster quality be interpreted as a probability. Although this is not formally the case for all measures, it serves as a reasonable explanation. We define the probability of a category in a cluster given the ground truth as, \( P_s(c_j|w_k) = \frac{|c_j \cap w_k|}{|w_k|} \) any measure of cluster quality.

The Purity measure assigns an actual probability to each cluster when there is a single label ground truth. All the probabilities combined accumulate to one, \( \sum_j \frac{|c_j \cap w_k|}{|w_k|} = 1 \), and the category with the largest maximum likelihood estimate is assigned to each cluster, \( P_{\text{purity}}(c_j|w_k) = \arg \max_j \frac{|c_j \cap w_k|}{|w_k|} \). This is the proportion of the cluster that has the majority category label. It also represents the same process of using clustering for classification with labeled data where an unseen sample is labeled based on the majority category label of the closest cluster. We define \( d \) as a document in \( D \). The ground truth is restricted to being single label where a document, \( d \), only has only one label in one category, in the ground truth, \( \forall d \in D, c_j \in \xi, c_j \in \xi : d \in c_j \wedge d \notin c_i \land c_i \neq c_j \).

The adjusted measure is the difference between submission and baseline. We define the adjusted probability of a category given a cluster as, \( P_s(c_j|w_k) = P_s(c_j|w_k) - P_b(c_j|w_k) \).
An alternative formal view of divergence from a random baseline can be defined by a quality function, \( m : \mathbb{P}(Z \times Z) \rightarrow \mathbb{R} \), that takes a set of clusters as a set of set of (document, category label) pairs, \( s \), and returns a real number indicating the quality of the clustering. Examples of these cluster quality functions are Entropy, F1, NCCG, Negentropy, NMI and Purity. There exists a function, \( r : \mathbb{P}(Z \times Z) \rightarrow \mathbb{P}(Z \times Z) \), that generates a random baseline, \( b \), given a clustering solution, \( s \). The baseline has the same number of clusters as the clustering solution, \( |b| = |s| \). For every cluster in each of the original clustering, \( s \), and the baseline, \( b \), the corresponding clusters contain the same number of documents, \( \forall k : |s_k| = |b_k| \). The adjusted measure, \( m_a : \mathbb{P}(Z \times Z) \rightarrow \mathbb{R} \), becomes, \( m_a(s) = m(s) - m(r(s)) \).

10.4. Conclusion

In this section, we introduced problems encountered in the evaluation of document clustering. This is the concept of ineffective clustering and a notion of work. The divergence from random baseline approach deals with these corner cases and increases the confidence that a clustering approach is achieving meaningful learning with respect to any view of cluster quality. It is also applicable to any clustering evaluation but it was only discussed in the context of document clustering. It allowed differentiation of actual problem clusterings at the INEX 2010 XML Mining track. For a more detailed analysis please consult the prior publication by De Vries et al. [2012]. Additionally, the evaluation software that implements the Divergence from Random Baseline approach is available on the Machine Learning Open Source Software website.

11. K-TREE AND TOPSIG

We introduce a new method for collection distribution using K-tree and TopSig to perform efficient and scalable clustering of large scale document collections into large numbers of clusters. Modifications to the K-tree algorithm and data structure have been made to work with the binary signatures produced by TopSig.

The previous work of Kulkarni and Callan [2010] only investigates clustering for collection distribution with relatively few document clusters. They produced 100 clusters for the 50 million document ClueWeb09 Category B collection and 1000 clusters for the 500 million document ClueWeb09 Category A collection. We present an approach that can cluster 50 million documents into approximately 140,000 clusters in ten hours. This is completed using a single thread of Java code. There has been no parallelisation or optimisation using native code to take full advantage of modern multi-core processors and other low level features.

We first introduce TopSig and K-tree and then explain how we combined them to produce an effective large scale clustering approach.

11.1. TopSig

TopSig [Geva and De Vries 2011] offers a radically different approach to the construction of file signatures when compared to traditional file signatures. Traditional file signatures [Faloutsos and Christodouakis 1984] have been shown to be inferior to approaches using inverted indexes, both in terms of the time and space required to process and store the index [Witten et al. 1999; Zobel 1998]. However, TopSig overcomes previous criticisms aimed at file signatures by taking a principled approach using the vector space model, dimensionality reduction and numeric quantization. Previous approaches to file signatures were constructed in an ad hoc fashion by combining random binary signatures using a bitwise XOR which is a Bloom filter [Bloom 1970] for

2http://www.mloss.org/software/view/468/
3http://topsig.googlecode.com
the terms contained in documents. In contrast, TopSig randomly indexes a weighted term-by-document matrix and then quantizes it. TopSig is competitive with state of the art probabilistic and language retrieval models at early precision, and clustering approaches [Geva and De Vries 2011].

Let $\mathcal{D} = \{d_1, d_2, ..., d_n\}$ be a document collection of $n$ documents signatures, $\mathcal{D} \subset \{+1, -1\}^d$, $|\mathcal{D}| = n$. Let $\mathcal{F} = \{f_1, f_2, ..., f_n\}$ be the same document collection as $\mathcal{D}$ where each document is represented by a $v$-dimensional real valued vector, $\mathcal{F} \subset \mathbb{R}^v$, $|\mathcal{F}| = n$, where $v$ is the size of the vocabulary of the document collection. $\mathcal{F}$ is the term-by-document matrix in the full space of the collection vocabulary which underlies most modern retrieval systems.

TopSig indexes documents using a mapping function, $m : \mathbb{R}^v \rightarrow \{+1, -1\}^d$, that maps a document from the original $v$-dimensional continuous real valued term space, to a $d$-dimensional discrete binary valued space. The index is constructed using a mapping function, $\mathcal{D} = \{f \in \mathcal{F} : m(f)\}$. The mapping function creates a sparse random ternary index vector of $d$-dimensions for each term in the document with +1 and -1 values in random positions and the majority of positions containing 0 values. These randomly generated codes are almost orthogonal to each other and have been shown to provide comparable quality to orthogonal approaches such as principle component analysis [Bingham and Mannila 2001]. The index vector is multiplied by the term weight and added to a $d$-dimensional real valued vector that represents the document. Once all the terms in a document have been processed, this reduced dimensionality document vector is then quantized to a $d$-dimensional binary vector by thresholding each value in each dimension to 1 if greater than 0 and 0 otherwise. The 1 and 0 values in the binary vector represent +1 and -1 values. This mapping function can be applied to each document independently, meaning that new documents can be indexed in isolation without having to update the existing index. This is a key advantage to random indexing [Sahlgren 2005] over other dimensionality reduction techniques such as latent semantic analysis [Deerwester et al. 1990] which requires global analysis of the term-by-document matrix using the singular value decomposition [Golub and Kahan 1965].

The indexing process of TopSig is similar to that of SimHash [Charikar 2002]. However, TopSig uses signatures an order of magnitude longer than SimHash, and it uses much sparser random codes. The search process for ad hoc retrieval also differs, where TopSig searches in the subspace of the query and applies relevance feedback. In contrast, SimHash has predominantly been applied to nearest duplicate detection where relatively short signatures are used to find the few nearest neighbours of a document.

The binary vectors in $\mathcal{D}$ provide a faithful representation of the original document vectors in $\mathcal{F}$. The topological relationships in the original space are preserved in the reduced dimensionality space. This is supported by the Johnson-Lindenstrauss lemma [Johnson and Lindenstrauss 1984] that states if points in a high-dimensional space are projected into a randomly chosen subspace, of sufficiently high-dimensionality, then the distances between the points are approximately preserved. It also states that the number of dimensions required to reproduce the topology is asymptotically logarithmic in the number of points.

TopSig is advantageous for increased computational efficiency of document-to-document comparisons. Examples of this include clustering, classification, filtering, relevance feedback, near duplicate detection and explicit semantic analysis. TopSig has been shown to provide a 1 to 2 magnitude increase in processing speed for document clustering [Geva and De Vries 2011] over traditional sparse vector representations when using the k-means algorithm. This paper introduced a modified k-means algorithm for working with binary document signatures. This is used when combining K-tree with TopSig as K-tree uses k-means to perform splits in its tree structure.
11.2. K-tree

K-tree [De Vries et al. 2009] is a height balanced cluster tree. It was first introduced in the context of signal processing by Geva [Geva 2000]. It has been applied to document clustering [De Vries and Geva 2009; ?; De Vries et al. 2010], image processing and computer vision [Girgensohn et al. 2011; Atsumi 2012a; Atsumi 2010; Atsumi 2012b; Atsumi 2013] and movement recognition from gesture data [?]. The algorithm is particularly suitable for clustering of large collections due to its low complexity. It is a hybrid of the B⁺-tree and k-means algorithm. The B⁺-tree algorithm is modified to work with multi dimensional vectors and k-means is used to perform node splits in the tree. K-tree is also related to Tree Structured Vector Quantization (TSVQ) [Gersho and Gray 1993]. TSVQ recursively splits the data set, in a top-down fashion, using k-means. TSVQ does not generally produce balanced trees.

K-tree achieves its efficiency through execution of the high cost k-means step over very small subsets of the data. The number of vectors clustered during any step in the K-tree algorithm is determined by the tree order and it is independent of collection size. It is efficient in updating the collection while maintaining clustering properties through the use of a nearest neighbour search tree that directs new vectors to the appropriate leaf node.

The K-tree forms a hierarchy of clusters. This hierarchy supports multi-granular clustering where generalisation or specialisation is observed as the tree is traversed from a leaf towards the root or vice versa. The granularity of clusters can be decided at run-time by selecting clusters that meet criteria such as distortion or cluster size.

11.3. Combining K-tree and TopSig

K-tree has only previously been combined with floating point vectors in the Random Indexing (RI) K-tree [De Vries et al. 2009]. Both RI and TopSig produce dense vectors from sparse document vectors. The advantage of TopSig is that it exploits bit level parallelism by processing 64 dimensions of the vector in each operation on a 64-bit CPU.

Both k-means and K-tree perform similar operations of assigning nearest neighbours to centroids and updating the centroids as means of the nearest neighbours. This corresponds to the expectation and maximisation steps of the EM algorithm [Dempster et al. 1977]. When using vectors that are not packed binary signatures, the nearest neighbour search that associates vectors with centroids dominates the execution times of the algorithms. The updating of centroids as means is relatively quick. When using binary signatures this phenomenon is reversed. When calculating the mean of a set of binary signatures, each bit is unpacked and added to a integer valued vector for each dimension. Once all examples have been inspected, the result is then packed back into a binary signature by taking the sign of the integer valued vector. This transformation of a set of binary vectors into an integer valued vector is understandably much slower than comparison of vectors using Hamming distance due to the larger intermediate representation of an integer valued vector. This poses a problem when combining K-tree and TopSig. The K-tree algorithm updates all centroids along the insertion path for every vector inserted into the tree. As updating of means occurs approximately as frequently as nearest neighbour comparisons in the K-tree this poses a problem. This problem does not effect k-means when using binary signatures in such a significant way, as there are relatively few updates of means to nearest neighbour comparisons, \( k \ll n \), when \( k \) is the number of clusters and \( n \) is the number of vectors being clustered. Therefore, to ensure the scalability of K-tree when combined with

---

4http://ktree.sf.net
Distributed Information Retrieval

TopSig, centroids in the trees have a delayed update mechanism. Upon insertion, centroids are only updated when a split happens with k-means, or there have been 1000 vectors inserted below the centroid. The choice of 1000 is arbitrary but has worked effectively for document clustering.

12. CBM625: CLUSTER BEST MATCH 625

In this section, we introduce a new collection selection approach called Cluster Best Match 625 (CBM625). It is based upon Okapi BM25 [Robertson et al. 1995] but combines document weights to rank clusters in the collection selection process. The name is derived directly from Best Match 25 where the document weights are squared, therefore, $BM25^2 = BM625$, and it is a cluster ranking process, hence the prefix.

Okapi BM25 has consistently performed well in the evaluation of ad hoc information retrieval at TREC, INEX, NTCIR and CLEF. It is considered to be state of the art in terms of document ranking functions. Therefore, we have built upon BM25 to create the collection selection approach CBM625.

BM25 is a probabilistic model where each term in each document is assigned an estimate of relevance to a query. There is a matrix representing each document in a collection, $D_{t \times n}$, where $t$ is the number of terms and $n$ is the number of documents. This is the inverted index that underlies most modern retrieval systems. We have used a modified implementation of Okapi BM25 in the ATIRE search engine \footnote{http://www.atire.org} [Trotman et al. 2010] which is defined in Equation 7.

$$RSV_d = \sum_{t \in q} \log \left( \frac{n}{df_t} \right) \frac{(k_1 + 1)tf_{td}}{k_1 \left(1 - b + b \left( \frac{L_d}{L_{avg}} \right) \right) + tf_{td}}$$

In Equation 7, $n$ is the total number of documents, and $df_t$ and $tf_{td}$ are the number of documents containing the term $t$ and the frequency of the term in document $d$, and $L_d$ and $L_{avg}$ are the length of the document $d$ and the average length of all documents. The empirical parameters $k1$ and $b$ have been set to 0.9 and 0.4 respectively by training on the previous years INEX 2008 ad hoc queries, which uses a smaller 659,338 document Wikipedia collection from 2006. These parameters have been used for all test collections.

When ranking clusters, these estimates of relevance from BM25 need to be combined in some manner; i.e. a cluster representative must be built. Therefore, another clustered inverted index, $C_{t \times k}$, is built where $t$ is the number of terms and $k$ is the number of clusters in the collection. We have found that using squared document weights is the most effective approach for combining BM25 documents weights to rank clusters. Squaring documents weights has the affect that heavily weighted documents have a higher impact in the ranking of a cluster; i.e. $2^2 << 10^2$. We initially experimented with a summation of all weights in the cluster, as each weight for each term in $D$ is an estimate of relevance of that term to a query containing the term. However, using squared weights proved more effective on the Wikipedia where more relevant documents were contained in highly ranked clusters. A justification for why this increases retrieval quality is that heavily weighted documents have a larger impact on the top of the ranked list. A document with a heavy weight is more important to that particular query, and therefore should be given higher importance. We also experimented with representing each document as the mean of all documents in the cluster. This reduced performance, requiring more clusters to be searched to find relevant documents. An explanation for this phenomena is that taking the mean normalises the BM25 weights
by the number of documents in a cluster. As BM25 already has length normalisation, we propose that normalising by cluster size, destroys the initial document length normalisation of BM25. CBM625 is defined in Equation 8 where \( RSV_c \) is the Retrieval Status Value assigned to clusters that is used to rank clusters in descending order of relevance. The terms in Equation 8 are the same terms as used in Equation 7.

\[
RSV_c = \sum_{d \in c} \sum_{t \in q} \left[ \log \left( \frac{n}{dft} \right) \frac{(k_1 + 1)tf_{td}}{k_1 (1 - b) + b \left( \frac{tf_{td}}{L_{avg}} \right)} \right]^2
\]  

12.1. The Ranking Process

Ranking documents using collection selection becomes a two step process. First the clustered index, \( C \), is consulted to rank the clusters of the collection. These can be any clusters or any grouping of documents. For example, random assignment of documents to clusters or source based groupings can be used. Once the clusters have been ranked, then a cutoff for when to stop searching clusters must be chosen. We simply search the first \( f \) most highly ranked clusters. The first step in the ranking process generates a list of clusters to be searched. In a distributed system, the separate indexes for each cluster are searched using standard BM25, and the results are combined to produce a single ranking. As we are able to index all of the test collections on a single machine, we simply filter the BM25 index to the set of documents contained in the first \( f \) clusters for each query. This simulates searching each of the clusters determined by the \( f \) most highly ranked clusters per query.

13. EVALUATION AND RESULTS

We have evaluated the collection distribution, selection and search approach based upon TopSig K-tree and CBM625 using the INEX 2009 Wikipedia collection, and the ClueWeb09 Category B collection. The INEX Wikipedia collection contains 2.67 million documents with a vocabulary of 2.1 million terms. The mean document length in the Wikipedia has 360 terms, the shortest has 1 term and the longest has 38,740 terms. We used all 52 queries from INEX 2010 [Arvola et al. 2011] for which there are relevance judgments. The ClueWeb09 Category B collection consists of 50 million documents and a vocabulary of 96.1 million terms. The mean ClueWeb09 Category B document length is 918 terms. We used queries 1-50 from the TREC 2009 Web Track [Clarke et al. 2009].

All experiments were conducted in the Queensland University of Technology Big Data Lab on a dual socket Xeon E5-2665 system. It has 16 CPU cores, 32 hardware threads and 256GB of main memory. Each CPU package has 20MB L3 cache and 8 cores connected in a ring architecture with a standard frequency of 2.4GHz, but when a single thread is running it has a frequency of 3.1GHz. It has become clear in this study that the root node of a search tree such should be adjusted to the cache size of the processors. This means that the search tree can have a root node containing many more centroids than the rest of the tree. This will improve cache hits and also has the advantage of dealing with the curse of dimensionality investigated by De Vries and Geva [2012]. However, we have left detailed investigations of the implications of these phenomena to later research. Note that at least 128GB of main memory is required to complete these experiments.

We have released a package containing all the software from the experiments. This includes ATIRE, TopSig, K-tree, CBM625 and all the scripts used to conduct the experiments. It is available for download from the K-tree project on SourceForge.

\[\text{http://sourceforge.net/projects/ktree/files/docclust_ir/docclust_ir.tar.gz}\]

In this section, we conduct several experiments to determine the effect of different parameters and clustering approaches on the INEX Wikipedia collection. These results are used to justify our choice of approach on the ClueWeb document collection, where experimentation is not particularly practical due to much longer processing times.

We have conducted an experiment to determine if there is any difference between using k-means and K-tree clusters for ranking with CBM625. K-tree has been shown [?] to produce lower quality clusters both in terms of internal quality via distortion as the Root Mean Squared Error of the clustering and in terms of external quality via the traditional classes-to-clusters approach. However, when producing a large number of clusters on a collection containing millions of documents there is a huge difference in execution time [?]. Interestingly, as the K-tree order becomes smaller, producing more clusters, the execution time is reduced because the clustering process to split nodes in the tree converges quicker [?]. Additionally, we present TopSig K-tree for the first time in this paper that further increases the computational efficiency of document clustering. 1559 clusters of the Wikipedia collection were produced using k-means and K-tree using 4096 bit TopSig signatures. The k-means implementation works directly with the bit signature representations of TopSig where both centroids and document vectors are all bit signatures and are compared using Hamming distance. Clustering using this modified k-means and 4096 bit signatures have previously been shown to produce no statistically significant difference in cluster quality compared to state of the art clustering approaches using sparse real value vectors while providing one to two orders of magnitude decrease in execution time [Geva and De Vries 2011]. Both of these 1559 clusterings of the collection were ranked using the CBM625 approach where only the first \( f \) highest ranked clusters were searched. 104 points were sampled along this curve, every 15 clusters, starting with only searching the highest ranked cluster; i.e. the first 1, 15, 30, 45, 60...1545 clusters. Exhaustive search by searching all clusters produces the same MAP score as reported by Trotman et al. [2010] of 0.38, and this is to be expected as we are using the same BM25 index exported from ATIRE.

Figure 1 demonstrates that using clusters produced by k-means or K-tree has very little difference in retrieval behaviour when using the CBM625 ranking approach. The x axis is the average fraction of the 2.67 million documents searched when searching the first \( f \) clusters, where a value of 1 indicates the whole collection was searched. The y axis is the fraction of the maximum MAP, where are value of 1 is equivalent to exhaustive searching. Therefore, when ranking clusters using CBM625 there is no advantage to use k-means over K-tree, while there is a large computational advantage in clustering time to using K-tree.

We have conducted an experiment to determine the effect of the number of clusters on the distribution of relevant documents when these clusters are ranked by CBM625. Varying the number of clusters in the CBM625 cluster ranking process changes its properties. When every document is in its own cluster it becomes similar to the BM25 ranking process except that the term-document weights are squared. In this case, every document is represented in the posting lists produced by CBM625. Note that CBM625 produces an extra inverted index that is consulted to determine which documents to search using the standard BM25 ranking process. As the number of clusters decreases, each element in the posting list represents more and more documents, with the extreme case of only having one cluster, where all documents are represented by one element in each posting list, always forcing exhaustive search. So, as the number of clusters increases, the number of documents summarised in each position in the posting lists of CBM625 decreases. We have clustered the Wikipedia into 100 and 1559 clusters using TopSig k-means as in the previous experiment. We have also pro-
duced clusters that do not consult document content and uniformly randomly assign documents to each of the clusters. This is similar to the divergence from random baseline approach except that the cluster size distribution is not matched. However, the TopSig k-means clustering approach was shown not to have any issues caused by cluster size distribution when evaluating using document categories or ad hoc relevance judgements at the INEX 2010 XML Mining track [De Vries et al. 2011]. For the 1559 clusters, retrieval performance was sampled every 15 clusters staring with 1 cluster, as in the previous experiment. For 100 clusters, retrieval performance was sampled after every cluster. Figure 2 demonstrates that using more clusters allows more relevant documents to be ranked in earlier clusters. By increasing the number of clusters the CBM625 ranking process is able to find clusters containing relevant documents and rank them higher. Both of the clusterings allow relevant documents to be ranked earlier than the uniform random baseline. This indicates that useful learning is taking place by clustering documents with respect to the ranking of relevant documents. This further confirms the clusters hypothesis.

We have conducted an experiment to determine the effect of using the sum of squared BM25 weights to represent clusters in the ranking process. As in the previous experiments we sampled 104 points from the 1559 clusters starting with 1 cluster. Clusters were represented as a sum of BM25 weights and the sum of squared BM25 weights. Figure 3 highlights that quality is improved for early recall. Therefore, we conducted a further experiment where each of the first 1 through 100 clusters of the 1559 clusters were selected in the search process. We measure the difference between the sum and sum of squares approach using P@10 in Figure 4, P@30 in Figure 5 and P@100 in Figure 6. As can been seen in all of the graphs of early precision, the sum of squares approach finds more relevant documents earlier than using the sum of BM25 weights.
13.2. Experiments using ClueWeb 2009 Category B

In this section, we conduct an experiment that has the same experimental setup as the experiment performed by Kulkarni and Callan [Kulkarni and Callan 2010] using the ClueWeb collection.

We have indexed the 50 million document collection using both TopSig and ATIRE. We have used BM25 with ATIRE to produce a quantized index. It has been used with exactly the same settings as previous experiments on the Wikipedia, where the tuning parameters are $k_1 = 0.9$ and $b = 0.4$. TopSig was used to create 4096 bit document...
signatures. These signatures were used to construct a TopSig K-tree with a node size $m = 1000$ in 10 hours. This created 139,641 document clusters of the collection by selecting the lowest level “codebook” clusters from the K-tree. The K-tree was 3 levels deep where the first level is the root of the tree, the second level is the codebook clusters and the third levels is the 50 million document signatures inserted into the tree. We evaluated P@10, P@20 and P@30 when only searching the first $f$ clusters as determined by CBM625. We have evaluated the efficiency of only searching the first $f$ clusters in the same manner as Kulkarni and Callan [2010]. This is determined by
the number of documents searched in the first $f$ clusters when averaged across all 50 queries, giving the average number of documents searched per query. We have performed significance tests for each of the rankings produced by CBM625 by comparing it to the exhaustive search produced by ATIRE BM25. This is performed by a two tailed paired $t$-test between the scores for each of the 50 queries for both CBM625 and ATIRE BM25. The results are displayed in Table 13.2 where the bold values highlight a $p$-value greater than 0.05 which indicates there is no statistically significant difference between the truncated CBM625 ranking and the exhaustive ATIRE BM25 ranking. The column labelled “Clusters” indicates the first $f$ clusters searched by CBM625. The column labelled “Documents” indicates the average number of documents searched when averaged over all 50 queries. Interestingly, ATIRE produced P@10, P@20 and P@30 scores of 0.43, 0.44 and 0.44 which were higher than those produced by Indri in the experiments by Kulkarni and Callan with all of the scores at 0.3. However, as we have not produced rankings with Indri we are not able to include them in this evaluation. However, the same behaviour is observed, where only a fraction of the collection has to be searched to achieve the same early precision. The previous results of Kulkarni and Callan [2010] clustered the collection into 100 clusters using k-means and solved the scalability problem of clustering 50 million documents by using sampling. Their approach searched the first 1 of 100 clusters to produce a ranking with no statistically significant difference to exhaustive search while searching 1.3% of documents in the collection. Our approach searched the first 8 of 139,641 clusters to produce a ranking with no statistically significant difference to exhaustive search while searching 0.1% of documents in the collection. This is a 13 fold decrease in the average number of documents searched per query!

As with the previous experiments on the Wikipedia we have graphed P@10, P@20 and P@30 when varying the number of clusters. The x axis is the average fraction of the collection searched and the y axis is the fraction of highest P@n score achieved. Figure 7 highlights this for P@10, Figure 8 highlights this for P@20 and Figure 9 highlights this for P@30. As can be seen in the graphs, early precision rises very quickly and only 0.1% of the collection has to be searched to retain early precision.
### Table I. ClueWeb 2009 Category B

<table>
<thead>
<tr>
<th>Approach Clusters</th>
<th>Documents</th>
<th>P@10</th>
<th>P@20</th>
<th>P@30</th>
</tr>
</thead>
<tbody>
<tr>
<td>CBM625 1</td>
<td>15073.7</td>
<td>0.15</td>
<td>0.14</td>
<td>0.13</td>
</tr>
<tr>
<td>CBM625 2</td>
<td>22959</td>
<td>0.28</td>
<td>0.25</td>
<td>0.24</td>
</tr>
<tr>
<td>CBM625 3</td>
<td>30624.5</td>
<td>0.35</td>
<td>0.32</td>
<td>0.31</td>
</tr>
<tr>
<td>CBM625 4</td>
<td>33514.7</td>
<td>0.36</td>
<td>0.34</td>
<td>0.33</td>
</tr>
<tr>
<td>CBM625 5</td>
<td>37340.8</td>
<td>0.38</td>
<td>0.35</td>
<td>0.34</td>
</tr>
<tr>
<td>CBM625 6</td>
<td>41610.4</td>
<td>0.40</td>
<td>0.39</td>
<td>0.37</td>
</tr>
<tr>
<td>CBM625 7</td>
<td>45559.7</td>
<td>0.41</td>
<td>0.40</td>
<td>0.39</td>
</tr>
<tr>
<td>CBM625 8</td>
<td>49359.7</td>
<td>0.45</td>
<td>0.42</td>
<td>0.41</td>
</tr>
<tr>
<td>CBM625 9</td>
<td>54047.4</td>
<td>0.44</td>
<td>0.42</td>
<td>0.41</td>
</tr>
<tr>
<td>CBM625 10</td>
<td>58189.7</td>
<td>0.45</td>
<td>0.43</td>
<td>0.42</td>
</tr>
<tr>
<td>CBM625 20</td>
<td>88368.9</td>
<td>0.43</td>
<td>0.43</td>
<td>0.42</td>
</tr>
<tr>
<td>CBM625 30</td>
<td>118113</td>
<td>0.42</td>
<td>0.42</td>
<td>0.42</td>
</tr>
<tr>
<td>CBM625 40</td>
<td>142540</td>
<td>0.44</td>
<td>0.43</td>
<td>0.42</td>
</tr>
<tr>
<td>CBM625 50</td>
<td>166920</td>
<td>0.43</td>
<td>0.43</td>
<td>0.42</td>
</tr>
<tr>
<td>CBM625 60</td>
<td>188790</td>
<td>0.43</td>
<td>0.44</td>
<td>0.43</td>
</tr>
<tr>
<td>CBM625 70</td>
<td>207586</td>
<td>0.44</td>
<td>0.43</td>
<td>0.43</td>
</tr>
<tr>
<td>CBM625 80</td>
<td>229300</td>
<td>0.44</td>
<td>0.44</td>
<td>0.43</td>
</tr>
<tr>
<td>CBM625 90</td>
<td>247519</td>
<td>0.44</td>
<td>0.43</td>
<td>0.43</td>
</tr>
<tr>
<td>CBM625 100</td>
<td>266513</td>
<td>0.42</td>
<td>0.44</td>
<td>0.44</td>
</tr>
<tr>
<td>CBM625 200</td>
<td>436066</td>
<td>0.44</td>
<td>0.42</td>
<td>0.42</td>
</tr>
<tr>
<td>CBM625 300</td>
<td>587127</td>
<td>0.43</td>
<td>0.42</td>
<td>0.43</td>
</tr>
<tr>
<td>CBM625 400</td>
<td>733771</td>
<td>0.44</td>
<td>0.44</td>
<td>0.43</td>
</tr>
<tr>
<td>CBM625 500</td>
<td>874275</td>
<td>0.45</td>
<td>0.44</td>
<td>0.43</td>
</tr>
<tr>
<td>CBM625 600</td>
<td>1.01E+06</td>
<td>0.42</td>
<td>0.43</td>
<td>0.43</td>
</tr>
<tr>
<td>CBM625 700</td>
<td>1.14E+06</td>
<td>0.41</td>
<td>0.42</td>
<td>0.41</td>
</tr>
<tr>
<td>CBM625 800</td>
<td>1.27E+06</td>
<td>0.41</td>
<td>0.43</td>
<td>0.41</td>
</tr>
<tr>
<td>CBM625 900</td>
<td>1.40E+06</td>
<td>0.42</td>
<td>0.42</td>
<td>0.40</td>
</tr>
<tr>
<td>CBM625 1000</td>
<td>1.52E+06</td>
<td>0.41</td>
<td>0.41</td>
<td>0.40</td>
</tr>
<tr>
<td>ATIRE BM25</td>
<td>NA</td>
<td>50220423</td>
<td>0.43</td>
<td>0.44</td>
</tr>
</tbody>
</table>

![Figure 7. ClueWeb 2009 Category B – P@10](image-url)
14. DISCUSSION
In this paper, we have presented theoretical evaluation strategies for clustering based on the upper bound performance of an “oracle” collection selection approach. The results from this evaluation at INEX indicated actual retrieval systems can take advantage of a more fine grained clustering to better exploit the cluster hypothesis to reduce the number of documents that need to be searched for a retrieval system. This has been demonstrated on two collections, the 2.67 million document INEX 2009 XML Wikipedia collection and the 50 million document ClueWeb09 Category B collection.
Both were evaluated using relevance judgments from human assessors using reusable information retrieval experiments from INEX and TREC. The TopSig K-tree solved the scalability problem of clustering 50 million documents in 10 hours, which to as far as we know, is not reported in the literature in either clustering a collection this large without sampling, or clustering it into such a large number of clusters, or simply clustering all 50 million documents. The CBM625 approach allowed us to rank clusters effectively and efficiently.

We have also introduced using ad hoc relevance judgments as an alternative to class or category based evaluation of document clustering. The primary advantage of this approach comes from pooling used in information retrieval system evaluation that reduces assessor load. Only several thousand documents need to be assessed, whereas the entire collection is labelled for category based evaluations. Furthermore, this ad hoc retrieval based evaluation is motivated by a real use case. It minimises the resources used in a retrieval system. We have confirmed the theoretical “oracle” results of the evaluation at INEX by implementing a cluster based search engine.

We have been able to replicate the results of cluster based search on large scale document collections presented by Kulkarni and Callan [2010]. We confirmed that cluster based search indeed works using different indexing, clustering and ranking approaches. Furthermore, we were able to decrease the number of documents searched by the retrieval system by 13 times.

Standard compression [Trotman 2003] and impact ordering [Anh et al. 2001] of posting lists for both the cluster, $C_{t \times k}$, and document, $D_{t \times n}$, inverted indexes can be implemented to further improve efficiency of retrieval. As the cluster scores in the ranked list of $RSV_c$ values is much more heavily skewed towards the head of the list than when using $RSV_d$ values, we suggest that stopping early via impact ordering will be highly effective for CBM625. Impact ordering with CBM625 may be able terminate earlier than impact order with BM25 relative to length of the postings list. Most weight in the cluster inverted index for a given query is contained in a few clusters. This is also confirmed by earlier experiments where only the 8 most highly ranked clusters of the approximately 140,000 clusters for ClueWeb09 Category B need to be searched to achieve early precision with no statistically significant difference from exhaustive ranking using BM25. Additionally, shard cutoff estimation presented by Kulkarni et al. [2012] as an extension to the original approaches in their cluster based search paper [Kulkarni and Callan 2010] can be implemented to further improve efficiency. The shard cutoff estimations improve the results by another 50% where as our approach improves results by 1300%. However, this shard cutoff estimation is likely to further improve results. We suggest as the clusters are smaller and finer grained, shard cutoff estimation may be able to work more effectively as it is not possible to stop half way through larger clusters.

We suggest that most of the performance improvement we have observed is contributed by the fine grain clustering produced by TopSig K-tree. With sampling approaches such as those used by Kulkarni et al. [2012], there are simply not enough documents available to provide adequate statistics to produce a clustering with this many clusters. It is the scalability of TopSig K-tree that allows us to produce these results. The effect of the number of clusters can be seen in Figure 2 where increasing the number of clusters from 100 to 1559 on the Wikipedia collection allows clusters containing relevant documents to be ranked much earlier in the CBM625 approach.

Apart from TopSig K-tree there are other factors different in our approach that may have contributed to the improved results. The use of the TopSig document signatures themselves may provide different clusters to other approaches by using the log likelihood weighting described by Geva and De Vries [2011]. The effectiveness of this weighting for document clustering has not been studied in detail. The ranking process,
CBM625, may have contributed to improved results. In Figures 4 to 5 we demonstrated that it increases early precision in highly ranked clusters on the Wikipedia collection. Additionally, the use of ATIRE rather than Indri may have contributed to the results. However, the approach we have described is able to drastically decrease the number of documents required to be searched compared to that of by Kulkarni and Callan [2010].

Additionally, the TopSig K-tree has other unique properties that make it particularly appealing for use with clustering of document collections for information retrieval. Document collections are rarely static in practice and have continual updates. The K-tree algorithm allows insertions and deletions with the tree in an online and dynamic fashion. New documents can be inserted without having to perform global analysis. The average case analysis of K-tree [De Vries 2010a] has an $O(\log n)$ time complexity for insertion of a single document. These properties allow it to efficiently cluster volatile document collections. Furthermore, which has not been discussed in this paper at all, is that the tree structure of the K-tree can be used to further increase search efficiency by recursively searching the clusters in the tree using the CBM625 approach. Alternatively, a signature based representation may be able to be found that is effective for ranking clusters in collection selection, eliminating the need for inverted indexes all together. K-tree provides a multi-granular view of clustering where the root node contains the most coarse and largest clusters. As the tree is traversed the clusters become more and more fine grained at lower levels. Furthermore, locating an entire sub-tree on a single machine allows semantically similar documents to be placed together and are likely to be relevant to similar queries due to the cluster hypothesis. In a distributed system, the higher levels of a K-tree may partition the collection into many separate machines. Then each sub-tree stored on each machine can still be used to increase search efficiency on each machine by exploiting the fine grained clusters at lower levels in the K-tree. This improves throughput by two means. Firstly it improves throughput by spreading the load over many machines. Secondly it improves throughput by increasing the efficiency on each machine in the parallel and distributed system.

REFERENCES


A:36

C.M. De Vries et al.


164


The EM-tree Algorithm

I contributed the majority of this paper. I proposed and wrote the algorithm and convergence proof and Lance and Shlomo aided this process. Lance made significant contributions to the implementation of the software\footnote{https://github.com/cmdevries/LMW-tree}. I wrote the manuscript, wrote the proof, wrote software, designed and conducted experiments, and performed data analysis.

16.1 Algorithms

A new algorithm called the EM-tree algorithm is introduced in this paper. It iteratively optimizes an entire $m$-way nearest neighbor search tree. It is proven to converge. Furthermore, it provides efficiency advantages over the TopSig K-tree when using TopSig bit vectors because means are updated less often.
The EM-tree Algorithm

Christopher M. De Vries  
*Electrical Engineering and Computer Science*  
*Queensland University of Technology*  
*Brisbane, Australia*

Lance De Vine  
*Electrical Engineering and Computer Science*  
*Queensland University of Technology*  
*Brisbane, Australia*

Shlomo Geva  
*Electrical Engineering and Computer Science*  
*Queensland University of Technology*  
*Brisbane, Australia*

Editor:

**Abstract**

This paper introduces the EM-tree algorithm. The algorithm iteratively optimizes the m-way nearest neighbor search tree in an Expectation Maximization like fashion. Both the data structure and algorithm are formally defined. A convergence proof for the algorithm is presented. The algorithm is validated experimentally using large scale document collections. The documents are indexed using the TopSig approach where documents are mapped onto binary vectors for efficiency in processing. The EM-tree produces lower distortion clustering solutions in less time than previous large scale clustering approaches. The algorithm also has advantages over other clustering algorithms when used in the streaming setting where document vectors are streamed sequentially from disk. Furthermore, due to the nature of the optimization process, the EM-tree is particularly amenable to parallel and distributed implementations for large scale learning.

**Keywords:** Clustering, Large Scale Learning, Random Projection, Document Signatures, Document Clustering

1. Introduction

This paper introduces the EM-tree algorithm. It is a tree structured clustering algorithm that optimizes a m-way nearest neighbor search tree in an Expectation Maximization like fashion. The structure of the tree adapts to the data with a pruning process that removes empty branches of the tree between each iteration. The data is assigned to leaves in the tree according to the current Expectation of the parameters. After the Expectation step, any empty branches are pruned from the tree. The parameters are then Maximized given the current assignment. The EM-tree algorithm is also similar to the k-means algorithm which is well known to be a special case of the Expectation Maximization algorithm (Dempster et al., 1977) with hard cluster assignment and isotropic Gaussian distributions. What differentiates EM-tree from previous tree structured clustering algorithms is that the entire
tree is optimized at each iteration which is followed by a pruning process that removes empty branches.

Several different clustering algorithms can store their data in a $m$-way tree. The k-means algorithm (Lloyd, 1982) can be constructed as a two level $m$-way tree. If the intermediate levels of the tree are ignored in the $m$-way tree, then the EM-tree algorithm becomes exactly the same as k-means at the root level. We show this formally later in the paper and it becomes the basis for a convergence proof of the EM-tree algorithm. Other hierarchical clustering algorithms that work with vectorial data such as TSVQ (Gersho and Gray, 1993), agglomerative hierarchical clustering (Steinbach et al., 2000) and K-tree (Geva, 2000; De Vries et al., 2009) \(^1\) produce $m$-way tree structures as well.

The EM-tree algorithm was first devised to improve point assignment to leaves in the K-tree algorithm. Early in the construction process, the K-tree makes decisions about where to places points within the tree without entire knowledge of the data set as vectors are presented one by one. After some time it became apparent to the authors that the EM-tree algorithm could be applied to any $m$-way tree, including a randomly initialized tree. At this point the algorithm became totally independent from K-tree. Initially it was not clear whether the algorithm would converge or not. After some thought it became apparent that k-means is in fact happening in the root node of the tree. This lead to the inductive proof in Theorem 4 demonstrating that the algorithm will eventually converge.

Firstly, in Sections 2 and 3, we introduce the k-means problem and the k-means algorithm that finds local approximations of the globally optimal solution. We then introduce and define the $m$-way nearest neighbor search tree data structure in Section 4. Section 5 introduces and defines the EM-tree algorithm. A convergence proof for the EM-tree algorithm is contained in Section 6. The TSVQ algorithm is defined in Section 7 and it is shown that it is not exactly equivalent to the EM-tree algorithm.

The paper then changes from a theoretical focus to perform experiments in document clustering to provide support for the theoretical claims. The experimental setup for the remaining sections of the paper is described in Section 8. Section 9 evaluates the convergence properties of the EM-tree algorithm. In Section 10, the EM-tree is compared to the K-tree, TSVQ and k-means algorithms. The EM-tree algorithm is compared to the K-tree algorithm on a large scale document collection containing 50 million documents in Section 11. A streaming variant of the EM-tree is discussed in Section 12 and how this provides advantages over TSVQ. The implications of parallel and distributed versions of the EM-tree are discussed in Section 13.

Finally, future work is discussed in Section 14 and the paper is concluded in Section 15.

2. The k-means problem

The k-means problem is a fundamental problem in geometry that minimizes the squared Euclidean distance between $k$ centroids and a set of points $X$. What follows is a formal definition of the problem.

\(^1\) http://ktree.sf.net
Let $C$ be a set of $k$ centroids in $d$ dimensions, $\{c_1, \ldots, c_k\}$, where, $C \subset \mathbb{R}^d$, and, $|C| = k$. Let $s$ be the squared Euclidean distance between two points,

$$s(x, y) = ||x - y||^2,$$

and, $x, y \in \mathbb{R}^d$. The nearest function produces the nearest centroid, $c \in C$, for point, $x \in X$, for the squared Euclidean distance function, $s$. Note that ties are broken in a consistent manner by taking the first member of the set if there are two or more centroids at equal similarity, hence the 1 subscript,

$$\text{nearest}(x, C) = \{c : \exists c \in C \wedge \forall d \in C \wedge s(c, x) \leq s(d, x) \wedge c \neq d\}_1.$$

The function neighbors produces the nearest neighbor set for a centroid, $c \in C$, where,

$$\text{neighbors}(c, X, C) = \{x \in X : \text{nearest}(x, C) = c\}.$$

The k-means problem is to find the set $C$ that minimizes the squared Euclidean distance, $s$, between each point in, $X$, and its nearest centroid in $C$, where,

$$C = \text{argmin}_C \sum_{c \in C} \sum_{x \in \text{neighbors}(c, X, C)} s(c, x).$$

Finding the globally optimal solution to the k-means problem is known to be NP-hard (Aloise et al., 2009). Therefore, the k-means algorithm (Lloyd, 1982) is used to approximate the optimal solution by converging to a local optima (Selim and Ismail, 1984).

3. The k-means algorithm

The k-means algorithm is one of the most popular unsupervised learning algorithms due to its simplicity, effectiveness and linear time complexity when the number of iterations is limited. However, it was only recently proven that the k-means algorithm has polynomial time complexity when the number iterations is not limited using a smoothed analysis (Arthur et al., 2009). It has been listed as one of the top 10 algorithms in data mining (Wu et al., 2008) and is used in many different applications.

It is often used to initialize more expensive algorithms such as Gaussian mixtures, Radial Basis Functions, Learning Vector Quantization and some Hidden Markov Models. Bottou and Bengio (1995) suggest that k-means performs most of the optimization in a small fraction of the time when it seeds more expensive algorithms.

The k-means algorithm has been defined in pseudocode in the procedure kmeans. The algorithm can be cast as gradient descent, probabilistic Expectation Maximization, and Newton’s method (Bottou and Bengio, 1995).

kmeans($k, X$)
1 \hspace{1em} select $k$ points uniformly at random from $X$ to initialize the centroid set $C$
2 \hspace{1em} while $C$ has changed between iterations
3 \hspace{2em} assign each vector in $X$ to its nearest neighbor in the centroid set $C$
4 \hspace{2em} recalculate each centroid in $C$ as the mean of all its nearest neighbors
5 \hspace{1em} return $C$
At each iteration of k-means the centroid set is updated. What follows is a formal
definition of the process. This will be used for later proofs. The function \textit{update}
produces an updated centroid set that is created at each iteration of k-means,

\[ \text{update}(C, X) = \{ c \in C : \text{mean}(\text{neighbors}(c, X, C)) \}. \]

4. \textit{m-way nearest neighbor search tree}

A \textit{m-way nearest neighbor search tree} is a recursive data structure that indexes a set of \( n \) points in \( d \) dimensions where, \( X = \{ x_1, \ldots, x_n \}, X \subset \mathbb{R}^d \) and \(|X| = n\). From here on out in
the text it will be referred to as a \( m \)-way tree which is not to be confused with the \( m \)-way
tree from the B-tree data structure. Let \( N \) be the set of all tree nodes. Each node is a
sequence of records which are (vector, child node) pairs. For example, if \( \text{root} \) is the root
node of a tree containing 4 records,

\[ \text{root} = \langle \text{record}_1, \text{record}_2, \text{record}_3, \text{record}_4 \rangle, \]

then, \( \text{root}_1 \), returns the first pair in the sequence, \( \text{record}_1 \), and, \( \text{key}(\text{root}_1) \), returns the
vector that is the key and, \( \text{child}(\text{root}_1) \), returns the child node associated with the key of
\( \text{record}_1 \),

\[ \text{root}_1 = \text{record}_1 = (\text{key} \in \mathbb{R}^d, \text{child} \in N). \]

The length of a node is denoted by bars, \(|\text{root}| = 4\). There are internal and leaf nodes in a
\( m \)-way tree. Leaf nodes, \( L \), contain the data inserted into the tree with empty child nodes
where, \( L \subset N \), and,

\[ \forall \text{leaf} \in L : \forall \text{record} \in \text{leaf} : \text{key}(\text{record}) \in X \land |\text{child}(\text{record})| = 0 \land 1 \leq |\text{record}| \leq |X|. \]

Note that leaf nodes are not restricted by the tree order and can contain the entire dataset
but must not be empty. Internal nodes, \( I \), have non-empty child nodes and contain between
1 and \( m \) records where, \( I \subset N \), and,

\[ \forall \text{internal} \in I : \forall \text{record} \in \text{internal} : |\text{child}(\text{record})| \neq 0 \land 1 \leq |\text{record}| \leq m. \]

The \textit{leaves} function takes any node and returns all data points associated with that node,
i.e., all descendant points in leaf nodes, where, \( \text{leaves}(\text{node}) \subset X \),

\[ \text{leaves}(\text{node}) = \{ \text{point} : \forall \text{record} \in \text{node} : \begin{cases} \text{point} = \text{key}(\text{record}) \quad \text{if node} \in L \\ \text{leaves}(\text{child}(\text{record})) \quad \text{if node} \in I \end{cases} \}. \]

All internal nodes contain keys as means of points in \( X \), where,

\[ \forall \text{internal} \in I : \forall \text{record} \in \text{internal} : \text{key}(\text{record}) = \text{mean}(\text{leaves}(\text{child}(\text{record}))). \]

Figure 1 illustrates how each key in an internal node is the mean of all keys in descendant
leaf nodes. Figure 2 illustrates how k-means can be constructed as a two level \( m \)-way tree,
where the centroids are in the root of the tree, and the data points being clustered are in
the leaves.
The EM-tree Algorithm

Figure 1: A $m$-way tree illustrating means

Each mean is the mean of all vectors in descendant leaf nodes.

Figure 2: A two level $m$-way tree illustrating the structure of the k-means algorithm

The k means or centroid set.

The $n$ points being clustered.

Figure 2: A two level $m$-way tree illustrating the structure of the k-means algorithm
5. The EM-tree algorithm

The EM-tree algorithm proceeds in a similar manner to most iterative optimization algorithms. The \( m \)-way tree model is initialized and it is then iteratively optimized until convergence. However, the EM-tree algorithm has an additional step during the iterative optimization process that removes sections of the model that no longer have data associated with them. This is the pruning process that removes empty branches from the \( m \)-way tree. Empty branches may result from the re-insertion step which is described below. By a process of insertion (Expectation), update (Maximization) and pruning, the \( m \)-way tree model adapts to the data as the clusters converge.

The EM-tree places further restrictions on the definition of the \( m \)-way tree. The points contained in each child tree associated with each key are the nearest neighbors of the associated key,

\[
\forall \text{internal} \in I : \forall \text{record} \in \text{internal} : \text{leaves} (\text{child} (\text{record})) = \text{neighbors} (\text{key} (\text{record}), Y, C).
\]

This invariant above is broken during the optimization process but holds once the tree has converged. The optimization process in EM-tree assigns points to leaves using the nearest neighbor rule during the Expectation step. However, the updating of means in the tree in the Maximization breaks the nearest neighbor assignment, and points must be reassigned to the new means in the tree.

The EM-tree algorithm can take any \( m \)-way tree as input and optimize it. For example, the tree produced by the K-tree algorithm can be taken and further optimized to correct decisions that were made without full knowledge of the entire data set. Additionally, the algorithm can be applied to any sub tree in a \( m \)-way tree. In a setting where a changing data set is being clustered, branches of the trees effected by insertions and deletions can be restructured to the data independently of the rest of the tree. Another possible use case is that the tree is built using a sufficient sample of a large collection to run to convergence. The remaining data can be inserted into the tree. It can either be left as is or a small number of iterations can adapt the tree to the remaining data.

The procedure \text{SEED} initializes the EM-tree algorithm where \( m \) is the order of the tree, \textit{node} is the current tree node, \( Y \) is the set of points to choose centroids from where, \( Y \subset X \), and \textit{depth} is the number of levels deep the tree will be. Note that this seeding procedure produces a height balanced tree as all leaves are at the same depth. However, it is possible to use other criteria for stopping the recursion such as the number of points in \( Y \). However, this does not guarantee a height balanced tree.
The EM-tree Algorithm

\textbf{SEED}(m, depth, Y)

\begin{verbatim}
  node = ∅
  if depth == 1
    for point ∈ Y
      append (point, ∅) to node
    return node
  else
    C is the centroid set
    C = m points selected uniformly at random from Y
    for centroid ∈ C
      neighbors = neighbors(centroid, Y, C)
      record = (mean(neighbors), SEED(m, depth − 1, neighbors))
      append record to node
    return node
\end{verbatim}

The \textbf{INSERT} procedure inserts a set of vectors, Y, into a height balanced m-way tree who’s root is, \textit{node}, removing all existing points it indexes. Points are inserted by following the nearest neighbor search path, where at each node in the tree, the branch with the nearest key is followed.

\textbf{INSERT}(node, Y)

\begin{verbatim}
  if |child(node)| == 0
    insert all points in Y into this leaf node
    node = ∅
    for point ∈ Y
      append (point, ∅) to node
  else
    not a leaf node so keep recursing and splitting Y according to nearest keys
    C = {} 
    for record ∈ node
      insert key(record) into C 
      for record ∈ node
        INSERT(child(record), neighbors(key(record), Y, C))
\end{verbatim}

The procedure \textbf{UPDATE} updates the means in the tree according to the current assignment of data points in the leaves.

\textbf{UPDATE}(node)

\begin{verbatim}
  if |child(node)| == 0
    return
  else
    for record ∈ node
      update(child(record))
    key(record) = mean(leaves(child(record)))
\end{verbatim}
The prune procedure removes any means with no associated data points from the tree. This is completed bottom up, where above leaf means are removed first. If any higher level leaves contain no means, then they are removed as well in the next pass. Once the m-way tree has been updated by the UPDATE procedure, and the data points have been reinserted via INSERT procedure, branches of the tree may contain no points. These empty branches are removed, and this allows the tree structure to adapt to the data.

prune(node, depth)
1  // means start one level higher than depth
2  depth = depth - 1
3 while depth ≠ 0
4  prunelevel(node, depth)
5  depth = depth - 1

prunelevel(node, depth)
1  if depth == 1
2  for record ∈ node
3    if |child(record)| == 0
4      remove record from node
5  else
6    for record ∈ node
7      prunelevel(child(record), depth - 1)

The emtree procedure initializes and iteratively optimizes an EM-tree of order m that is depth levels deep.

emtree(m, depth, X)
1  root = seed(m, depth, X)
2  converged = false
3 while not converged
4  root' = root
5  insert(root', X)
6  prune(root')
7  update(root')
8  if root == root'
9    converged = true
10  else
11    root = root'
12 return root

6. Convergence proof
The proofs outlined in this section demonstrate that the EM-tree algorithm will converge. They show that the k-means algorithm is being performed at the root node of the tree, and then at successively lower internal nodes of the tree after the nodes above have stabilized.
The EM-tree Algorithm

Until the parent nodes have stabilized, the input vectors of a given node can change between iterations. The k-means update procedure does happen for non-root nodes, but it is not exactly the same as the k-means algorithm because the input vectors being optimized can change between iterations. However, at some point, the input vectors for each node do stabilize. From this point on-wards, the optimization process for any tree node becomes exactly the same as the k-means algorithm seeded with the means at the point the input vectors stabilize.

**Definition 1** Internal nodes, $I$, in an EM-tree index a subset of all points in $X$, $\forall node \in I : leaves(node) \subset X$, where the root node indexes all points in $X$. The subset indexed by a particular node, $Y$, is a subset of all points, $Y \subset X$. The centroid set, $C$, of an internal node, is all the vectors contained in the keys of the internal node,

$$C = \{ \forall record \in internal : key(record) \}.$$  

Each internal node is a set of cluster means. The points contained in each child tree are the nearest neighbors of each point in the centroid set,

$$\forall internal \in I : \forall record \in internal : leaves(child(record)) = neighbors(key(record), Y, C).$$  

(1)

All the keys are means of the leaves contained in the associated sub tree,

$$\forall internal \in I : \forall record \in internal : key(record) = mean(leaves(child(record))).$$  

(2)

**Lemma 2** At each iteration of the EM-tree algorithm the k-means Expectation and Maximization step, $update(C,Y) = \{ c \in C : mean(neighbors(c,Y,C)) \}$, is performed at each internal node in the $m$-way tree, $\forall internal \in I : \{ c \in C : mean(neighbors(c,Y,C)) \}$.

**Proof** By Definition 1, the means in the EM-tree are updated in the same manner as k-means. The leaves in a child node are nearest neighbors of the corresponding key in the node, which are cluster means.

$$\forall internal \in I : \forall record \in internal : key(record) = mean(leaves(child(record)))$$

By substition of Equation 1 into Equation 2.

$$\forall internal \in I : \forall record \in internal : key(record) = mean(neighbors(key(record), Y, C))$$  

(3)

Each key contained in a given internal node has a one to one correspondence with the centroids of the k-means algorithm as in Definition 1, $C = \{ \forall record \in internal : key(record) \}$. Therefore, by subsitution of $key(record)$ for $c$ in Equation 3, it is reduced to the following.

$$\forall internal \in I : \{ c \in C : mean(neighbors(c,Y,C)) \}$$

This takes exactly the same form as the $update(C,Y)$ function that performs the update of k-means. All internal nodes perform $update(C,Y)$ and therefore the same operations as k-means.  


Note that Lemma 2 does not prove the equivalence of the EM-tree algorithm to the entire k-means algorithm, only the update procedure applied at each iteration. For complete equivalence to the k-means algorithm, the subset of data indexed by a given node, $Y$, has to be fixed between each iteration. This is not the case for all internal nodes and is proven later.

**Definition 3** The nodes $N$ in a $m$-way tree change between iterations of the EM-tree algorithm. Therefore, $N$ represents the nodes at the current iteration and $N'$ at the previous iteration.

**Theorem 4** The points contained in any node, $n \in N$, in the $m$-way tree will converge when using the EM-tree algorithm. In a finite amount of time the following will hold, $N' = N$.

**Proof** By structural induction on the definition of $n \in N$. The induction hypothesis is that after a finite amount of time

$$P(N) ::= N' = N.$$ 

**Base case** (k-means is performed in the root node):
It has previously been proven that the k-means algorithm converges (Selim and Ismail, 1984). Lemma 2 states that the k-means update procedure is performed at any node in the tree. It says nothing about the set of points being included in the optimization process at each iteration. By Definition 1, the EM-tree indexes a fixed set of points, $X$, so the root node is guaranteed to converge as it indexes these points over all iterations. This set of points, $X$, being indexed will never change between iterations. The root node is performing k-means on the entire data set, $X$, until convergence by repeated application of the \textit{update}$(C,Y)$ function. The set of vectors indexed by the child trees of the root node will stabilize by the convergence of k-means happening in the root node. Therefore, the set of indexed vectors for each child tree, $Y \subset X$, will no longer change between iterations.

**Constructor Case:**
Assume the induction hypothesis that any node, $n \in N$, in the tree converges so that the keys in $n$ are fixed and the nearest neighbors assigned to the child trees becomes fixed. Consider the set of all the children of node, $n$, in the tree, $N^{+1} = \{\forall \text{record} \in n : \text{child} (\text{record})\}$. The set of data points indexed by each child in this set, $\forall \text{child} \in N^{+1} : \text{leaves} (\text{child})$, is now fixed and does not change between iterations of the EM-tree algorithm. Lemma 2 proves that any tree node performs the k-means operations and the node will converge once the set of input points has become fixed. This means all child nodes, $N^{+1}$, are also guaranteed to converge.

This proves that $P(N)$ holds as required for the constructor case. By structural induction we conclude that $P(N)$ holds for all nodes, $n \in N$. \[\square\]
7. Tree Structured Vector Quantization

TSVQ (Gersho and Gray, 1993) applies k-means to an entire data set until convergence splitting it into \( m \) clusters. It then applies k-means recursively on each subset. This recursively splits the data into \( m \) clusters until a stopping criteria has been met for the recursive application of k-means to stop. Like the EM-tree, this process also forms a \( m \)-way tree. As Lemma 2 indicates, the k-means update rule is being applied at each iteration of the optimization process in the EM-tree algorithm. Therefore, it is easy to incorrectly conclude that the EM-tree algorithm is the same as TSVQ. Firstly, EM-tree has a pruning process that can remove clusters contained in internal nodes. However, this can also happen with TSVQ, as k-means can have cluster centroids having no data points associated with them. In this case, these clusters are usually dropped from the solution, and therefore in the TSVQ algorithm not all splits are guaranteed to be \( m \)-way.

```plaintext
TSVQ(m, depth, Y)
1  node =<>  
2  if depth == 1  
3      for point ∈ Y  
4          append (point, <> ) to node  
5      return node  
6  else  
7      C is the centroid set  
8      C = KMEANS(m, Y)  
9      for centroid ∈ C  
10         neighbors = neighbors(centroid, Y, C)  
11         record = (centroid, TSVQ(m, depth - 1, neighbors))  
12         append record to node  
13      return node
```

**Theorem 5** The EM-tree algorithm is not equivalent to the TSVQ algorithm.

**Proof** The only node guaranteed to have the same input set of vectors to optimize at each iteration of the EM-tree algorithm is the root node, which has all vectors in the data set, \( X \), as input. The set of points for non-root nodes, \( Y \), can change between iterations because the clusters defined in nodes higher in the tree have not yet converged. This is not the same as k-means performed in TSVQ. No recursive relationships between clusters are created until k-means has completely converged in the TSVQ algorithm. The initial point set for k-means used in TSVQ is always fixed and does not change between iterations.

The observation to be made is that the basic difference between TSVQ and EM-tree is that TSVQ optimizes level by level, while EM-Tree optimizes the entire tree in each iteration. This results in a further difference between the two algorithms regarding the required number of passes over the data which is discussed in Section 12.
8. Experimental Setup

The following sections of this paper test the proposed theoretical results by measuring the properties of the EM-tree algorithm by clustering document collections.

We have used the INEX 2010 XML Mining collection (De Vries et al., 2011) which is a 144,265 document subset of the INEX 2009 Wikipedia collection. The INEX Wikipedia has been used for the evaluation of ad hoc retrieval in the INEX evaluation forum (Arvola et al., 2011). The 2010 XML Mining subset contains 144,265 documents determined by the reference search run from the ad hoc track. These documents were used for evaluation of the clustering and classification tasks in the INEX 2010 XML Mining track. INEX is a collaborative forum where researchers from different organizations take different approaches to the same task.

For larger scale clustering we have used the the ClueWeb09 Category B collection consisting of 50 million documents and a vocabulary of 96.1 million terms. The ClueWeb09 Category B collection was introduced for evaluation at the TREC 2009 Web Track (Clarke et al., 2009). It is a subset of the 1 billion document ClueWeb09 collection.

Each of the documents have been represented using bit vectors produced by TopSig. TopSig (Geva and De Vries, 2011) creates dense binary document vectors using random projections and numeric quantization. The bit vectors are compared using the Hamming distance measure. This TopSig paper introduced a variant of k-means that works directly with the binary representations produced by TopSig and has been shown to lead to a one to two order magnitude increase in efficiency over traditional sparse vector approaches to document clustering. All centroids are also bit vectors where the centroid is calculated by taking the most frequently occurring bit in each dimension. If there are more bits set in the given dimension for vectors associated with the centroid, then it is also set in the centroid. In the TopSig model, the 0 and 1 values represent -1 and +1 values respectively. Therefore, this is also equivalent to accumulating all the vectors and taking their sign, to map the values back to -1 and +1. The TopSig search engine 2 was used to produce 4096-bit vectors for each document. The signature density was set to 21 and the log likelihood document weighting (Geva and De Vries, 2011) was used. For both document collections, the term statistics from the Wikipedia were used for the log likelihood document weighting. When indexing the ClueWeb collection, terms not contained in the Wikipedia are assumed to only appear in the document in which they occur. This allows terms without statistics in the Wikipedia to be used for the log likelihood weighting function.

The signatures produced by TopSig and the software used to create them are available from the K-tree SourceForge page 3. The implementations of the K-tree, k-means, TSVQ and EM-tree algorithms used in this paper are available from the LMW-tree GitHub repository 4.

All experiments have been conducted on a system with 2 × 8 core 2GHz AMD Opteron 6128 CPUs with a total of 64GB of main memory. All experiments were single threaded.
9. Convergence properties

We have tested the convergence properties of the EM-tree algorithm in comparison to TSVQ, as it is conceptually quite similar but not exactly the same algorithm. We used the 144,265 document INEX 2010 XML Mining collection described in the previous section. Document were represented as 4096-bit vectors. Both algorithms were used to create an order 10 tree with a depth of 3. There are therefore two internal levels of the tree with the first level containing 10 clusters and the second 100. Occasionally, the EM-tree would prune 1 cluster from the second level, resulting in 99 clusters. The number of iterations of optimization in the TSVQ algorithm is controlled by limiting k-means to the desired number of iterations. EM-tree proceeds by iteratively optimizing the entire tree and the error can be measured after each iteration.

Figure 3 highlights the convergence properties of the two algorithms as the number of iterations of optimization is increased along the x-axis. Each algorithm has been executed 20 times at each iteration limit with a different random starting condition. The distortion in terms of RMSE was measured at the most fined grained level, which contains 100 clusters. Iteration 0 represents the state when no centroids have been updated. This is the distortion of using randomly selected points as centroids where there was been no optimization performed. The nearest neighbors are associated with the randomly selected points from the data set. Iteration 1 is the state after the centroids have been updated once, once per level for TSVQ and once for the entire tree for EM-tree. Iteration 2 is after they have been updated twice and so on. The y-axis represents the Root Mean Squared Error (RMSE) between the 100 cluster centroids and the associated TopSig document bit vectors. As the distance measure is the Hamming distance, this represents the expected number of bits difference between a document and its nearest centroid. The graph highlights that
Table 1: INEX 2010 XML Mining Collection – Convergence

<table>
<thead>
<tr>
<th>Iterations</th>
<th>TSVQ Average RMSE</th>
<th>EM-tree Average RMSE</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1957.40 ± 3.03</td>
<td>1957.62 ± 2.28</td>
<td>0.79</td>
</tr>
<tr>
<td>1</td>
<td>1808.60 ± 5.65</td>
<td>1810.07 ± 5.56</td>
<td>0.41</td>
</tr>
<tr>
<td>2</td>
<td>1720.61 ± 4.39</td>
<td>1734.64 ± 3.81</td>
<td>3.93 × 10^{-13}</td>
</tr>
<tr>
<td>3</td>
<td>1689.68 ± 3.04</td>
<td>1712.35 ± 4.72</td>
<td>3.15 × 10^{-20}</td>
</tr>
<tr>
<td>4</td>
<td>1677.99 ± 3.84</td>
<td>1703.09 ± 4.66</td>
<td>1.18 × 10^{-20}</td>
</tr>
<tr>
<td>5</td>
<td>1672.83 ± 1.96</td>
<td>1696.83 ± 3.67</td>
<td>1.11 × 10^{-25}</td>
</tr>
<tr>
<td>6</td>
<td>1671.05 ± 3.07</td>
<td>1697.11 ± 4.68</td>
<td>2.29 × 10^{-22}</td>
</tr>
<tr>
<td>7</td>
<td>1669.95 ± 3.49</td>
<td>1692.51 ± 3.89</td>
<td>3.09 × 10^{-21}</td>
</tr>
<tr>
<td>8</td>
<td>1668.43 ± 2.30</td>
<td>1690.46 ± 6.15</td>
<td>1.48 × 10^{-17}</td>
</tr>
<tr>
<td>9</td>
<td>1667.98 ± 2.56</td>
<td>1687.55 ± 3.21</td>
<td>1.00 × 10^{-22}</td>
</tr>
<tr>
<td>10</td>
<td>1666.97 ± 2.53</td>
<td>1687.33 ± 5.05</td>
<td>1.37 × 10^{-18}</td>
</tr>
<tr>
<td>11</td>
<td>1665.35 ± 2.94</td>
<td>1686.37 ± 4.68</td>
<td>2.31 × 10^{-19}</td>
</tr>
<tr>
<td>12</td>
<td>1666.05 ± 2.88</td>
<td>1683.58 ± 2.41</td>
<td>2.11 × 10^{-22}</td>
</tr>
<tr>
<td>13</td>
<td>1664.97 ± 1.80</td>
<td>1681.41 ± 3.45</td>
<td>6.66 × 10^{-21}</td>
</tr>
<tr>
<td>14</td>
<td>1665.02 ± 2.21</td>
<td>1681.09 ± 4.64</td>
<td>1.53 × 10^{-16}</td>
</tr>
<tr>
<td>15</td>
<td>1663.53 ± 2.18</td>
<td>1679.33 ± 3.76</td>
<td>1.06 × 10^{-18}</td>
</tr>
<tr>
<td>16</td>
<td>1664.40 ± 2.47</td>
<td>1679.38 ± 3.59</td>
<td>7.47 × 10^{-18}</td>
</tr>
<tr>
<td>17</td>
<td>1664.51 ± 3.07</td>
<td>1677.82 ± 2.93</td>
<td>1.38 × 10^{-16}</td>
</tr>
<tr>
<td>18</td>
<td>1663.28 ± 2.20</td>
<td>1677.85 ± 3.54</td>
<td>4.02 × 10^{-18}</td>
</tr>
<tr>
<td>19</td>
<td>1663.57 ± 2.56</td>
<td>1676.30 ± 4.15</td>
<td>4.07 × 10^{-14}</td>
</tr>
<tr>
<td>20</td>
<td>1663.30 ± 2.94</td>
<td>1675.86 ± 3.45</td>
<td>6.41 × 10^{-15}</td>
</tr>
</tbody>
</table>

The EM-tree algorithm has the general behavior of convergence as the number of iterations increases. Note that these algorithms are performing a different number of passes over the data and this is discussed when using the algorithms in a streaming setting in Section 12.

Table 1 contains the RMSE for each algorithm at each iteration. These numbers are the values for the plots in Figure 3. Additionally, there is a p-value for a two tailed t-test between the both algorithms. There is no statistically significant difference between EM-tree and TSVQ for iterations 0 and 1 where p > 0.05. For Iterations 2 and higher, TSVQ converges to lower distortion and therefore statistically significantly better solutions than EM-tree where p < 0.05. We suggest that changing the input vectors between iterations for non-root nodes introduces noise into the optimization process for EM-tree. However, there are differences in the way EM-tree performs optimization and they can lead to better solutions when in a streaming setting or when combined with sampling. These two settings are discussed in Sections 11 and 12.

In Table 1, the RMSE does not monotonically decrease for either algorithm. In the case of TSVQ, when changing the maximum number of iterations, the splits determined in a given level of the tree are different. Therefore, the input vectors to lower levels of the tree are not the same, as the maximum number of iterations for k-means has changed.
Therefore, it is possible to converge to a higher RMSE solution. The same applies to EM-tree as discussed in Theorem 5, as the set of input vectors to any non-root node changes between iterations.

10. Comparison to other Algorithms

We have performed experiments to compare the EM-tree to K-tree, k-means and TSVQ as the number of clusters varies. This allows visualization of the trends of RMSE and runtime as the number of clusters are changed. These experiments used the 144,265 document INEX 2010 XML Mining collection described earlier. All algorithms used 4096-bit TopSig document vectors.

The K-tree algorithm builds a m-way tree as data arrives. The size of tree nodes are controlled by the tree order. Once a node becomes full, it is split in 2 using the k-means algorithm and the 2 centroids are promoted to create a new node. The K-tree was run with tree orders of 1000, 750, 500, 400, 300, 250, 200, 150, 100 and 75 to produce a different number of clusters. Note that as the tree order becomes smaller, K-tree produces more clusters.

The computational advantages of K-tree are no longer apparent when using the TopSig bit vector representations in comparison to TSVQ. When using real valued vectors the K-tree has been shown to be faster than TSVQ (Geva, 2000). The centroids contained in the search path of a vector inserted into the tree are updated upon every insertion. For bit vector representations, this is relatively slow compared to using integer or real valued vectors. For a centroid to be updated, all the associated nearest neighbors must be unpacked from the bit representation and added to an integer valued vector. In contrast, the nearest neighbor comparison using the Hamming distance works directly with the bit vector without unpacking each bit into a larger representation. Therefore, K-tree becomes slower than TSVQ or EM-tree because updating the centroids dominates the computational cost. However, delayed updates of the centroids reduces the cost associated with updating centroids. In these experiments, the delayed update K-tree only has updates along the insertion path of every 1000th vector inserted into the tree. Note that this does not change the computational complexity of the algorithm. However, it reduces large constant overheads associated with updating means when using bit vectors. The delayed update K-tree was run with tree orders of 1000, 750, 500, 400, 300, 250, 185, 138, 85, 65 to approximately match the number of clusters produced by the K-tree with updates for every insertion.

The TSVQ algorithm was used to build trees that are 3 levels deep. The tree orders were varied to match the number of clusters produced by K-tree. The tree orders used were 15, 17, 21, 24, 28, 31, 36, 42, 53 and 63.

The EM-tree algorithm was used to build trees that are 3 levels deep. The tree orders were varied to match the number of clusters produced by K-tree. Note that the tree orders are different to TSVQ because the EM-tree prunes clusters during optimization. The tree orders used were 15, 18, 23, 26, 33, 39, 46, 61, 86 and 107.

The k-means algorithm was used to produce approximately the same number of clusters as K-tree. The number of clusters produced was 230, 310, 470, 590, 810, 1010, 1300, 1800, 2900 and 4000.
All algorithms were limited to 10 iterations of optimization for a fair comparison. This includes the k-means algorithm used in K-tree for splitting nodes. All algorithms were run 20 times with different starting conditions. All values in Figures 4 and 5 are average values for the 20 runs.

Figure 4 displays the RMSE as the number of clusters changes. Only a small difference in RMSE of around 6 bits is required to create a statistically significant different result. This was observed when testing RMSE in the streaming setting in Table 3. For all of the points in Figure 4, the variance of RMSE is low, with a standard deviation between 0.5 and 5 bits for any algorithm. This is the same variance as reported in Table 3. The only exception was with K-tree when producing 4000 clusters with a standard deviation of 10 bits and 16 bits for the delayed udpate K-tree. Therefore, the differences in RMSE are likely to be statistically significantly different in Figure 4 as they are greater than 6 bits.

The RMSE of the K-tree algorithm as the number of clusters increases does not always decrease. This is due to decreasing the tree order to achieve the desired number of clusters. The first increase in RMSE, while also increasing the number of clusters is when the K-tree goes from a 2 level tree to a 3 level tree between 500 and 750 clusters. The deeper tree introduces more inaccuracies to the nearest neighbor search and therefore the error increases even though there are more clusters. The other increase is from 3000 to 4000 clusters where the tree order becomes smaller. We suggest that once the tree order becomes too small, in this case, going from 100 to 75, there are not enough vectors to perform meaningful learning with k-means. In fact, the decrease in distortion as the tree order increases, does not behave like the other algorithms at all. It is much flatter as can be seen in Figure 4. We suggest that this is due to the tree order being decreases further and further, and the effect this has on the ability of k-means to perform meaningful splits of the data.

The K-tree algorithm without delayed updates is not appropriate for clustering TopSig document vectors. This is due to the high runtime due to many updates of centroids. However, the delayed update K-tree addresses this problem and provides the best runtimes, but at the cost of higher RMSE. It also appears that a small tree order adversely effects the K-tree algorithm. We suggest this is because the k-means algorithm does not have enough information to effectively learn how to split tree nodes in two.

For in memory clustering of TopSig document signatures, the TSVQ algorithm appears to be the best choice for a single shot approach to learning when the best trade-off between RMSE and runtime is required. It is almost the fastest algorithm, with only the delayed update K-tree being marginally faster for large numbers of clusters, but the delayed update K-tree has much higher RMSE. The k-means algorithm provides the lowest RMSE solution but has a relatively high computational cost. However, the EM-tree and K-tree algorithms have other advantages. Both support incremental clustering of the data. In the case of the K-tree, it is incrementally updated for every vector inserted into the tree. However, the EM-tree algorithm can take any m-way tree as input and optimize it. Therefore, every now and then 1 or 2 iterations of the algorithm can be run to adapt the tree to new data inserted. TSVQ does not have any of these properties. Once the tree is built, it is final. Vectors can be inserted into the m-way tree produced by TSVQ but the algorithm has no means to update it. However, a m-way tree can be built using a sample of the data and TSVQ. Then the EM-tree can be used to perform 1 or 2 iterations of optimization of the whole data set, based on the better optimized tree found by TSVQ. This is addressed in Section
The EM-tree Algorithm

11 when clustering large scale document collections of 50 million documents. EM-tree has advantages over TSVQ in the streaming setting and this is described in Section 12.
Table 2: ClueWeb 2009 Category B – 104,000 Clusters

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Clusters</th>
<th>RMSE</th>
<th>Runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>K-tree with delayed updates</td>
<td>103935</td>
<td>1296.76</td>
<td>16 hours</td>
</tr>
<tr>
<td>EM-tree initialized with sampled TSVQ</td>
<td>104021</td>
<td>1242.38</td>
<td>9.5 hours</td>
</tr>
</tbody>
</table>

11. Comparison to K-tree on ClueWeb

All of the previous experiments have been conducted on the relatively small 144,265 document INEX 2010 XML Mining collection. This was to enable reasonable running times with multiple runs of the algorithms and matching the number of the clusters by trying different tree orders. Clustering the 50 million document ClueWeb collection can take upwards of 10 hours. Therefore, the experiments in this section contain only single runs of the algorithms.

The TopSig K-tree with delayed updates has been used in a paper submitted for peer review (De Vries et al., 2013). It was used to cluster the 50 million document ClueWeb collection into 140,000 clusters. These clusters were used for ad hoc retrieval, where only the first 8 most highly ranked clusters have to be searched to retain search quality. This allowed 13 times less documents to be retrieved than the previous best known result. Only 0.1% of all documents were contained in the most 8 highly ranked clusters when averaged across all 50 queries used for evaluation. Complete clustering of 50 million documents into hundreds of thousands of clusters has not been reported in the literature prior to this result, never mind in a single threaded, non-distributed algorithm. Other approaches have used sampling to address the scalability of clustering 50 million documents (Kulkarni and Callan, 2010). The clusters found using the sample were used to associate documents to the cluster centroids. However, the centroids were not updated and therefore out of vocabulary terms were encountered when ranking clusters. The approaches presented here performs complete clustering of the ClueWeb collection where centroids are updated and iteratively optimized for all documents in the collection.

The EM-tree algorithm can be initialized with any $m$-way tree. In all previous experiments the tree has been initialized using random seeding. In the following experiments we have initialized the EM-tree using a $m$-way tree produced by TSVQ using a sample. We randomly sampled 2 million documents from the 50 million document ClueWeb collection and built a $m$-way tree using TSVQ limited to 5 iterations of optimization. The choice of 5 iterations was influenced by the results in Table 1 where on average, 97% of the optimization occurred by the 5th iteration of optimization when using the smaller INEX 2010 XML Mining collection. The resulting $m$-way tree was used to initialize the EM-tree algorithm which was run for 2 iterations. The tree order was 331 and was 3 levels deep.

We built a K-tree with delayed updates for every 1000th vector on the 50 million document ClueWeb collection. All vectors were reinserted into the tree once the tree had been built. The tree order was 1000 and the resulting tree was 3 levels deep.

Table 2 contains the results of the experiments. The EM-tree initialized with TSVQ is able to produce lower RMSE solutions in less time than the delayed update K-tree. As previous results were stable across different random initializations we conclude that the same properties will hold here and therefore it is highly likely a difference of 56 bits in RMSE will
statistically significantly different. Running each algorithm 20 times is impractical in this case, as the experiments would take approximately 20 days to complete. Because the EM-tree algorithm can be initialized with any $m$-way tree, it has allowed a sample based tree built with TSVQ to provide lower distortion solutions than the previous approach based upon the K-tree with delayed updates.

12. Streaming EM-tree

We now consider the properties of the EM-tree algorithm in comparison to the TSVQ algorithm when the data set is too large to fit in main memory and it is streamed sequentially from disk for each pass over the data. The goal in this setting is to minimize the number of passes over the data. Furthermore, random access to the data set is not available and each example is presented one at a time. Note that the K-tree is not considered in this situation. While the K-tree algorithm performs incremental learning where examples are presented one by one, it is not streaming because it has to revisit previous examples from the stream when a leaf node becomes full and is split in two.

The main distinction between the EM-tree and TSVQ algorithms is that EM-tree optimizes the entire tree structure at each iteration while TSVQ optimizes the tree structure at each level before proceeding to the next. Therefore, we perform an analysis where the number of iterations of optimization is fixed. We then determine the number of passes over the data required by each algorithm.

Both algorithms require the data set to be streamed for the comparison of points to the centroids, and recalculation of the means. How the algorithms are presented in prior sections requires two passes of the data set to perform these operations. However, if an extra vector is kept for each centroid, it can be used as an accumulator when calculating the means of the vectors. Once all points in the data set have been streamed, the accumulators can be divided by the associated number of vectors. Therefore, one iteration of the optimization process of k-means in TSVQ or the insert and update procedures in EM-tree can be performed in a single pass over the data.

The seed procedure introduced in Section 5 performs a pass over the entire data set for each level in the tree. It is TSVQ limited to 1 iteration; i.e. $m$ points are selected, their nearest neighbors associated and the centroids updated. Therefore, we have defined a single pass approach to seeding the $m$-way tree structure in the following SINGLEPASSSEED psuedocode where splits occur in the tree after $s$ vectors are contained in a given leaf. The algorithm proceeds by creating a tree containing a linked list of centroids for the first $m$ points from the stream. In the streaming setting the points are not stored in the tree, so, only the cluster centroids are created. Points are then inserted into the tree one at a time, taking a point to be a new centroid once a leaf has seen $s$ insertions.

Alternatively, a tree can be built using TSVQ on a sample of the data that fits in memory. This can then be used to initialize the EM-tree algorithm.
De Vries, De Vine and Geva

SinglePassSeed($m, depth, Y, s$)
1  for each of the first $m$ points from the stream
2    create a branch that is $depth - 1$ levels deep containing 1 node at each level
3    use the point for the 1 centroid required in each node
4  for the remaining points in the stream
5    insert the point into the tree following the search path
6    if the lowest level centroid has seen $s$ insertions
7      make the point a new centroid
8    if the node is full
9      keep traversing up the tree until a non-full node is reached
10     if all nodes are full
11       discard the point
12     else
13       create a new branch to tree $depth - 1$
14       use the point as the centroid in each node in the new branch
15    update the centroids along the insertion path

Consider two $d$ level $m$-way trees, where all internal nodes are full, and where one is built using the EM-tree algorithm and the other TSVQ. Because all the internal nodes are full, both will have the same number of clusters. We define the number of iterations of optimization as, $i$. As described earlier, a single iteration requires the entire data set to be streamed from disk. In the case of TSVQ each non-leaf level requires the entire data set to be streamed from disk $i$ times by the k-means algorithm. Therefore, in a $d$ level tree produced by TSVQ, $(d - 1)i$ passes over the data will be performed. For an iteration of optimization in the EM-tree algorithm, each vector is read and then inserted into the tree while updating the accumulators along the search path, requiring $i$ passes over the data. For a, $d = 5$, level $m$-way tree performing, $i = 2$, iterations of optimization, TSVQ requires 8 passes over the data while EM-tree requires only 2. As the tree depth, $d$, increases, the disparity in the relationship increases. This says nothing about the convergence properties for the number of passes over the data. However, in Figure 3 it can be seen that 80% of the optimization happens after 2 iterations for both TSVQ and EM-tree.

Figure 6 compares EM-tree and TSVQ in the streaming setting. The INEX 2010 XML Mining collection described earlier was used. For the purpose of these experiments we do not stream the data from disk but simulate it by streaming data from main memory. 4 level trees were produced by both EM-tree and TSVQ to produce approximately the same number of clusters. The distortion of the lowest level, most fined grained clusters, in the tree was used. EM-tree was run with tree orders of 4, 5, 6, 7, 8, 10, and 12. TSVQ was run with tree orders of 4, 5, 6, 7, 8, 9, and 10. TSVQ was run using 2 iterations for k-means when recursively splitting the data set. The nearest neighbors are associated with centroids twice, and the centroids are updated twice. In both algorithms, 2 iterations performs approximately 80% of the optimization from the initial randomly initialized state. This can be seen in Figure 3 where 80% of the decrease in distortion occurs by iteration 2. Running TSVQ to create a 4 level tree, where k-means is run for 2 iterations, requires 6 passes over the data; i.e. 2 passes over the data for each non-leaf level in the tree. Therefore, EM-tree was run for 6 iterations to perform the same number of passes over the data for
The EM-tree Algorithm

![Graph](image_url)

Figure 6: INEX 2010 XML Mining Collection – Streaming EM-tree vs TSVQ

<table>
<thead>
<tr>
<th>TSVQ 6 passes over data</th>
<th>EM-tree 6 passes over data</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average Clusters</td>
<td>Average RMSE</td>
<td>Average Clusters</td>
</tr>
<tr>
<td>64</td>
<td>1760.57 ± 4.6</td>
<td>64</td>
</tr>
<tr>
<td>125</td>
<td>1728.07 ± 4.2</td>
<td>124.7</td>
</tr>
<tr>
<td>216</td>
<td>1702.82 ± 4</td>
<td>213.4</td>
</tr>
<tr>
<td>343</td>
<td>1677.70 ± 5.1</td>
<td>331.6</td>
</tr>
<tr>
<td>512</td>
<td>1654.34 ± 3.9</td>
<td>463.25</td>
</tr>
<tr>
<td>729</td>
<td>1632.82 ± 4.9</td>
<td>762.25</td>
</tr>
<tr>
<td>1000</td>
<td>1613.36 ± 5.3</td>
<td>1047.05</td>
</tr>
</tbody>
</table>

Table 3: INEX 2010 XML Mining Collection – Streaming EM-tree vs TSVQ

the simulation. Note that each tree was built 20 times given a different random initial state to measure the variance. The result is that EM-tree produces lower distortion solutions in 6 iterations than TSVQ does in 2 iterations, while both algorithms still perform 6 passes over the data.

Table 3 contains the numerical values for the plots in Figure 6. It also includes the p-value for a two tailed t-test between each 20 samples of the RMSE of the clusterings found. In all cases, EM-tree produces distortions statistically significantly lower than TSVQ with a p-value < 0.05 and almost equal to 0. This is even the case when a slightly larger number of clusters is produced by EM-tree, which slightly disadvantages EM-tree because a smaller number of clusters produces a higher distortion solution. This can be seen by the downward trend in distortion in Figure 6 as the number of clusters increases. Note that the exact number of clusters produced by EM-tree can not be controlled because of the pruning...
process. The same applies to TSVQ as k-means does not always produce k clusters. Some centroids can have no points associated with them.

In conclusion, the EM-tree performs less passes over the data for each iteration of optimization. This allows the algorithm to do something that TSVQ can not. It can perform 1 or 2 passes over the data. Where as TSVQ, for a 4 level tree, requires 3 or 6 passes over the data.

13. Parallel and Distributed EM-tree

The EM-tree algorithm is particularly amenable to simple parallel and distributed implementations due to the nature of the optimization process. Unlike K-tree or TSVQ, the entire tree is frozen at each iteration. The data points can be inserted according to the nearest neighbor search path while updating the accumulators along the way. Once all of the points have been reassigned to leaves, the accumulators can be synchronized between all threads or compute nodes co-operating in the algorithm. Each thread or compute node can work independently without any communication to other threads or compute nodes. The only time communication is needed is when the tree is synchronized after inserting all points.

The K-tree algorithm updates the model for every vector that is inserted into the tree. Additionally, this can cause nodes in the tree to split. All of these operations must be synchronized and require communication between threads or compute nodes.

In a distributed implementation, the data set is likely to be very large and streamed from disk. In this situation, as highlighted in Section 12, the EM-tree algorithm performs 1 pass over the data per iteration where as TSVQ performs as many passes as there are non-leaf levels in the tree.

14. Future Work

The experiments in this paper tested a single threaded implementation of the EM-tree algorithm. In future work, the parallel, distributed and streaming EM-tree can be implemented and tested on real data. This implementation will likely be able to cluster collections containing billions of documents in under 24 hours.

Further studies can possibly investigate the effect of having different order nodes at different levels of the $m$-way tree. The intuition is that the root node would benefit from containing more centroids. If there are too few points in the root node, a large fraction of the data set may be equidistant from all centroids due to the curse of dimensionality. In high dimensional spaces, such as documents, only the local neighborhood allows differentiation. These properties were analyzed by De Vries and Geva (2012) for TopSig document vectors.

15. Conclusion

The novel EM-tree clustering algorithm was presented. The $m$-way nearest neighbor search tree data structure and the algorithm have been formally defined. Furthermore, other geometric clustering algorithms such as k-means, TSVQ, agglomerative hierarchical and K-tree can be represented using the $m$-way tree. It was proven that the EM-tree algorithm con-
The EM-tree Algorithm

verges, which was also validated experimentally. The algorithm can produce lower distortion clusterings in the streaming setting than TSVQ, where the goal is to minimize the number of passes over the data set. Furthermore, the EM-tree can be initialized with a tree built using a sample and the TSVQ algorithm. This provides lower distortion clusterings in less time than the previous best known large scale approach to clustering 50 million documents in hundreds of thousands of clusters using the K-tree with delayed updates. Clustering of this scale has applications for search engines, where the clusters can be used to reduce the number of documents that are required to be searched by only searching the most relevant document clusters for a given query.

References


Conclusions, Implications and Future Directions

This thesis presented new approaches to knowledge representation, learning algorithms, evaluation and information retrieval with respect to document clustering. The primary focus was on the efficiency and scalability of the algorithms that allowed a 13 fold reduction in the number of documents that have to be searched in a cluster-based search engine compared to the best known approach on large scale document collections.

The TopSig representation was introduced and used for improving the efficiency of document clustering by an order of magnitude with no reduction to cluster quality [90]. Additionally, further gains in efficiency can be made by sacrificing quality by using shorter document signatures. This required development of learning algorithms that work directly with the binary representation of TopSig document signatures.

Two algorithms based upon the $m$-way nearest neighbor search tree were presented for further increases in the efficiency of clustering. The TopSig K-tree [61] allowed document clustering at a scale not previously reported in the literature, by clustering 50 million documents into 140,000 clusters. It did this in a single thread of execution. The EM-tree algorithm [55] allowed further increases in single threaded efficiency. Additionally, the EM-tree algorithm has properties that make it suitable for further scaling via streaming, distributed and parallel implementations of the algorithm. Both of these algorithms allow a further trade-off between cluster quality and efficiency. However, without the scalability of these approaches to produce hundreds of thousands of clusters, the improvements seen in CBM625 would not have been realized.

The results presented suggest that clustering of the entire searchable Internet
of approximately 45 billion documents, into a hierarchy of millions of clusters, is a tractable problem. The implications and applications of clustering at such a scale are unknown. However, it is easy to see how these scalable hierarchical algorithms can be used to build a machine learned hierarchical browsable catalog of the entire searchable Internet. Additionally, the clusters found by these scalable approaches may be able to reduce the cardinality of a data set that can be fed into more expensive learning algorithms to fine tune the clustering.

Category based evaluations require labeling of an entire document collection with multiple categories. When hundreds of thousands of categories exist for a general purpose knowledge repository such as the Wikipedia or the Internet, this becomes a daunting problem. There have been reports of labeling document collections causing “terrible memories” \[152\] for assessors even though the document collection consisted of less than a million documents, had hundreds of categories, had a tool to help automate classification based on rules, and was labeling short news wire articles. The problem of large scale assessment has already been addressed via the use of pooling in ad hoc information retrieval evaluation. Additionally, queries used for evaluation are often specific and well defined, unlike some of the broad, lofty categories used for document categorization.

This thesis highlights that the cluster hypothesis holds with very fine grained document clusters when they are used for ad hoc retrieval \[61\]. This was further reinforced by the analysis of the topology of document signatures \[58\], that indicated that only the local neighborhood of the representation allows differentiation of meaning. Therefore, creating document clusters too large is likely to contain many equidistant documents. It was shown that fine grained clusters are more effective for cluster-based search.

The novel evaluation of document clustering at INEX \[171, 62\] using ad hoc relevance judgements produced several findings. The use of ad hoc relevance judgements overcomes the short-falls of using categories. Category based evaluation of document clustering only has use for classification. However, ad hoc retrieval based evaluation has a use case for cluster-based retrieval. We suggest that evaluating document clustering in the situation of its use is more sound than a classification based evaluation. If you want to do classification, build a classifier. Document clustering is driven by the idea of placing similar items together. Just because two documents exist in a category does not necessarily mean they are similar. The cluster hypothesis is directly driven by the similarity between documents, where documents who are similar to each other, are likely to be relevant to the same information need.

This thesis investigates the Divergence from a Random Baseline approach to the evaluation of clustering \[55\]. It allows for the differentiation of ineffective
clusterings that perform no useful learning with respect to a given measure of cluster quality. This allowed for identification of ineffective clusterings in the INEX 2010 XML Mining track.

17.1 Future Directions

Due to the broad and inter-related nature of the topics in this thesis, there are a multitude of possible directions that future research may take. The following material highlights some possibilities that I think are interesting. It is by no means an exhaustive list.

Clustering of billions of documents into fine grained clusters is an exciting avenue for the future of this research. The results already indicate that streaming, parallel and distributed implementations of the EM-tree algorithm are straightforward. Once clusters of this scale are available then there are many new possibilities for use in different applications.

One possible application for fine grained clusters is the sparsification of pairwise distance matrices to only include the local neighborhood of the objects being clustered.

Large scale clustering can be applied to other domains such as feature learning for image classification. Recent results [43] have used k-means to cluster 57 million images into 150,000 clusters. The approach outlined in the paper uses 30 machines and takes 3 days to train. This is a new domain algorithms in this thesis can be applied to.

Analysis of documents by breaking documents into separate sub-document pieces could prove useful for document clustering and cluster-based retrieval. The cluster hypothesis could be stronger at the sub-document level. It also allows documents to be allocated to one or more clusters without using fuzzy clustering algorithms; i.e. separate parts of a document can belong to different clusters. The primary motivation for this approach is to deal with multi-topic documents.

The tree based algorithms presented in this thesis could be adapted to produce soft clusters where an example can be associated with more than one cluster at any given level in the tree.

Automatically learning all the parameters for building a cluster tree would be useful. This way, the learning approaches can be thrown at a large document collection and a tree that best fits the data according to some measure of fitness can be produced. This requires determining the appropriate length of signature to use for each cluster in the tree and determining the best tree structure in terms of splits and tree depth.

Different document representations can be used with document signatures.
It would be interesting to investigate the effect of the different representations for document clustering with signatures. For example, simply using term frequencies with signatures may prove effective for document clustering.

The cluster-based retrieval experiments indicated that squaring BM25 document weights proved more effective with respect to the cluster hypothesis. The query is also compared to documents using the inner product rather than the cosine angle between the query and documents. Therefore, squaring BM25 weights and using the inner product distance measure may well prove to be effective for improving the quality of document clusters. Furthermore, inner product based compressed representations such as Reflexive Random Indexing may prove more effective for representing documents based on BM25 for document signatures.
Bibliography


197


