Using Wikipedia Knowledge and Query Types in a New Indexing Approach for Web Search Engines

by

Falah Hassan Ali Al-Akashi

Thesis submitted to the
School of Electrical Engineering
and Computer Science of the University of Ottawa
in partial fulfillment of the requirements for the Degree of
Doctor of Philosophy

University of Ottawa

Faculty of Graduate and Postdoctoral Studies

This is to certify that I have examined this copy of a doctoral dissertation

by

Falah Hassan Ali Al-Akashi

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examination committee have been made.

Defense Committee:

_____

Stan Matwin

_____

Charle Clarke

_____

Thomas Tran

_____

Babak Esfandiari

Supervisor

_____

Diana Inkpen

License and Agreement

In presenting this dissertation in partial fulfillment of the requirements for the doctoral degree at the University of Ottawa, I agree that the Library shall make its copies freely available for inspection. I further agree that extensive copying of this dissertation is allowable only for scholarly purposes, consistent with “fair use" as prescribed in the University of Ottawa Copyright-Law.

Signature _____

Date _____

ABSTRACT

The Web is comprised of a vast quantity of text. Modern search engines struggle to index it independent of the structure of queries and type of Web data, and commonly use indexing based on Web's graph structure to identify high-quality relevant pages. However, despite the apparent widespread use of these algorithms, Web indexing based on human feedback and document content is controversial. There are many fundamental questions that need to be addressed, including: How many types of domains/websites are there in the Web? What type of data is in each type of domain? For each type, which segments/HTML fields in the documents are most useful? What are the relationships between the segments? How can web content be indexed efficiently in all forms of document configurations? Our investigation of these questions has led to a novel way to use Wikipedia to find the relationships between the query structures and document configurations throughout the document indexing process and to use them to build an efficient index that allows fast indexing and searching, and optimizes the retrieval of highly relevant results. We consider the top page on the ranked list to be highly important in determining the types of queries. Our aim is to design a powerful search engine with a strong focus on how to make the first page highly relevant to the user, and on how to retrieve other pages based on that first page. Through processing the user query using the Wikipedia index and determining the type of the query, our approach could trace the path of a query in our index, and retrieve specific results for each type.

We use two kinds of data to increase the relevancy and efficiency of the ranked results: offline and real-time. Traditional search engines find it difficult to use these two kinds of data together, because building a real-time index from social data and integrating it with the index for the offline data is difficult in a traditional distributed index.

As a source of offline data, we use data from the Text Retrieval Conference (TREC) evaluation campaign. The web track at TREC offers researchers chance to investigate different retrieval approaches for web indexing and searching. The crawled offline dataset makes it possible to design powerful search engines that extends current methods and to evaluate and compare them.

We propose a new indexing method, based on the structures of the queries and the content of documents. Our search engine uses a core index for offline data and a hash index for real-time

data, which leads to improved performance. The TREC Web track evaluation of our experiments showed that our approach can be successfully employed for different types of queries. We evaluated our search engine on different sets of queries from TREC 2010, 2011 and 2012 Web tracks. Our approach achieved very good results in the TREC 2010 training queries. In the TREC 2011 testing queries, our approach was one of the six best compared to all other approaches (including those that used a very large corpus of 500 million documents), and it was second best when compared to approaches that used only part of the corpus (50 million documents), as ours did. In the TREC 2012 testing queries, our approach was second best if compared to all the approaches, and first if compared only to systems that used the subset of 50 million documents.

Acknowledgements

Firstly, I would like to express my deepest appreciation to my dissertation supervisor, Dr. **Diana Inkpen**, for her valuable guidance, support, encouragement, kindness, feedback, and enthusiasm through the journey of my Ph.D., and my entire graduate study at University of Ottawa. Diana, I am grateful for your advice and the discussions we had; you have always pointed me to very interesting directions of thinking and research.

Secondly, I would like to thank the rest of my thesis committee members: Dr. **Charles Clarke**, Dr. **Stan Matwin**, Dr. **Thomas Tran**, Dr. **Babak Esfandiari**, for their insightful comments, questions, and directions. Also, I thank the chairperson, **Luise Von Flotow**, for actively participating in my committee.

Deep thanks to the Faculty of Graduate and Postdoctoral Studies, especially the coordinator Miss **Victoria Stewart**, for preparing the room and her encouragements.

Thirdly, I would also like to thank the following people:

- TREC team for providing me a full dataset and for evaluating our system for three times, and for providing me all the tools required for testing the system.
- All the members of the Natural Language Group, past and present, for making it a great place to discuss research issues. And all those with whom I had a wonderful time in the past and at present.
- My parents for their unconditional support. Their care and love made it possible for me to complete this work.
- My wife, Hiba, and my kids, Mohammed and Sama, for giving me the enough time to be far from them from time to time.
- My father in law and mother in law for encouraging me to do the PhD.
- My brother, Karar, and all of my family for supporting me to do the Ph.D.
- The government of Iraq for offering me the scholarship to complete the PhD degree.
- Finally, I also want to thank all the people who graciously participated in my experiments and provided me with valuable suggestions and comments.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

Abbreviation	Definition
AP	Average Precision
BM25	non-Binary Model
BOW	Bag of Words
CC	Cluster Contamination
CD	Classification Dictionary
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
DCF	Description-Comes-First
DCG	Discounted Cumulative Gain
DF	Document Frequency
ERR	Expected Reciprocal Rank
ERR-IA	Intent-Aware Expected Reciprocal Rank
ExDB	Extraction DataBase
FIFO	First In First Out
FOAF	Friend Of A Friend
FS	Feature Selection
GCMD	Global Change Master Directory
GUI	Graphical User Interface
GVSM	Generalized Vector Space Model
HITS	Hypertext-Induced Topic Search
HTML	Hyper Text Markup Language
HCI	Human-Computer Interaction
IDF	Inverse Document Frequency
IG	Information Gain
IR	Information Retrieval
LSA	Latent Semantic Analysis
LSI	Latent Semantic Indexing
MAC	Matching Anchor Count
MAP	Mean Average Precision
ME	Maximum Entropy
ML	Machine Learning
MSE	Multiple Sources of Evidence
NB	Naive Bayes
nDCG	intent normalized Discount Cumulative Grade
NIST	National Institute of Standards and Technology
NPOV	Neutral Point Of View
ODP	Open Directory Project
OPTICS	Ordering Points To Identify the Clustering Structure
PAM	Partitioning Around Medoids

RBP	Rank Biased Precision
SALSA	Stochastic Approach for LinkS Analysis
SC	Snippet Coverage
SDT	Statistical Decision Theory
SensorML	Sensor Modeling Language
SEO	Search Engine Optimization
SIFT	Segmented Information Fusion Techniques
SM	Similarity Merge
SOM	Self-Organizing Map
SQL	Structured Query Language
SRR	Search Result Records
STC	Suffix Tree Clustering
SVD	Singular Value Decomposition
SVM	Support Vector Machine
SynSet	Synonyms Sets
TC	Text Categorization
TC	Topic Coverage
TF	Term Frequency
TF/IDF	Term Frequency/Inverse Document Frequency
TM	Term Match
TOPHITS	Topical Hypertext-Induced Topic Selection
TREC	Text REtrieval Conference
UDP	Unique Document Percentage
URL	Uniform Resource Locator
VSM	Vector Space Model
WARC	Web ARChive
WEBSOM	WEB Self-Organizing Map
WePS	Web Person Search
WIR	Web Information Retrieval
WLC	Weighted Linear Combination
WN	WordNet
WRS	Weighted Rank Sum
WSD	Word Sense Discrimination
WSE	Web Search Engine
WSI	Word Sense Induction
WSR	Web Search Results
WWW	World Wide Web
XML	eXtensible Markup Language
xQuAD	eXplicit Query Aspect Diversification

Chapter 1

Introduction

1.1 Introduction

The Web contains a huge amount of meta-data and information about the structure of the data (Cafarella, M., 2009) that is currently beyond the reach of search engines. Despite the use of structure information implicitly found in Web content, modern search engines struggle to index this content independent of the structure of queries and topical resources. Content-based Web page retrieval is different from anchor-based retrieval, as it analyzes the actual content of web pages rather than the anchor texts that are extracted from anchor links associated with each page. The term 'content' in this context can refer to URLs, titles, headings, meta-data (such as keywords), other HTML tags, descriptions and/or any other information that can be derived from the web content itself.

However, content-based indexing is not the only challenge in web searching; the process of detecting the type of queries submitted by users can also be difficult (Lioma et al., 2009). In order to address these challenges, our work creates a centralized index with the aim of retrieving relevant pages from different types of index classes by detecting the type of queries first. The main concept of our framework is based on designing an approach for indexing a Web collection, and selecting an appropriate class of data from the index repository on a per-query basis. The selection of the relevant class is based on the outcome of tests performed on a sequence of index classes.

This thesis investigates several approaches to build a centralized index in three broad types of progressively developed experiments. The first experiment exploits the tags of meta-data in web documents. The second experiment considers the relationship between meta-data and the document content itself, and how to exploit the meta-data for indexing the content of related documents efficiently. This experiment also considers the effectiveness of Wikipedia content for home page finding tasks and predicting other relevant pages, how to exploit Wikipedia knowledge to enhance the search results, and the importance of term impact rather than term frequency. The third type of experiment addresses the pitfalls and drawbacks in the previous two experiments that caused failure to capture the relevant pages for some types of queries. It expands and develops the previous experiments by proposing the efficiency of topical phrases

more broadly in realistic settings (unlike domain-specific phrases), such that the topical phrases are extracted from a dynamic and knowledgeable resource such as Wikipedia, and from search engine log files. In addition, we propose two enhancements that aim to make our search engine more diverse by indexing recent social media data and by better visualization of the retrieved results. The real-time data available in the social network is exploited by our search engine because, in our perspective, users are not always searching in archived data; they often look through real-time data such as news, events or recent activities. Visualization techniques can reduce the time required for satisfying the information need by allowing quicker inspection of various aspects of the retrieved results.

In our experiments, we investigate the importance of Wikipedia as a reference for Web page indexing and query expansion for both types of data. Our thesis also demonstrates how organizing the index in the centralized structure is better than in the distributed structure, which deals with presented controlling issues by removing the duplicated documents and displaying the search results in a high-level user interface. The experimental results show that our approach can significantly improve retrieval efficiency, compared to several models that use other algorithms based on content or web graph.

1.2 Motivation

Information retrieval (IR) is the process of obtaining data relevant to an information need from a collection of resources. IR has been an active field of research for over 60 years. The idea of using computers to search for relevant information was first popularized in the article “As We May Think”_by Vannevar Bush in 1945. Over the last decade, many techniques have been proposed related to searching for a particular information need. Searching can be based on meta-data or full-text indexing. Automated information retrieval systems are used to reduce what has been called ‘information overload‘. Many universities and public libraries use IR systems to provide access to books, journals and other documents. Web search engines are the most prevalent IR applications.

An IR process begins when a user enters a query into the system. Queries are formal statements of information needs, such as search strings in web search engines. In IR, a query does not uniquely identify a single item in the collection; several items could match the query, and have different degrees of relevancy.

Often, documents themselves are not kept or stored directly in IR systems. Instead, they are represented in the system by document surrogates (index terms). Most Web information retrieval (WIR) systems compute a numeric score of how well items in the index file match the query, and rank the items accordingly. The top ranking documents are displayed to the user, and the process can be repeated if the user wishes to refine the query.

Classical IR systems have been primarily used in controlled settings, where documents are static or rarely updated, information is considered to be reliable and the users are search experts. However, in WIR the environment is different, because the Web is dynamic and highly diverse, and information is often added, updated or, in some cases, unavailable. Indeed, even the available information is sometimes erroneous or intentionally misleading. Users that access the Web have wide ranges of backgrounds and interests, which often make it impossible to match the topics they are searching for with the few words they enter in the query field. In addition, the Web is continually growing and becoming more complex. Likewise, queries are becoming more complicated, though they can generally be classified into four broad categories that cover most web search topics (Manning et al., 2008). The first category is informational queries that cover a broad topic (e.g., 'colorado' or 'trucks') for which there could be thousands of relevant results. The second is navigational queries that seek a single website, or the webpage of a single entity (e.g., 'youtube' or 'delta airlines'). The third category is transactional queries that reflect the intent of the user to perform a particular action, such as purchasing a car or downloading a program. Search engines often support a seldom used fourth category, connectivity queries, which report on the connectivity of the indexed web graph (e.g., which links point to this URL? or How many pages are indexed from this domain name?).

Not all websites are equal; some are popular and trustworthy, while others do not have credibility. Only a few websites provide connectivity and engagement between popular websites in a social manner. Wikipedia, for example, has a type of dynamic content; the writers chose representative web pages as backgrounds for writing and citing articles on specific topics. As a result, Wikipedia seeks to create a summary of all human knowledge in the form of an online encyclopaedia, with each topic covered in one or many related articles. Since Wikipedia has many terabytes of disk space, it can address far more topics than can be included in a conventional printed encyclopaedia. It also contains materials that some people might find objectionable.

However, content in Wikipedia is subject to the copyright laws of the United States. A topic must also meet Wikipedia's standards of ‘notability‘, which usually means that it must have significant coverage in reliable secondary sources, such as mainstream media or major academic journals that are independent of the subject of the topic. Furthermore, Wikipedia is intended to convey only knowledge that is already established and recognized. It does not support opinions and viewpoints if they are attributable to external sources, and must offer an appropriate share of coverage within an article; this is known as the neutral point of view (NPOV)[1]. Consequently, Wikipedia gained early contributors from ‘Nupedia’, Slashdot postings, and web search engines.

Wikipedia is the largest and most popular general reference works on the Internet[2]. In 2006, Time magazine recognized Wikipedia’s contribution to the rapid growth of online collaboration and interaction by millions of people around the world. According to *comScore* Networks Inc[3], in January 2007, Wikipedia made the top ten list of the most popular websites in the United States for the first time. And according to ALEXA NET[4], as of May 2012, Wikipedia was the sixth most popular website worldwide, receiving more than 2.7 billion U.S. page views per month out of a global monthly total of over 12 billion, and with an estimated 365 million readers worldwide. Finally, some web search engines make special use of Wikipedia content when displaying search results, including ‘Google’, ‘Bing’, and ‘Duck Duck Go’.

Recent investigations into query log files of popular search engines, such as Google, revealed that one in three queries entered by users referred to topics covered by the Wikipedia. Other resources are providing highly structured representations of Wikipedia, such as DBPedia[5], Yago[6] and Wordnet[7]. DBPedia not only describes typed relations between Wikipedia entities, it also contains contextual links to other repositories, and provides a basic ontology ideally suited for entity ranking tasks[8]. Wikipedia is a more dominant repository than DBPedia, as only a limited number of Wikipedia articles are categorized by the DBPedia ontology. WordNet, which is a lexical database, computes the semantic distance between concepts or words, and then uses this

[1] http://en.wikipedia.org/wiki/Wiki
[2] http://en.wikipedia.org/wiki/Wikipedia
[3]http://www.comscore.com/
[4]http://www.alexa.com/
[5] http://dbpedia.org/
[6] http://yago-knowledge.org/
[7] http://wordnet.princeton.edu/
[8] http://dbpedia.org/ontology/

term distance to compute the similarity between a query and a document. However, Mandala et al., (2000) argued that WordNet failed to improve the retrieval effectiveness of information retrieval applications. They found that the main reason is that relationships between terms are not in WordNet, and some terms, such as proper names, are also not included. The above capabilities motivated us to use the Wikipedia resource (available in the main content of our dataset ClueWeb09) to address the challenges in Web document content indexing and query processing.

1.3 Thesis Goal

Our research goal is to design a Web search engine in a central server that can index and search in both offline and real-time data, to improve the retrieval relevancy. This is considered difficult, due to the data properties mentioned above and other related task properties, including a lack of clear topic boundaries in most Web documents and user queries, the fact that many of the relevant topics are available as sub-topics or even semantically similar to other topics in other documents, users' points of view, spam documents and the effort required for homepage and entity finding tasks. Another recent challenge in Web search is how to show the relevant pages in an effective manner, and how to distribute them in the list to satisfy different types of queries, different tasks, and different user needs in terms of user-provided queries. We think that documents should not be ranked only by term occurrences, but according to the types of queries, quality of the documents, and user site preferences. In addition, real-time data and recent activity are also important for indexing, and recent search engines are unable to index real-time activities and recent blogs from social sites (e.g., Facebook and Twitter), unless they use a new approach for indexing these types of data. We designed our search engine to exploit this data, to satisfy diverse user needs.

There are many techniques and algorithms in the literature related to Web search engines that improve information retrieval from Web collections. However, the key question that we address in this thesis is: Which WIR model is most efficient for Web collections, and how could that model effectively overcome the difficulties and drawbacks of current systems in order to improve the retrieval results?

1.4 Thesis Contributions

The main contributions of this thesis are:

1- **New Indexing Platform:** The major contribution of this thesis is to design a novel and powerful indexing method for different types of data (real-time and offline) in a centralized system in an efficient manner, and retrieve pages through search queries that satisfy diverse user' requirements. Most search engines use a distributed index, but our method allows the use of a centralized index. We exploit the implicit query structure and the web page content (meta-data) to improve retrieval relevancy. We also show how Wikipedia knowledge can be used to find relevant documents. Since the articles are semantically connected in different ways, Wikipedia is a good reference for knowledge discovery. We propose a novel method of using Wikipedia knowledge for Web indexing and searching for home page finding tasks, to determine the correlation between the relevant pages, and for query normalization and expansion. Our motivation is that the knowledge available in Wikipedia can leverage information that cannot be deduced solely from the texts being indexed.

2- **Term Impact:** We propose a novel method for computing the term importance for document weighting. We call it "term impact" and we compute it as the cosine similarity between the vector of the term built on meta-data and the vector of the document's text. The method is more powerful than term frequency, documents length normalization, and inverse document frequency for computing the term importance. Similarly, we present a novel method for computing the importance of each document based on its location in a hosted site, as well as the structure of the website.

3- **Real time and Social Search:** This aspect of our work enhances our novel index to include recent Facebook and Twitter data. We use Tweeter text messages rather than anchor texts to increase the relevance of the documents that are used by many users. Our search engine is therefore able to retrieve recent information from social media for any search query**.**

4- **User Interface:** Results displayed by current search engines are organized as a simple list. We propose a new kind of interface that displays the results more efficiently as they are presented as titles, descriptions, urls, and screenshots, rather than presenting only titles, descriptions, and URLs, as in the traditional search engine. Users are able to explore the results in the same interface, to keep them in the same context of searching. This kind of interface helps users to access the relevant pages and the right entity information quickly.

Though validation with user studies need to be done in the future to prove the value of this visualization method.

5- **Research Publication:** as part of our contribution, we published our approaches in TREC proceedings and WIMS conferences, as shown below:

- Al-akashi, F. and Inkpen, D. (2010), 'University of Ottawa at TREC 2010 Web Track: Ranking Web Documents Using Meta-Data'. In Proceedings of 19th Text Retrieval Conference (TREC).

- Al-akashi, F. and Inkpen, D. (2011), 'Ranking Web Pages Using Collective Knowledge'. In Proceedings of the 20th Text Retrieval Conference (TREC).

- Al-akashi, F. and Inkpen, D. (2012), 'Intelligent Web page retrieval using Wikipedia knowledge'. In Proceedings of the 2nd International Conference on Web Intelligence, Mining, and Semantics (WIMS), ACM, ISBN 978-1-4503-0915-8, pages 51-57.

- Al-akashi, F. and Inkpen, D. (2012), 'Query-Structure Based Web Page Indexing'. In Proceedings of 21st Text Retrieval Conference (TREC).

- Al-akashi, F. and Inkpen, D. (2014), 'Term Impact-Based Web Page Ranking'. In Proceedings of the 4th International Conference on Web Intelligence, Mining, and Semantics (WIMS), ACM, ISBN 978-1-4503-2538-7.

1.5 Thesis Outline

The rest of this thesis is organized as follows:

1- **Chapter 2** summarizes the background literature on Web information retrieval. It describes a series of WIR models, provides a brief description of our Web collection, and presents the standard methods for evaluating WIR systems.

2- **Chapter 3** presents our first approach to WIR in detail, and examines the impact of meta-content for Web page indexing.

3- **Chapter 4** enhances our approach by adapting knowledge from Wikipedia, and combining the meta-data with Wikipedia knowledge. We also describe how Wikipedia knowledge can enhance the retrieval results.

4- **Chapter 5** demonstrates our approach to building a high quality, query-based index, in which the query structure is matched with document content in order to close the gaps for types of queries that could not be improved by the previous approach.

5- **Chapter 6** describes our final enhancements, which expand our current approach to social networks data (e.g., Facebook and Twitter). We demonstrate how to crawl, index and display the data in real-time. We also showed how to expand our approach to index the anchors texts of the whole collection. Finally, we propose a novel interface for result presentation.
6- **Chapter 7** describes our conclusions and suggests possible future directions to extend the proposed approach.

Chapter 2

Background

2.1 Introduction

Web Information Retrieval (WIR) is about efficient storage of information items, and easy access to them. Information items include hyper-text, images, video and more. A common scenario of using a WIR system could be: While performing a task, a user needs to locate information in a repository of crawled documents. The user expresses the information need in the form of a query, which usually corresponds to different lengths of keywords. The user is only interested in the documents that are relevant to his information need.

The main goal of a WIR system is to return all relevant documents, and to not retrieve those that are not relevant. Furthermore, the retrieved documents should be ranked from most to least relevant. This process is iterative, in the sense that a user can refine the initial query, or the system can provide feedback from the initial result, both of which lead to repetition of the retrieval process. Automatically deciding whether a document is relevant to the information need of a user is not simple, due to the uncertainty in formulating a query for information, as well as the inherent ambiguity of information in Web documents. This is a key difference between Information Retrieval and Data Retrieval, since to retrieve data items they must clearly satisfy a set of conditions which can be easily verified (Plachouras, 2006). This chapter provides an overview of the basic concepts of WIR. Section 2.2 discusses the architecture of WIR systems, Section 2.3 reviews the mathematical models behind Web indexing, Section 2.4 presents the mathematics behind Web ranking, Sections 2.5 and 2.6 review the topic distillation and enhancing tasks, Section 2.7 presents the entity finding task, Section 2.8 examines the query expansion and refinement task, Section 2.9 presents Text Retrieval Conference (TREC) and its related tasks, Section 2.10 discusses the parameter settings used in some models, and Section 2.11 presents a short definition of stop words and stemming.

2.2 The Components of a WIR System

In this section, we describe the key components of current Web information retrieval systems. As the World Wide Web has grown, the systems and methods for designing web searching have

changed significantly. We discuss document gathering, document indexing, query searching, and result display. Document gathering and document indexing need to be run only when the underlying set of web documents has changed (Abhishek and Ankit, 2012). Figure 2.1 below shows a basic WIR system. We will explain each component in the next subsections.

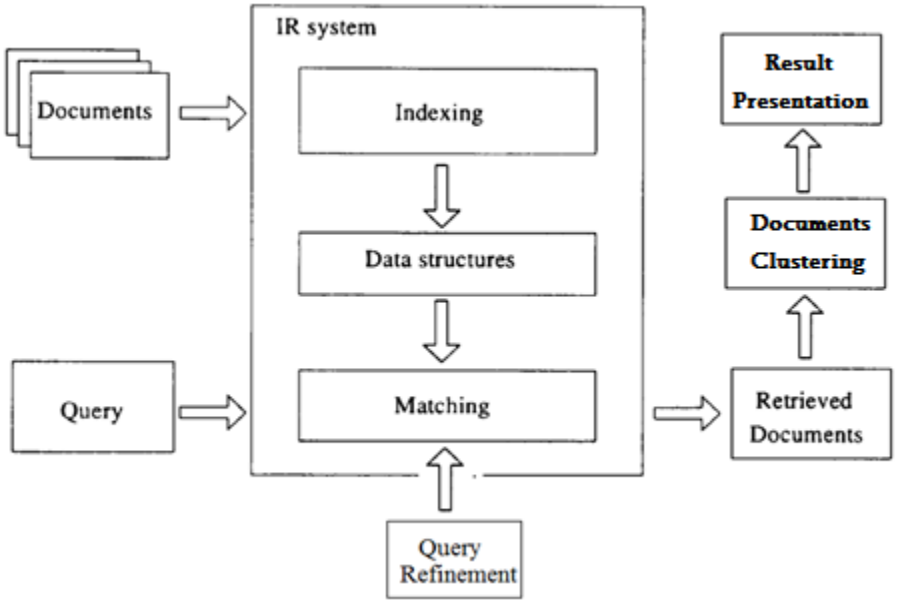


Figure 2.1 The architecture of a basic Web information retrieval system

2.2.1 Document Gathering

Web documents are normally gathered using a crawler. Crawlers traverse a web graph by recursively following hyperlinks, storing each document they encounter, and parsing stored documents for URLs to crawl. Crawlers typically maintain a frontier, which is a queue of the pages remaining to be downloaded. The frontier may be a FIFO queue, or sorted by some other attribute such as perceived authority or frequency of change. Crawlers also typically maintain a list of all downloaded or detected duplicate pages (so pages are not fetched more than once), and a scope of pages to crawl (i.e., maximum depth, specified domain or timeout value), both of which are checked prior to adding pages to the frontier. The crawler frontier is initialized manually with a set of pages from which the crawl starts. Crawling ceases when the frontier is empty, or occasionally when a resource limit is reached. Once crawling is complete, the downloaded documents will be indexed as per the following stage (Naveen et al., 2012).

2.2.2 Document Indexing

The indexer distills information in the corpus of documents into a format that is suitable for quick access by the query processor. This typically involves extracting document features by breaking documents down to their constituent terms, extracting statistics related to the terms present in the documents in the corpus, and calculating any query related evidence. After the index is built, the system is ready to process queries (Naveen et al., 2012).

2.2.3 Query Processing and Document Ranking

The principal component of a query processor is the document ranking function. The ranking functions of modern search systems frequently incorporate many forms of document evidence. Some of the evidence, such as textual information, is collected locally for each document in the corpus (see Section 2.5.4). Other evidence, including external document descriptions or recommendations, is collected by examining the context of a document within the web graph (described in Section 2.5.1).

For example, given a query composed of three terms, the first step is to find those terms in the index file. Then the corresponding posting lists are fetched (and transferred to memory if they reside on the disk), intersecting between document identifiers that are posted as a list in the posting list memory. A key procedure is to start with the most frequent terms, since their posting list will be the largest. The similarity formulation between documents and queries is based on models of documents and queries, the most effective of which is the vector space model (Wong et al., 1975). The cosine measure has consistently been found to be the most successful similarity measure in this model. It represents document properties as vectors, and uses the distance function (the cosine of the angle) between each vector pair. Finally, the retrieved set of documents are ranked and re-ordered for presenting to the user.

2.2.4 Result Presentation Interface and Grouping

Current popular web search systems present a linear list of ranked results, sometimes with the degree of relevance and/or summaries (snippets) of the matching documents. Later on in this thesis, we will experiment with new ideas about presentation issues. For example, the results will be displayed and grouped based on the types of queries and types of results, in such a way that users could retrieve the relevant documents faster.

However, current Web search engines (WSE) retrieve too many documents for a query; that is, only a small percentage is relevant to the user query, and the most relevant documents are shared

topically. To improve WSE performance, grouping and clustering techniques are being used to achieve meaningful search results. Clustering techniques are used to group similar documents' topics, to facilitate presentation of the results in more compact form, and to provide optimal search results. There are three main criteria for creating cluster categories: titles that are concise, accurate and recognizable. The key requirements for Web document clustering are defined in Zamir (1999). With clustering, search engines gather the results into groups related to a certain theme, or in some cases, they just provide users with related keywords that they may not have thought of themselves (Naveen et al., 2012).

2.3 Information Retrieval Models

An indexing model is required to build a retrieval model. While the indexing models discussed below share similar document statistics, they were derived through different relevance matching assumptions. The experiments in this thesis use a vector space model (outlined in Section 2.3.1.1), and other full-text ranking methods are discussed for comprehensiveness. The notation used in the model discussions below is as follows: 'D' denotes a document, 'Q' denotes a query, 't' represents a term, 'W_t' denotes a weight or a score for a single term, and *'S(D,Q)'* is a score assigned to the query '*Q'* for documents matching '*D'*.

2.3.1 Classic Information Retrieval Models

In this section, we discuss the three classic matching models: Vector Space (the most widely used), and Boolean and Probabilistic (which are used the least).

2.3.1.1 The Vector Space Model

The VSM has been a standard model for representing documents in information retrieval for almost three decades (He et al., 2006).

Generally, a standard retrieval algorithm takes a query Q and a set of documents $D1, D2, D3\ldots, Dn,$ and identifies the similarity score by *Similarity Coefficient* $SIM(Q, Di)$ between the query and each document. In VSM, both the query and the documents are represented as vectors in the term space. Documents with content that corresponds most closely to the content of the query, as measured by the terms in the document, are deemed to be the most relevant. To construct a vector that corresponds to each document, the model needs the following definitions: (i) the term frequency in each document tf_{ij}, (ii) the document frequency df_j in the

collection that contains the term t, and (iii) the inverse document frequency, which is the document frequency in the whole collection (of size N), $\text{idf}_j = \log(\frac{N}{\text{df}_j})$. So, the term weighting scheme is given as follows:

$$W_{ij} = \text{tf}_{ij} * \text{idf}_j$$

A simple similarity coefficient between any two vectors representing documents, queries, snippets or combinations of these is computed by using the cosine similarity, *SIM(Q, D)*.

$$\text{SIM}(Q, D_i) = \frac{\sum_{i=1} W_{Q,j} W_{i,j}}{\sqrt{\sum_{j=1} W_{Q,j}^2} \quad * \quad \sqrt{\sum_{i=1} W_{i,j}^2}}$$

Salton and Buckley [9] improved query term weighting by using a variation of the basic tf/idf, using the following weighting scheme:

$$W_{ij} = \frac{(\log \text{tf}_{ij} + 1.0) * \text{idf}_j}{\sum_{j=1}^{t} [(\log \text{tf}_{ij} + 1.0) * \text{idf}_j]^2}$$

Finally, there are other related models that are not widely used in Web information retrieval, including the Boolean and the Probabilistic model.

2.3.1.2 Boolean Model

In the Boolean model, retrieved documents are matched to queries created with logic operators. There are no degrees of matching; a document either satisfies the query or it doesn't. Thus, Boolean models are often referred to as 'exact matching' techniques. While Boolean matching makes it clear why documents were retrieved, its syntax is largely unfamiliar to ordinary users. Nevertheless, empirical evidence suggests that trained search users prefer Boolean search, as it provides exact specifications for retrieving documents (Tang, 2006). However, without any ranking by degree of match, the navigation of the set of matching documents is difficult, particularly on large corpora with unstructured content. Empirical evidence also suggests that using term weights in the retrieval model results in large gains. Upstill (2005) stated that employing Boolean matching techniques on corpora would have to be supplemented by some other document statistics, to provide a ranked list of results.

Generally, the Boolean model is considered to be the weakest classical method. Its main problem is the inability to recognize a partial match to a query, which frequently leads to poor performance (Creswell, 2002).

2.3.1.3 Probabilistic Model

The probabilistic model attempts to capture relevant documents within a probabilistic framework. The fundamental idea is as follows: If, given a user query q, there is a set of documents d_j, (which is the ideal answer set in the collection which contains exactly the relevant documents and no others), the probabilistic model tries to estimate the probability that the user will find the document d_j relevant. The model assumes that the relevance probability depends on the query and document representations only, and that there is a subset of all documents which the user prefers as the answer set for the query q. Such an ideal answer set is labeled R, and should maximize the overall probability of relevance to the user. Documents in set R are predicted to be relevant to the query, and documents not in set R to be non-relevant. This assumption is problematic, because it does not explicitly indicate how to compute the probabilities of relevance.

In this model, the index term weight variables are all binary (i.e.$W_{i,j} \in \{0,1\}$, $W_{i,q} \in \{0,1\}$), and a query q is a subset of index terms. Let R be the set of documents known (or initially estimated) to be relevant, and Ř be the complement of R and the set of non-relevant documents. Let $P(R|d'_j)$ be the probability that the document d_j is relevant to query q, and $P(\check{R}|d'_j)$ be the probability that d_j is non-relevant to q. The similarity $SIM(d_j,q)$ of document d_j to the query q is defined as:

$$SIM(d_j, q) = \frac{P(R|d'_j)}{P(\check{R}|d'_j)}$$

Using Bayes' rule:

$$SIM(d_j, q) = \frac{P(d'_j|R) * P(R)}{P(d'_j|\check{R}) * P(\check{R})}$$

where $P(d'_j|R)$ is the probability of randomly selecting document d_j from the set R of relevant documents, P(R) is the probability that the document randomly selected from the entire collection is relevant, and $P(d'_j|R)$ and $P(\check{R})$ are complementary (Drymonas, 2006).

Okapi BM25 (Garcia, 2011), developed in the 1970s and '80s by Stephen E. Robertson and Karen Spärck Jones, is one of the best probabilistic retrieval frameworks. It is also used by some search engines to rank matching documents to a given search query according to their relevance. BM25 is a 'bag-of-words' retrieval function that ranks a set of documents based on the query terms in each document, regardless of the inter-relationship between the query terms within the document (e.g., their relative proximity). It is not a single function, but a family of scoring functions with slightly different components and parameters. One of the most prominent instantiations of the function is as follows.

Given a query *Q*, containing keywords $q_1 ... q_n$, the BM25 score of a document *D* is:

$$score(D,Q) = \sum_{i=1}^{n} idf(q_i).\frac{f(q_i,\ D).\ (k_1+1)}{f(q_i,D) +\ k_1\ .\ (1-b+b\ .\ \frac{|D|}{avgdl})}$$

where $f(q_i,\ D)$ is q_i's term frequency in document *D*, $|D|$is the length of document *D* (i.e., number of words), $avgdl$ is the average document length in the text collection from which the documents are drawn, K_1 and *b* are free parameters usually chosen in the absence of an advanced optimization (e.g., $K_1 \in [1.2,2.0]$) and $b = 0.75$), and $idf(q_i)$ is the inverse document frequency weight of the query term q_i. It is usually computed as:

$$idf(q_i) = log\ \frac{N-n(q_i)+0.5}{n(q_i)+0.5}$$

where N is the total number of documents in the collection, and $n(q_i)$ is the number of documents containing q_i.

2.3.2 Linear Algebraic Models

Linear algebra is a branch of mathematics related to the study of vectors. Working according to certain rules, it generally uses families of vectors called vector spaces (or linear spaces), as well as functions that input one vector and output another. A vector space is a set of elements that can be added together and multiplied by scalars or numbers. If the functions work as intended they are called linear maps, linear transformations or linear operators, and can always be represented by matrices. Linear algebra is central to modern mathematics and its applications.

In this section, we discuss three alternative algebraic models: the generalized vector space model, the latent semantic indexing model and the neural network model.

2.3.2.1 Generalized Vector Space Model

The Generalized Vector Space Model (GVSM) extends the standard Vector Space Model (VSM) by embedding additional types of information, other than terms, in the representation of documents. An interesting type of information used in these models is semantic information, from thesauri such as WordNet[9]. Previous attempts to construct GVSM reported contradictory results. The most challenging problem is incorporating the semantic information in a rigorous manner, and modifying the standard interpretation of the VSM.

Fundamentally, this model is based on the idea that, in addition to exact keyword matching, there are at least two other basic directions for embedding term-to-term relatedness into a retrieval model: (a) by computing the semantic correlations between terms, or (b) by computing the frequency co-occurrence statistics from large corpora.

On the other hand, a GVSM extends the pairwise assumption. More specifically, it considers a new space where each term vector t_i is expressed as a linear combination of 2n vectors.

For a document d_k and a query q, the similarity function now becomes:

$$SIM(d_k, q) = \frac{\sum_{j=1}^{n} \sum_{i=1}^{n} w_{i,k} * w_{j,q} * t_i \,.\, t_j}{\sqrt{\sum_{i=1}^{n} w_{i,k}^2} * \sqrt{\sum_{i=1}^{n} w_{i,q}^2}}$$

where t_i and t_j are now vectors in a 2n-dimensional space.

The term correlation $t_i \,.\, t_j$ can be implemented in several ways. For example, Wong et al. (1975) used the term occurrence frequency matrix obtained from automatic indexing as inputs to their algorithm, and the term correlation between any pair of index terms as output.

2.3.2.2 Latent Semantic Indexing Model

Latent Semantic Indexing (LSI) is an indexing and retrieval method that uses a mathematical technique known as Singular Value Decomposition (SVD) to identify patterns in the relationships between the terms and concepts in an unstructured collection of text. LSI is based on the principle that words used in the same context tend to have similar meanings. A key feature of LSI is the ability to extract the conceptual content of a body of text by establishing associations between terms that occur in similar contexts. The main disadvantage of LSI is that it is based on matrix decomposition, and cannot be scaled up to very large collections of text such as web collection.

[9] http://wordnet.princeton.edu/

2.3.2.3 Neural Network Model

Neural networks are bio-inspired data processing mechanisms that enable computers to learn in a manner similar to the brain, and even to generalize once the solutions to sufficient problem instances are taught.

Neural Networks have been used in information retrieval in several ways: to implement the vector space model, to implement the probabilistic model, to implement relevance feedback, and to implement learning models to improve the other applications. Three types of nodes (i.e., query, term and document) are used to set up a neural network for information retrieval, and the links between these nodes are known as query-term links and document-term links. A query-term link indicates that the term occurs in the query, and the weight of the link is defined as the tf /idf score for the vector space model, or the score for the probabilistic model. Similarly, a document-term link indicates that the term is in a document, with an appropriate weight. A feed-forward network is employed, which activates a specific node whose output exceeds a given threshold (Vesanto and Alhoniemi, 2000). Technically, networks can only propagate information in one direction, or the information can bounce back and forth until self-activation at a node occurs, and the network settles on a final state. Thus, the process can be seen as the activation of a built-in thesaurus. As mentioned, the query-term nodes are assigned an initial (fixed) maximum activation level of 1. The query-term nodes then send signals to the document-term nodes, which are attenuated by normalized query term weights $\grave{W}_{i,q}$. For a vector-based ranking, these normalized weights can be derived from the weight $W_{i,q}$ defined for the vector model (Salton and Buckley, 1987), as follows:

$$W_{i,q} = \left(0.5 + \frac{0.5\ freq_{i,q}}{max_l\ freq_{l,q}}\right) * idf$$

where the normalization is done using the norm of the query vector

$$\grave{W}_{i,q} = \left(\frac{W_{i,q}}{\sqrt{\sum_{i=1}^{t} W_{i,q}^2}_{l}}\right)$$

Once the signal reaches the document term nodes, they could send new signals toward the document nodes. These signals are attenuated by normalized document term weight $\grave{W}_{i,j}$ derived

from the weights $W_{i,j}$ defined for the vector model below, which is the best known term-weighting scheme:

$$W_{i,j} = tf/idf$$

where the normalization is done using the norm of the document vector

$$\grave{W}_{i,j} = \left(\frac{W_{i,j}}{\sqrt{\sum_{i=1}^{t} W_{i,j}^2}_{l}} \right)$$

The signals that reach a document node are counted. Thus, after the first round of signal propagation, the activation level of the document node associated with a document d_j is given by:

$$\sum_{i=1}^{t} \grave{W}_{i,q} \grave{W}_{i,j} = \frac{\sum_{i=1}^{t} w_{i,q} w_{i,j}}{\sqrt{\sum_{i=1}^{t} w_{i,q}^2} * \sqrt{\sum_{i=1}^{t} w_{i,j}^2}}$$

which is equal to the ranking provided by the classic vector model. The experiment of using this model shows that there is no conclusive evidence that a neural network provides superior retrieval performance with general collections (Kriesel, 2005).

2.4 Mathematics behind Web Ranking

Nearly all the major Web search models use a standard vector space model (Wong et al., 1975), as well as link analysis to improve their search results. Link analysis on the hyperlink graph uses fundamental matrix theory. Link analysis and its underlying linear algebra have helped revolutionize Web search, and in this section we summarize the mathematical algorithms behind the most popular searching algorithms used or frequently mentioned by most researchers.

2.4.1 The HITS Algorithm

Hypertext Induced Topic Search (HITS), also known as hubs and authorities, is a topic-specific local ranking and link analysis algorithm, and a precursor to the Google-PageRank algorithm. It operates on a small number of the rates Web pages developed by Kleinberg (1999), where resources pertaining to a specific topic are likely to exist. A page is an authority on a topic if it contains high quality, useful information, and a page is a hub on a topic if it links to well-regarded authorities (i.e., it is a list of quality resources on the topic). Some pages serve as hubs

or portal pages, such as those with numerous out-links, while others are authorities on topics because they have many in-links. Good hubs seem to point to good authorities, and good authorities point to good hubs (Brin and Page, 1998) and (Baeza and Ribeiro, 1999). Each page 'i' has both a hub score h_i and an authority score a_i, and for every page 'i' there is a hub score at iteration k as $h_i^{(k)}$ and an authority score as $a_i^{(k)}$. To compute the ranking for each page p, the algorithm starts with uniform scores for all pages in the ranked list (i.e., *hub(p) = 1/n* and *authority(p) = 1/n*), where n is the number of pages in the so-called the neighborhood set for the query list. The neighborhood set consists of all pages in the query list, and all pages pointing to or from the query pages. There are two types of updates: authority-update rule and hub-update rule. To calculate the hub/authority scores of each node, repeated iterations of the authority-update rule and the hub-update rule are executed. A k-step application of the Hub-Authority algorithm involves applying the authority-update rule and then the hub-update rule, *k* times. The authority-update rule for a page *p* is:

$$authority(p) = \sum_{i=1}^{n} hub(i)$$

where *n* is the total number of pages connected to page *p,* and *i* is a page connected to *p*. That is, the authority score of a page *p* is the sum of all the hub scores of pages that point to it, whereas, the hub-update rule *hub(p)* is:

$$hub(p) = \sum_{i=1}^{n} auth(i)$$

where *n* is the total number of pages *p* connects to, and *i* is a page which *p* connects to. Thus, a page's hub score is the sum of the authority scores of all its linking pages.

2.4.2 The PageRank Algorithm

PageRank is the second link analysis algorithm proposed by Sergey Brin and Larry Page from Google (Brin and Page, 1998). It uses a recursive scheme similar to Kleinberg's HITS, and it defines a random walk with random jumps over the entire Web graph. The idea behind *PageRank* is that a page is important if other important pages point to it; that is, the importance of a page (the score) is determined by summing the PageRank of all pages that point to it, and reduced if these pages point to other pages. For example, if page 'yahoo.com' points to 999

pages, in addition to a particular page, then the particular page only gets credit for 1/1000 of the Yahoo.com page PageRank.

Like HITS, PageRank starts with a uniform rank for all pages (i.e., $r_i^{(0)}$ = 1/N) and successively refines this score; N is the total number of Web pages in the graph. To allow for random clicks on links, a dumping factor is used to adjust the derived value downward (usually d=0.85). The equation is as follows:

$$PR(P_i)=\frac{1-d}{N} + d \sum_{Pj \in M(Pi)} \frac{PR(P_j)}{L(P_j)}$$

where $P_1, P_2 ... P_N$ are the pages under consideration, $M(P_i)$ is the set of pages that link to P_i, $L(P_j)$ is the number of outbound links on page P_j, and N is the total number of pages. Figure 2.2, shows an example of PageRank document ranking.

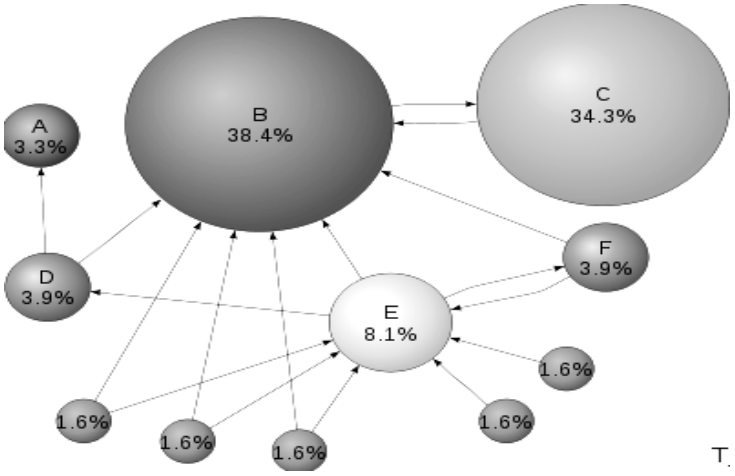


Figure 2.2 An example of PageRank algorithm (Google)

2.5 Models behind Web Page Topic Finding

Web page indexing refers to various methods of indexing the contents of web pages. Search engines usually use keywords and meta-data to provide more useful vocabularies for indexing the pages. Meta-data web indexing involves assigning keywords or phrases to web pages or web sites within a meta-tag field, so the page or site can be retrieved with a search engine that is customized to search the keyword's field. This may or may not involve using keywords restricted to a controlled vocabulary list, and is commonly used for search engine indexing. While recent research has focused on indexing Web pages based on information other than the text appearing in the documents, in our approach we used the information available in Wikipedia

to help find the tasks involved in the documents. So, many documents that share the same topic make the collective information more extensive than when restricted to one document. In this section, we review several methods related to the generic topic finding.

2.5.1 Using Web Graph

As discussed in the previous section, link analysis is used by many major search engines, including Google. Link-based retrieval extracts latent information from web graphs (link structure) and describes how the documents connect. In other words, all of the link-based ranking algorithms proposed so far operate by first assigning numerical scores to web pages, and then ranking the pages by the scores.

Ranking Web pages is not limited to hyperlink structure, and combining anchor text with the hyperlink structure is a novel idea that extends the HITS algorithm. Topical Hypertext-Induced Topic Selection (TOPHITS) (Kolda et al., 2007) was a new methodology that used multi-linear algebra to elicit more information from a higher-order representation of the hyperlink graph. It initially labels the edges in the graph with prominent topics (using the anchor text of the hyperlinks), so that the associated linear algebra representation is a sparse three-dimensional tensor. The first two dimensions of the tensor represent the authority scores and hub scores of Web pages, and the third dimension adds the anchor text (topic score). The tensor is computed by counting the number of links from host i to host j with term k, and storing the result as C_{ijk}. Figure 2.3 shows an example of anchor texts used to point to titles of other documents; thus the topic tensor is computed for each document by the number of links that point from other documents.

$$h_i^{(t+1)} = \sum_{i \xrightarrow{k} j} a_j^{(t)} t_k^{(t)} \qquad \text{For j=1… n}$$

$$a_j^{(t+1)} = \sum_{i \xrightarrow{k} j} h_i^{(t+1)} t_k^{(t)} \qquad \text{For j=1… n}$$

$$t_k^{(t+1)} = \sum_{i \xrightarrow{k} j} a_j^{(t+1)} h_i^{(t+1)} \qquad \text{For j=1… n}$$

Here, the notation $\mathrm{i} \xrightarrow{\mathrm{k}} \mathrm{j}$ means page i links to page j with anchor text k.

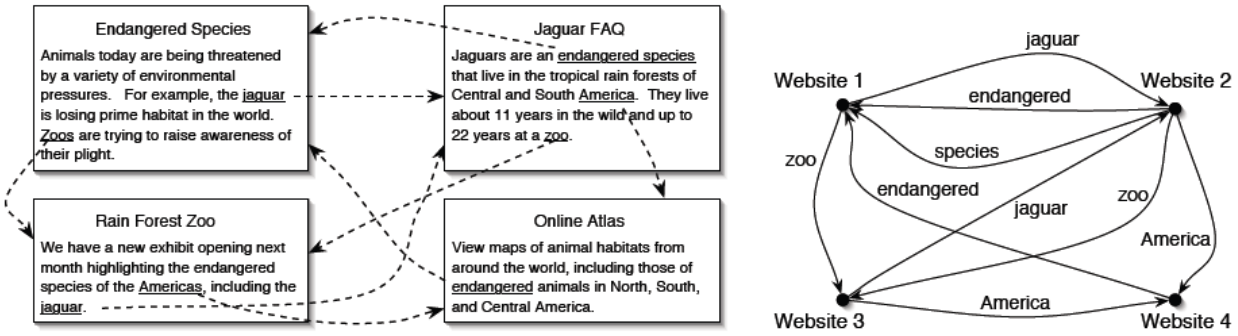


Figure 2.3 Hubs, Authorities, and Topic Tensors (Kolda et al., 2007)

The algorithm only considers host-to-host links (rather than page-to-page links), and removes all self-links that point from a host to the same host.

One factor that makes link-based search algorithms realistic methods is that the ranked pages are stable even if minor changes occur in the link structure of the input graph. The rank-stability and rank-similarity of the page ranking in authority-connected graphs was confirmed by Lempel and Moran (2005). This means that if some ranked documents were deleted from the final result (e.g., 10%), the rank was not affected.

Gurrin and Dmeaton (2003) experimented with this algorithm in TREC 2003, using the SPIRIT-Web collection of 94,552,871 documents.

Results from the TREC 2002 web track suggest that an average of 1.98 off-site in-links for each document is enough for improvements in retrieval performance to begin to show. In our approach we normalized the HIST algorithm for ranking the pages in our collection of 500 million documents, using their anchor links (this is explained in Chapter 6).

2.5.2 Using Meta-Content

Although anchor text provides very useful information for web searching, a large portion of web pages have few or no incoming hyperlinks (anchors); this is known as the anchor text sparsely problem. Xing and James (2010) proposed a language model based technique for overcoming anchor text sparsely: discover a web page's plausible missing anchor text from similar web pages' in-link anchor text. Automated search engines that rely on keyword matching and support text-based searching usually return too many low quality matches due to the enormous amount of data available on Web, and the index becomes quite expensive to maintain and update frequently. Thus, meta-information is preferable for indexing.

Extracting data from document meta-content is one such type of indexing method. Li (2003) extracted data from the tags ‘title’, ‘anchor’, ‘keywords’, description’, and ‘paragraph’, and concluded that this approach uses significant CPU time because the inverted index is structured around single words, without consideration of phrase queries or semantic and linguistic aspects of the queries.

The importance of URLs in web topic finding was demonstrated by Baykan et al. (2009) and Song et al. (2006). These methods were based on extracting features from URL structures. The features provide valuable information about the document, and the required computation is relatively inexpensive. The researchers experimented with two methods for extracting the features from the URLs: whole tokens and letter $n-grams$ of tokens. Experiments with this method typically achieved precision of approximately 85% (n-grams of tokens).

2.5.3 Using Meta-Search and Fusion

The previous methods, based on a single source of search, were not usually powerful enough to retrieve relevant information, and they could even decrease retrieval performance in some cases (Yang, 2002). Meta-search is another technique that has been used recently by many researchers. This method applies a search tool that sends a user request to several other search engines and/or databases, aggregates the results into a single list, and displays them accordingly.

Experiments performed by Selberg (1999) showed that no single Web search service provides comprehensive searching, in terms of calculating the Unique Document Percentage (UDP) for each search engine. The researcher’s reasonable assumption was that if a document can be found through a search service, and it does not change, then it can be found through that search service again. *Husky* is an example of collaborative index enhancement, as it is a meta-search service that contains no index of its own; its indexes can be considered a union of the indexes of the services they queried previously. The Returned-URLs and Clicked-URLs indexes are used to return useful documents not returned by other search services in response to a user query, and to increase the ranking of useful documents returned by other search services.

Documents that are viewed often should be ranked higher, and those that are frequently returned but never viewed should be ranked lower. This method addresses the problem of unstable searching. The Clicked-URLs index provides a similar function, but only for documents that are viewed by users to retrieve the following information: the query it was associated with, its ranking and the time required for processing. When two or more search engines returned the

same document, the confidence score of the document was the sum of each service's score. They used eight popular search engines in their experiments.

Several researchers achieved improved effectiveness by combining the results of different retrieval strategies and query representations. Each retrieval strategy should retrieve different sets of relevant documents, and combining the results could lead to a better result than individual strategies. This is somewhat intuitive, and many meta-search engines on the Web were developed to capitalize on the concept of fusion providing improved effectiveness. However, it is difficult to assess the effectiveness of meta-search engines, because they are not typically run against a standard document collection with known relevant results.

Fusion of retrieval results from different models to improve retrieval performance was reported in works such as Vogt and Cottrell (1999), McCabe and Chowdhury (1999), Montague and Aslam (2002) and Voorhees et al. (1994). Performance improvements were also reported by Fox and Shaw (1994), and by Kolda et al. (2007).

Vogt and Cottrell (1999) concluded that five characteristics are required for effective fusion: at least one result with high precision/recall, high overlap of relevant documents, low overlap of non-relevant documents, similar distribution of relevance scores, and that each retrieval system ranks relevant documents differently.

2.5.4 Using Document Evidences

Document topics finding using web evidence could likely improve retrieval effectiveness in navigational search tasks. Such methods are based on combining several tasks in document retrieval methods that used specific evidence. The most effective combination of evidence methods treat document-level and web-based evidence as separate components, using a linear combination to sum the scores. Ranking documents using one evidence can lead to inaccurate results, because text-based approaches have difficulty dealing with the vocabularies (i.e., different expressions of the same concept in Web documents and queries), the diversity of document quality and content, fragmented documents, and documents with minimal textual content (e.g., hubs, index pages, bookmarks). Link-based approaches do not perform well with a variety of link types (e.g., citation links, navigational links, commercial links and spam links), or emerging communities with incomplete link topologies.

As well as classification and vocabulary problems (i.e., different categorizations of the same Web document and different labeling of the same category), Web directories contain only a

small percentage of the documents. Yang (2002) showed the complementary strength in the combination of text, link and Web directory information (i.e., Yahoo) by using: a text-based approach, the Vector Space Model, the Link-based approach and the classification-based approach. Thus, fusion could be beneficial and viable, and could improve retrieval effectiveness for Web IR. However, recent TREC experiments that combined text-based and link-based methods did not find fusion improved retrieval performance. Since Google encompasses a fusion method[10], it seems that fusion does not work well in the Web context.

Upstill et al. (2003) demonstrated how web evidence can be used to improve the retrieval effectiveness of navigational search tasks. The researchers discussed some important evidence that was classified into two types, Web Evidence and Document Evidence. The first Evidence emphasized: (i) the full text, title, URL length, anchor text and in-degree evidence, (ii) finding a home page using query-independent web evidence, (iii) combining query-independent web evidence with query-dependent evidence, (iv) analysis of hyperlink evidence and (v) other evidences (including URL information, meta-data, document structure, tag information and quality metrics).

The second evidence emphasized: (i) Boolean matching, (ii) vector space model, (iii) probabilistic ranking and (iv) statistical language model. According to their study, using anchor-text evidence to rank web documents, rather than document full text, seems to provide significant effectiveness gains in Home Page Finding and Topic Distillation tasks. Removing aggregated anchor-text length normalization altogether, or normalizing according to full text document length, was found to improve retrieval effectiveness. The most effective use of hyperlink scores was reducing the corpus size without reducing home page search performance. Document evidence should include full-text evidence and other useful document-level evidence, and Web-based evidence should use incoming anchor-text and other useful external document descriptions. For a home page search, a URL depth component either measured by characters or classified by type ought to be included. This could be done either by re-ranking documents that are within n% of the top score of the URL length, or by adding a normalized URL length score to the query-dependent score. The results of this experiment demonstrated: (i) the importance of both anchor-text and URL length measures in home page finding tasks, (ii) both PageRank and in-degree performed similarly and were highly correlated, (iii) using anchor-text evidence to

[10] http://www.google.com/fusiontables

rank documents, rather than document full-text, provided significant effectiveness gains in home page finding and Topic Distillation tasks and (iv) hyperlink recommendation evidence was far less effective than URL-based measures.

Plachouras (2006) and Amati et al. (2003) demonstrated the importance of using document evidence. Selective Web IR explicitly aims to predict the most effective retrieval approach from a set of at least two available approaches $\mathrm{MAX}(a1, a2 \ldots)$. The retrieval approaches used field-based weighting models that perform term frequency normalization, then weighted each document field, both independently and in combination, with the query-independent evidence from the URLs of Web documents (URL path length), PageRank, the Absorbing Model and a hyperlink structure analysis algorithm (Web graph). The experiment showed that the URL path length was particularly effective for topic distillation tasks, while both the URL path length and PageRank resulted in considerable improvements in the retrieval effectiveness of the home page finding tasks. With respect to the named page finding tasks, the most effective query-independent sources of evidence were PageRank and the Absorbing Model (Amati et al., 2003).

2.5.5 Using User Preferences

Another possibility for finding document topics is to use a user preferences model. Most-search engines apply explicit user preferences in the form of category-specific web search, to identify useful documents and enhance the performance. Two users with the same query, but different preferences, might search different sources with different (modified) search engine requests, and the results might be scored differently. Glover (2001) incorporated a method for learning search-engine and query-modification pairs to facilitate category-specific web search, known as Query Modification Learning Procedure (QMLP). The results of the procedure were used to generate several categories. The study quantified the significance of user categories for differentiating between highly topically relevant but useful documents, and highly topically relevant not useful documents identified as ‘not topically relevant’, ‘a little topically relevant’, ‘somewhat on topic’, ‘very topically relevant’ and ‘exactly the right topic’. Such techniques included an incremental user interface, need-based source selection, source and category-specific query modification, selective downloading of results and need-based scoring. For each category of processing, the experiment started with a set of training and testing documents and the set of sample queries, and used four search engines (i.e., ‘AltaVista’, ‘AllTheWeb’, ‘Northern Light’ and ‘Google’). An SVM classifier was used in each case.

The researcher concluded that a query modification with 100% precision that only returns three results is likely too restrictive, and probably worse than a query modification that results in 90% precision, but returns 10,000 results.

2.6 Enhancing Topic Finding

Clustering Web search results is another method for topic finding. Clustering techniques are used to group similar documents together, to facilitate the presentation of results in more compact form. Clustering essentially involves partitioning a given set into disjoint subsets. There are many ways to cluster documents, including K-means, PAM, Hierarchical Clustering and OPTICS. In this section, we briefly summarize some of these methods.

2.6.1 Web Search Results Clustering

This was first introduced with the Scatter/Gather system (Cutting et al., 1992), which was based on a variation of the classic K-Means algorithm, and followed by Suffix Tree Clustering (STC) proposed by Zamir (1999). In this algorithm, snippets that share the same sequence of words should be grouped together. Since the clustering algorithm was focused on online Web search results, it was fast and of high quality. STC relies on a data structure for string matching (finding the longest repeated sub-string), and querying to efficiently identify sets of documents that are internal nodes of the tree that shares common phrases. This information was used to create clusters, and to succinctly summarize their content for users. Clustering by phrases is better than the bag-of-words method for word proximity and, moreover, constructing concise and accurate descriptions of clusters can improve the quality of the clusters.

If a document appears in more than one cluster, the overlap between clusters should occur but not be too high (it depends on a threshold), otherwise clusters would merge into a single cluster. However, the experiment improved the average precision by using clustering at indexing time (dynamic index), and it was recommended for creating an inverted index file to identify common phrases; not only of words, but of 2-grams and 3-grams as well.

Bisecting K-means (Weiss, 2006) is a more efficient version of the K-means algorithm. It starts with a single cluster of all the documents, then at each iteration step it chooses a cluster to split into two sub-clusters, using the basic K-means algorithm, and repeats this until the desired number of clusters is reached. The largest cluster is selected to be split, where the size of a cluster is defined by the linear combination of convergence and diameter measures as follows:

$$Size_c = k_1 \times Convergence(C) + k_2 \times Diameter(C)$$

where $Size_c$ denotes the size of cluster *C*, Convergence is the average distance between every two documents in the cluster, and Diameter is the longest (max) distance among the documents within the cluster.

There are several methods that can be used to calculate the distance between clusters; the cosine distance was used in this algorithm. First, all the documents in each cluster were sorted by the distance between document and the center of the cluster, in ascending order. Then the clusters were sorted by the number of documents they contained, in descending order.

Description-Comes-First (DCF) is another method (Weiss, 2006). With this approach, the process of clustering is changed by first finding meaningful cluster labels, and then assigning snippets to them to create proper groups. The Lingo and Suffix Tree Clustering (STC) algorithms (Osinski and Weiss, 2005) and (Weiss, 2001) were used to ensure that good quality label clusters had the following five characteristics: accuracy, conciseness, unambiguity, diversity, and distinctness. Lingo performs clustering in two major phases, cluster label induction and cluster content assignment. During the cluster label induction phase, the term-document matrix of all input snippets, as well as the snippets' frequent phrases, were used to discover labels for not yet existing clusters; Linear Algebra reduces the dimension, which facilitates finding the labels. In the cluster content assignment phase, snippets were allocated to previously discovered labels, in order to create the actual groups. The simplest method here applied the Vector Space Model, and used cluster labels as queries against the input snippets.

Unlike the previous methods, Garcia (2011) took the relation between clusters into account. The algorithm integrated the DCF algorithm with several dimensionality reduction techniques (matrix factorizations as two variants of Non-negative Matrix Factorization and Local Non-negative Matrix Factorization). The clustering scheme relied on a special collection of texts created by merging smaller document groups, each of which related to some well-defined topic. Three cluster evaluation measures were used in the evaluation methodology: Cluster Contamination, Topic Coverage, and Snippet Coverage. The experiment used the Open Directory Project (ODP)[11] as the test dataset. There was a transparent relationship between the cluster label and its content (Weiss, 2006). The algorithm integrated the Description Comes First (DCF) with the Descriptive K-Means, and it meet the requirements of descriptive clustering. The

[11] http://www.dmoz.org/

main idea of this method was to separate the selection of candidate cluster labels from cluster discovery, and it was achieved in three phases: Cluster label Discovery, Dominant Topic Detection and Pattern Phrase Selection and Document Assignment. The purpose of this step is to find the cluster label candidates that were most similar to the representation of previously discovered dominant topics (pattern phrases). Once the pattern phrases have been identified, the representation of dominant topics was discarded and replaced by the pattern phrases as the seeds of final document groups. The last step (pruning) was done to remove any pattern phrases which failed to collect enough documents.

However, many objectives were not met in the experiments. One of these was a combined approach that could extract frequently occurring terms, filter out the junk, and then put the terms back in the right order. This would be a fast and accurate candidate cluster extraction method, but using predefined cluster labels from ontology seems to be more realistic. As well, there are many semantic networks of relations on the Web, and this knowledge could be used to create a set of comprehensible cluster labels candidates.

2.6.2 Using Latent Semantic Indexing (LSI)

Latent Semantic Indexing (LSI) is an attempt to capture some semantic aspects. Similar documents could have very few words in common, and thus the previous methods cannot identify them as similar. They do not take word senses into account, and fail if there is no exact match with the query words. LSI considers documents to be semantically close even when they do not share the same words. Relevance feedback, spam document flagging and combining documents based on human generated taxonomies are also very important aspects of recent studies.

Automating, clustering, and organizing the web pages according to the page content are important (Manning et al. 2008). In 2009, Majavu (2009) and Majavu et al. (2008) improved document clustering by using LSI. Sensor Web is a global sensor system that implemented a modified LSI by making use of ontology to classify Web Resident Resources found in geospatial web portals. The algorithm was based on combining LSI with ontology concepts of the topic of interest, resulting in a modified LSI. It effectively extracts knowledge web pages to separate those that may contain links to sensor resources, as well as irrelevant pages in the geospatial portal.

A semantic-based, grouping snippet proposed by Zhao et al. (2007), is another algorithm which used the semantic electronic dictionary WordNet[12]. The query ran in parallel, by finding results from any search engine, as well as synonym sets (synsets) and the semantic relationship between them. The Search Result Records (SRRs) are grouped by the senses of the query terms. Two SRRs that talk about very similar topics but use different words are similar if their words have shared synsets, while other SRRs might have common words but less similar topics if their words are in different synsets. The Okapi BM25 formula (Garcia, 2011) was revised to compute the similarity scores, and the experiments showed that for queries with multiple terms, the specific meaning of each term was easier to determine, because terms in the same query can provide the context. Then, it achieved an accuracy of approximately 90%. The researchers used recall, precision, and F-measure in their experiment.

2.6.3 Using Feature Extraction

Web pages belonging to one particular category have some similarity in their structure. Document structure is needed to classify a huge repository of information (Asirvatham and Ravi, 2007). Feature extraction is the first step of document classification/categorization; they can be extracted from images, videos (multimedia content), document structure, and web text content. Any web page can be categorized into three broad types: information, personal home page, and research. Information pages typically have a logo at the top, followed by a navigation bar linking the page to other important pages. Personal home pages also tend to have a common layout. For example, the name, address, and photograph of the person might appear prominently at the top of the page, and they could provide links to their publications or contact details toward the bottom of the page. Research pages generally contain huge amounts of text, equations, and graphs in the form of images and such, and a set of features can be used to capture the structure and classify a page by analyzing the placement of text, links, images and other items. Such features are known as ‘Textual Information’ and ‘Image Information’. Each feature contributes, positively or negatively, to a page being classified as belonging to a particular category. Asirvatham and Ravi (2007) proposed that a feature value was proportional to the number of times it was present in a document. This method was based on heuristic assumptions to assign different weights to each feature of different categories. Pages containing less than 200 characters were excluded from the classification, and under this assumption the following formula was used.

[12] http://wordnet.princeton.edu

$$W = \begin{bmatrix} W_{00} & W_{01} & - & - & W_{0c-1} \\ W_{10} & W_{11} & - & - & W_{1c-1} \\ - & - & - & - & - \\ - & - & W_{ij} & - & - \\ W_{n-10} & W_{n-11} & - & - & W_{n-1c-1} \end{bmatrix} \quad F = \begin{bmatrix} F_{00} \\ F_{01} \\ - \\ F_{0i} \\ - \\ F_{0n-1} \end{bmatrix} \quad V = \begin{bmatrix} V_{00} \\ V_{01} \\ - \\ V_{0i} \\ - \\ V_{0c-1} \end{bmatrix}$$

where W→[n x c] is a matrix of weighted features, F→ [n x 1] is a vector for feature counts, V=W*F → [c x 1] is a final result, c= number of categories, n=number of features, W_{ij} →[-1, +1] is the weight assigned to the i^{th} feature to contribute to the j^{th} category, and F_{0i} counts the features present in the document. However, feature extraction is not suitable method if used in large corpus; therefore, in this experiment the researcher used a small dataset of 4000 pages.

2.7 Entity Finding Task

Search engine functionality includes many tasks, one of which is people search (information about people), in which the name homonymy is a problem that needs to be addressed. As the amount of information available on the Web increases, a higher number of people are mentioned in different web pages. According to data from the U.S Census Bureau in 1990, approximately 90,000 different names were shared by 100 million people (Gao, 2010), and three of four users are seeking information on non-celebrities. Also, search engines are likely using biased searches to find relevant names. For example, a query for 'Sharon Goldwater' returns 165 pages related to a recognized researcher, while a music critic is in position 166 in the ranked list of the search engine. Even common names can be shared by many celebrities or historical figures at the same time. In these cases, large amounts of information are available for each individual.

Finding information about people has a potential use in informative 'Friend of a Friend' (FOAF) meta-data regarding the Semantic Web and Social Networking. Keyword extraction is one approach of Entity Finding to extract person information (meta-data) from the web by determining the context of relations; as a result, this is different from meta-data tags and RDF (Resource Description Framework) meta-data, such as name, project, conference, workshop and organization. To some extent, the algorithm depends on the scoring of initial pages by a certain search engine, and then tracking the links in the relevant pages. The idea behind this method is if two terms co-occur in many pages, the terms have a strong relation; and also, if two terms are

similar for different persons, then the different pages that contain the two terms may reflect the different context of a person (Aslam and Montague, 2001). The relevance between the name (N) and term (w) is computed by the Jaccard Coefficient (Manning et al., 2008), which captures the degree of co-occurrence of two terms by the degree of overlapping between them. The relevance of person *N* and term *w* in the context C, denoted by Score(N, C, w), was calculated as follows:

$$\mathrm{Score}(\mathrm{N},\mathrm{C},\mathrm{W}) = \frac{\mathrm{r}(\mathrm{N},\mathrm{W})}{\mathrm{MAX}(\mathrm{r}(\mathrm{N},\mathrm{W}))} + \alpha\frac{\mathrm{r}(\mathrm{C},\mathrm{W})}{\mathrm{MAX}(\mathrm{r}(\mathrm{C},\mathrm{W}))}$$

where the alpha weight denotes the relevance r between the person and the context (e.g., it can use $\mathrm{r}(\mathrm{N},\mathrm{C})$ as alpha). $\mathrm{MAX}\big(\mathrm{r}(\mathrm{N},\mathrm{W})\big)$ is the maximum value of the Jaccard coefficient in the term set W. A threshold for $\mathrm{r}(\mathrm{N},\mathrm{w})$ can be used to exclude terms that are not relevant for a person, but have a strong relation to the context. The term w with the higher $\mathrm{score}(\mathrm{N},\mathrm{C},\mathrm{w})$ is considered to be a more relevant keyword for person N in context C. The researchers also considered the relation between two persons (N1, N2) in terms of their contexts, as one person can be regarded as a part of the context of another person.

$$\mathrm{score}(\mathrm{N1},\mathrm{N2},\mathrm{w}) = \frac{\mathrm{r}(\mathrm{N1},\mathrm{w})}{\mathrm{MAX}(\mathrm{r}(\mathrm{N1},\mathrm{w}))} + \beta\frac{\mathrm{r}(\mathrm{N2},\mathrm{w})}{\mathrm{MAX}(\mathrm{r}(\mathrm{N2},\mathrm{w}))}$$

This formula shows the relevance of person *N1* for term *w* in relation to person *N2*. β is the parameter that weights the relevance between persons r(N1,N2).

Moreover, because there are many contexts to a person, the relations among persons have a variety of contexts. For example, the relation of two persons in the academic field (e.g., co-authors) has the same affiliation, the same project, and they may even be friends. The relevance of person N term w in relation to person N2 in the context C, $\mathrm{score}(\mathrm{N1},\mathrm{N2},\mathrm{C},\mathrm{w})$ is given as follows:

$$\mathrm{score}(\mathrm{N1},\mathrm{N2},\mathrm{C},\mathrm{w}) = \mathrm{score}(\mathrm{N1},\mathrm{N2},\mathrm{w}) + \gamma\frac{\mathrm{r}(\mathrm{C},\mathrm{w})}{\mathrm{MAX}(\mathrm{r}(\mathrm{C},\mathrm{w}))}$$

where gamma is the parameter weighting the relevance between the persons and the context, r(N1 and N2,C). While the method applied tf/idf for keyword extraction, and 3981 web pages were used in the experiment, two problems were not addressed: meta-data was not extracted, and though certain keywords properties were labelled, the process was not efficient for storing properties for a large number of extracted keywords.

Weiss (2006) proposed grouping the personal pages from returned citations of each name, in terms of web classification and categorization. Three facets (evidence in citations) were used in

the construction of three confidence matrices (attributes, links, and page similarity), which reflected the shared information between two citations. The information included: name, phone, email, city, zip, state, URL and shared words. The final confidence matrix was the result of grouping the matrices using the Stanford certainty theory. If we consider two citations, C1 and C2, which refer to the three facets A, L and P, then the combination is: $\mathrm{FCM} = (\mathrm{A} + \mathrm{L} + \mathrm{P}) - (\mathrm{A} * \mathrm{L}) - (\mathrm{A} * \mathrm{P}) - (\mathrm{L} * \mathrm{P}) + (\mathrm{A} * \mathrm{L} * \mathrm{P})$. Normalized Split and Merge scores were used in the experiment for ten arbitrary names, and it was found that no individual facet scored better than using all the facets together. Distinguishing between general proper-noun queries such as 'John Smith', and actual noun queries such as 'Stephen Liddle', could have been useful, but this was not addressed.

Pic´on (2009) proposed another method, based on finding the optimal query refinements (i.e., number of words added to the person name) that lead to near optimal precision and recall, in order to help users find all the relevant information about the person they are querying. The Word Sense Induction task (WSI), also known as Word Sense Discrimination (WSD), involves discovering the senses of a word in given context, without using a dictionary to guide the disambiguation process. Semantic clustering techniques typically have a predominant role in both Web Person Search (WePS) and WSI. Token-based features were considered, including document full text tokens, lemmas, title, snippet, emails, outgoing links found in the other web pages, two Boolean flags that indicate whether a pair of documents is linked or belongs to the same domain, and URL tokens (tokenized using non-alphanumeric characters as boundaries). The experiment used n-grams (2-5) in length, and named entities extracted by the OAK tool for features extraction. The experiment found there was a trade-off between precision and recall. For example, token-based features such as lemmas and tokens, result in high recall and overly low precision; whereas URL and title result in high precision and low recall. Analysis of word of n-grams showed that an increase in the number of grams results in an increase in the precision and a decrease in the recall, and vice versa. The experiment also showed that named entities were not necessarily more useful than other features such as word n-grams. Finally, Pic´on (2009) recommended other directions that were not explored: features weighting techniques, searching the organizations, and addressing the relations between clustering and attribute extraction in an integrated task.

2.8 Query Expansion

Without detailed knowledge of the collection makeup and the retrieval environment, most users find it difficult to formulate well-designed retrieval queries, and they often spend a lot of time reformulating their queries to achieve effective retrieval. This suggests that the first query formulation should be treated as an initial attempt to retrieve relevant information, after which the initially retrieved documents are examined for relevance, and new query formulations are constructed to possibly retrieve additional useful documents. Such query reformulation involves analyzing words or term co-occurrences, expanding the original query with new terms, and reweighing the terms in the new documents with the expanded query. In this section, we summarize the three approaches for improving the performance of IR through query expansion and term reweighing:

- approaches based on feedback information from the user (a large collection of queries);
- approaches based on information derived from the set of documents initially retrieved (top-ranked) document; and,
- approaches based on global information derived from the document collection.

Automatic expansion techniques that use a general thesaurus, such as *WordNet*, have not proven effective. The key to optimal expansion is to choose words that are appropriate for the context or topic of the query (Billerbeck and Zobel, 2004).

2.8.1 Query expansion based on user feedback

Recent studies and experience have shown that the best way to capture query context is to use a query log, and that a large query log is the best resource for query expansion. Log files contain not only short pieces of text but other data as well, such as information about which documents were accessed during the search. As an example of how a query log can be used for expansion, the following are the ten most frequent words associated with queries that contain ‘tropical fish’ in a recent query log sample from a popular web search engine: stores, pictures, live, sale, types, clipart, blue, freshwater, aquarium and supplies. The types of queries most submitted about tropical fish are sales, supplies and pictures, thus these words are most suitable for expanding the query ‘tropical fish’ (Billerbeck and Zobel, 2004).

2.8.2 Query expansion based on local context analysis

This method is based on the local clustering techniques related to the set of documents retrieved by the original query; the top ranked documents are used for clustering the neighborhood terms. Such clustering depends on the term co-occurrences inside documents, and terms that are best neighbors for each query term are then used to expand the original query.

First proposed by Rocchio (1971), this method was originally implemented with manual relevance feedback; that is, after a user judged the relevant documents, they used the vector space model and modified the query vector to add the sum of the vectors for all relevant documents, normalized by dividing by the number of documents. Rocchio used a similar process for documents judged non-relevant, with the sum of the vectors of the non-relevant documents subtracted from the resulting query vector. To adjust the impact of the feedback, weights were added to give the original query terms the highest weights, as well as positive and negative weights. The process could be repeated with the results of the query as many times as required. Rocchio's formula for relevance feedback is:

$$q_m = \propto q_0 + \beta \frac{1}{|D_r|} \sum_{d_j \in D_r} d_j - \gamma \frac{1}{|D_{nr}|} \sum_{d_j \in D_{nr}} d_j$$

where q_m is the modified query, q_0 is the original query vector, D_r and D_{nr} are the set of known relevant and non-relevant documents respectively, and $\propto$,β, and γ are weights attached to the terms. Rocchio focused on the impact of the relevant and non-relevant document weight multipliers β and γ.

Subsequent work found that the best results were achieved when only one non-relevant document was subtracted from the query vector. Nonetheless, many systems eliminate the negative component altogether and only add terms with positive weights. For example, Salton and Buckley (1971) showed improvements with relevance feedback in the SMART system by setting the weight of non-relevant documents to zero, but the improvement was less than when the setting was β=0.75 and γ =0.25.

It has been shown that the best results were achieved when using only some of the terms from the top documents. For example, the top terms (based on normalized IDF, or another measure for term quality) from the relevant documents were added to the original query terms, and the query was then rerun. Buckley added the 50 most frequently-occurring single terms, and 10 phrases from the top 20 documents; the terms in the query were re-weighted using the

Rocchio's formula with $\propto = 1$, $\beta = 1$, and γ=0. Many approaches to weighting the feedback terms were introduced; feedback term weights are often discounted by a factor such as 0.5. The INQUERY system has a weighting scheme based on the rank of the feedback term in the list of possible feedback terms (the lower ranks get the highest discounts). Harman (1995) experimented with term re-weighting in query expansion, feedback term selection techniques or sort orders, and the effectiveness of performing multiple iterations of relevance feedback, and determined that using only selected terms for relevance feedback produced better results than using all the terms from the relevant documents. This researcher also found that using a feedback term sort order that incorporated information about the frequency of the term in the document, as well as the term's overall collection frequency, produced improved results. Landquist et al. (1997) also showed that using the best 10 to 20 terms and phrases from the top 5 to 20 documents, with a scaling factor of 0.5 for the terms added to the query, performed best. Imran and Sharan (2009) applied a co-occurrence measure of terms to local feedback, and took advantage of both global and local analysis.

Huang et al. (2007) used local context information to modify the weights of the query terms without adding additional query terms, and Amati (2003) implemented local context analysis for query expansion.

The advantages of local context analysis are that it is computationally practical, and once the top ranked passages are available, the query expansion is fast. The disadvantage is that it is based on the hypothesis that a frequent term from the top-ranked relevant documents will tend to co-occur with all query terms within the top-ranked documents, which is not always the case.

2.8.3 Query expansion based on global information

Global techniques use global information to build a thesaurus that identifies term relationships in the whole collection. The terms are treated as concepts, and the thesaurus is viewed as a concept relationship structure.

The building of a thesaurus typically involves using small contexts (synonyms and related words) and phrase structures, rather than simply adopting the context provided by a whole document. This is very important, because collections have usually been manually indexed (tagged) using the terms in the thesaurus. Since the terms in the thesaurus were chosen carefully, and were subject to quality control (controlled vocabulary).Bby using the thesaurus, a system could determine what words and phrases were used in queries, and expand an initial query using

synonyms and related words. In our framework, we used this approach to deal with complex queries.

2.9 Text Retrieval Conference (TREC)

The annual Text Retrieval Conference (TREC)[13] is an opportunity for organizations with an interest in information retrieval research to take part in a coordinated series of experiments, using the same experimental data. The goal of the conference series is to create the infrastructure necessary for large-scale evaluation of information retrieval systems, and thereby foster research into effective techniques for information access.

TREC is co-sponsored by the National Institute of Standards and Technology (NIST) and the U.S. Department of Defense, and began in 1992 as part of the TIPSTER Text program[14]. Its purpose was to support research in the information retrieval community, by providing the infrastructure necessary for large-scale evaluation of text retrieval methodologies.

TREC is overseen by a program committee made up of representatives from government, industry and academia. For each TREC, NIST provides a dataset, training queries and relevant judgment files[15] for training the proposed systems. In the testing session, the participants are provided with another set of 50 testing queries only, using the same dataset. Without the relevant judgment file, participants run their own retrieval systems on the data, and return a list of the retrieved top-ranked documents for each query to NIST, where the individual results are pooled, the retrieved documents judged for correctness, and the results evaluated. The TREC cycle ends with a workshop/forum, to allow participants to share their experiences and determine the best approaches.

This evaluation venture has experienced annual growth in the number of participating systems and complexity of the tasks. Clarke et al. (2012) improved the complexity of their Web retrieval track at TREC.

2.9.1 Training and Testing Collection

A basic requirement of evaluating search engines is that the results of different techniques need to be comparable, and to make comparisons fair the experimental settings and data used must be

[13] http://trec.nist.gov/
[14] The TIPSTER Text Program was run by the Defense Advanced Research Projects Agency (DARPA).
[15] A file that composited by TREC professional judgers and contains the relevancy results for each query.

fixed; the test collection, queries and relevance judgments[16] for a particular search task must be collected. Test collections have changed over the years, to reflect changes in data and users' needs for typical search applications; for example, the Web collection (GOV2) was crawled in 2004 with a size of 426GB, whereas ClueWeb09 was crawled in 2009 with a size of 25TB. ClueWeb09 consists of Web data in 10 languages, and involves one billion documents (500 million documents in English language and the balance in other languages). This collection is split into two categories: 'category A' represents 10 segments with each segment holding 50 million documents in English, and 'category B' represents a subset of 'category A' (segment 1) holding 50 million documents. The total number of websites in category A is 4,780,950,903, and the total number of websites in the subset B is 428,136,613.

Experimentally, using the full collection does not necessarily improve the retrieval efficiency; as shown in in the evaluation for each approach, set A or subset B produces highly relevant results. Thus, our system will use subset B and the anchor texts of category A, due to the lack of hardware and the amount of storage available.

2.9.2 Web Tasks, Queries, and Topics

A task in Web search engine is referred to a common topic set. Topics are created from the logs (queries) of commercial search engines, with the assistance of tools that extract and analyze the grouping of related queries and identify clusters of queries that highlight different aspects and interpretations of the target query. This section compares tasks, topics and queries.

2.9.2.1 Web Tasks

The task in Web search is a user intention that is restricted to input queries and output results. There are two types of tasks, with differences in their evaluation methodologies: adhoc and diversity.

2.9.2.1.1 Adhoc Task

An adhoc task investigates the performance of systems that search a static set of documents using previously-unseen topics. The goal of this task is to return a ranked list of documents from the collection and increase the probability of relevance documents. The relevancy probability for

[16] The relevance judgment file is provided by the TREC Web Track organizers and it holds all the relevant documents for a set of queries. For the test queries, the relevance judgment file is provided to the participants only after the evaluation campaign.

each document is considered independent of other documents that appear before it in the result list.

More precisely, adhoc signifies a solution designed for a specific problem or task that is non-generalizable, and cannot be adapted for other purposes. An adhoc task allows users to create specific, customized queries, typically via a user-friendly GUI-based system, without requiring in-depth knowledge saved in database schema.

However, with adhoc tasks, documents are judged with respect to the homepage finding and topic distillation tasks (Zhao et al., 2007).

2.9.2.1.2 Diversity Task

The diversity task differentiates the search results by finding the various aspects that the original query implies. While the ad-hoc task focuses specifically on relevance, the diversity task aims to predict the varieties of user intentions, and organizes the results to meet different needs.

Despite the fact that adhoc results are similar to diversity results, in most cases they apply different judging criteria and evaluation measures. The goal of a diversity task is to return a ranked list of pages that together provide complete coverage for a query, while avoiding excessive redundancy in the result list. With this task, probability of the relevance of a document depends on the documents before it in the result list. To evaluate the diversity, documents are judged on both the sub-topics and the topic as a whole. For each sub-topic, the assessor makes a judgment value as to whether or not a document satisfies the information need associated with the sub-topics. Moreover, relevance is judged separately with respect to each sub-topic, so sub-topics of each topic should be mined in order to diversify the search results.

2.9.2.2 Web Queries

A web search query is a string that a user submits to a web search engine to satisfy his or her information needs. Web search queries are distinctive, and are often input as plain text or hypertext, with optional search-directives; such as ‘and’, ‘or’, and ‘-’ (to exclude). As we mentioned on page 4, there are four broad categories include most web search queries (Manning et al., 2008).

Statistics from popular search engines show that the average length of a search query is 2.4 terms, and that about 50% of users entered a single query while approximately 30% entered three or more unique queries.

2.9.2.3 Web Topics

A topic usually involves more underlying information than a simple query (i.e., a query field, a description field and several sub-topic fields). Each topic is structured as a set of sub-topics, each related to different user needs, and the selection of sub-topics is intended to reflect genuine user requirements for the topic. Practically, TREC categorizes the topic as either 'ambiguous' or 'faceted'. Ambiguous topics have multiple distinct interpretations, and a user who is interested in one interpretation would not be interested in the others; one of the interpretations is chosen for the description, and a wider range of interpretations appear in the sub-topics. Meanwhile, faceted topics reflect the underspecified aspects covered by sub-topics. It is worth mentioning that the topic, sub-topics and description field for each query are available for training queries, but not available for testing queries. All the fields (i.e., topic, sub-topics and description) for each query were revealed after the results had been submitted. The example below shows two types of topics, and the underlying sub-topics.

For all topics, the description field and the first sub-topic field are identical, and all sub-topics are either navigational or informational. Navigational sub-topics (type 'nav') assume the user is seeking a specific page or site, and may often have only one relevant page. Informational sub-topics (type 'inf') assume the user is seeking information regardless of its source, provided that the source is reliable. Informational sub-topics often have a large number of relevant pages. Sub-topics chosen are roughly balanced in terms of popularity. Strange and unusual aspects and interpretations have to be avoided as much as possible (Zhao et al., 2007).

<topic number='#' type='***faceted***'>
<query>***adobe indian houses***</query>
<description>
How does one build an adobe house?
</description>
<sub-topic number='**1**' type='inf'>
How does one build an adobe house?
</sub-topic>
<sub-topic number='**2**' type='inf'>
information about Indian tribes that used adobe houses
</sub-topic>
<sub-topic number='**3**' type='nav'>
I'd like to order books or videos/CDs about how to construct adobe buildings.
</sub-topic>
</topic>

<topic number='#' type='***ambiguous***'>
<query>***east ridge high school***</query>
<description>
demographics of East Ridge High School in Lick Creek, Kentucky
</description>
<sub-topic number='1' type='inf'>
demographics of East Ridge High School in Lick Creek, Kentucky
</sub-topic>
<sub-topic number='2' type='nav'>
home page for East Ridge High School in Chattanooga, Tennessee
</sub-topic>
<sub-topic number='3' type='inf'>
information about the sports program at East Ridge High School in Clermont, Florida
</sub-topic>
<sub-topic number='4' type='inf'>
description of the sports facilities at East Ridge High School in Woodbury, MN
</sub-topic>
</topic>

2.9.3 Evaluation Methodology

Evaluation is the key to building better search engines. It is also essential to understand whether or not a search engine is efficient. More specifically, one of the primary distinctions in the evaluation of search engines is the compromise between two factors: effectiveness and efficiency. *Efficiency* is defined in terms of the time and space required by the algorithm that produces the ranked documents. This means that the main issues are how to minimize the index structure, how to build the index in a reasonable time, and how fast the relevant documents can be retrieved. While the time factor is not a significant issue in some competitive environments, due to the availability of quality hardware, efficiency is an important factor in making search engines practical. Most important is the *effectiveness* of retrieval, which is a search engine's ability to find the relevant documents. For a given query, and a specific definition of relevance, the effectiveness factor is defined as a measure of how the ranking algorithm produces the right documents based on their relevance to the users. Relevance is subjective, as users have different points of view; the retrieved documents could be useful to some users and useless to others. Effectiveness and efficiency are affected by many other factors as well, including the interface used to display the results, and query refinement techniques; such as query suggestion and relevance feedback. It is difficult to improve search engines because certain factors must be

controlled (Yang, 2002). Often, neither efficiency nor effectiveness are adequately achieved with the current physical components; even if some techniques could significantly improve effectiveness, they cannot be incorporated into large-scale search engines due to efficiency considerations.

In addition to efficiency and effectiveness, cost is a significant consideration in search engine design. Building an efficient system can require huge investments in processors, memory, storage or disk space, and networks. In general, if we choose any two of these three factors, the third will be determined (Bruce et al., 2010). Achieving particular levels of effectiveness and efficiency, for example, will determine the cost of the system. Alternatively, focusing on efficiency and cost will have an impact on effectiveness. Typically, the best values for effectiveness and efficiency are determined using training data and the cost factor. The training data could be samples of queries with relevance judgments, and the ranking algorithm`s effectiveness is measured on this training data. Search engine optimization (SEO) uses the training data to learn parameter settings for the ranking algorithm that maximizes effectiveness and efficiency.

For this reason, evaluation is usually done in highly defined experimental settings, and we focus on this type of evaluation here.

2.9.3.1 Judgment Criteria

2.9.3.1.1 Adhoc Task

An adhoc task in TREC investigates the performance of systems that search a static set of documents using previously unseen topics. An adhoc task returns a ranking of the documents in the collection in order of decreasing probability of relevance. The probability of relevance of a document is considered independent of documents that appear before it in the result list. For the adhoc task, documents are judged on the basis of their description field, using six-point scales defined as follows (Clarke et al., 2010)[17]:

- ***Nav***: This page represents the home page of an entity directly named by the query; the user could be searching for this specific page or site (grade 4).

[17] The information was taken directly from http://plg.uwaterloo.ca/~trecweb/2010.html

- ***Key*:** This page or site is dedicated to the topic; it is authoritative, comprehensive and worthy of being a top result in a web search engine (grade 3).
- ***HRel:*** The content of this page provides substantial information on the topic (grade 2).
- ***Rel:*** The content of this page provides some information on the topic, and the relevant information must be on the page (grade 1).
- ***Non:*** The content of this page does not provide useful information on the topic, but it could provide useful information on other topics (i.e., other interpretations of the same query) (grade 0).
- ***Junk:*** This page does not appear to be useful for any reasonable purpose; it could be spam or junk (grade 0).

The relevance grade assigned to a level for this purpose is used to calculate retrieval effectiveness measures.

The primary effectiveness measures for the adhoc task are: expected reciprocal rank (ERR) as defined by Chapelle et al. (2010), normalized discount cumulative grade (nDCG) as defined by Jarvelin and Kekalainen (2002), and standard binary measures, including mean average precision (MAP) and precision in the top k documents retrieved (P@k).

2.9.3.1.2 Diversity Task

The diversity task is similar to the adhoc retrieval task, but its judging criteria and evaluation measures differ. The goal of the diversity task is to return a ranked list of pages that, together, provide complete coverage for a query, while avoiding excessive redundancy in the result list. For this task, the probability of relevance of a document is dependent on the documents that appear before it in the result list.

With the diversity task, documents are judged on the basis of sub-topics. For each sub-topic, the assessor makes a binary judgment as to whether or not a document satisfies the information need associated with that sub-topic (Clarke et al., 2010).

The primary effectiveness measures for the adhoc task are: intent-aware expected reciprocal rank (ERR-IA) as defined by Chapelle et al. (2010), intent normalized discount cumulative grade nDCG@k, NRBP, and MAP-IA (Croft et al., 2009).

2.9.3.2 Effectiveness Metrics

Evaluating effectiveness has gained increased attention in the web search field recently, primarily due to the rapidly changing scope of information retrieval systems. How to properly

evaluate web search engines continues to be a challenging and open research issue. While many metrics for Web information retrieval are available in the case of binary relevance, including Precision and Recall as defined by Cranfield's studies to summarize and compare search results, there are only two commonly used metrics for grading relevance: the Discount Cumulative Gain (DCG) and the Expected Reciprocal Rank (ERR). In this section, we discuss these and other metrics used for search engine assessment.

2.9.3.2.1 Precision and Recall

The two most common effectiveness measures are Recall (R) and Precision (P). Recall intuitively measures how well the search engine finds the relevant documents for a particular query, and Precision measures how well it rejects non-relevant documents. In other words, for a given query Recall and Precision expect that there is a set of documents that is relevant and a set that is not relevant; and, thus, that there is a set of documents that is retrieved and a set that is not retrieved. To compute Recall and Precision, the relevancy of the documents must be a binary value; if the relevancy is graded the values are mapped into binary values. As shown in the figure below, 'tp' is the set of relevant documents that are retrieved for a certain query, 'fp' is the set of non-relevant documents that are retrieved, 'fn' is the set of relevant documents that are not retrieved , and 'tn' is the set of non-relevant documents that are not retrieved.

To summarize, Recall is the proportion of relevant documents that are retrieved, and Precision is the proportion of retrieved documents that are relevant.

	actual class (observation)	
predicted class (expectation)	**tp** (true positive) Correct result	**fp** (false positive) Unexpected result
	fn (false negative) Missing result	**tn** (true negative) Correct absence of result

The precision and recall are then defined as:

$$Precision = \frac{tp}{tp+fp}$$

$$Recall = \frac{tp}{tp+fn}$$

Consider we have 10 documents that are retrieved by a search engine for a certain query, and distributed as shown below. The number of relevant documents (R) is 6 (i.e., 5 documents are retrieved and 1 is missed), and the number of retrieved irrelevant documents (N) is also 5. To compute the recall and precision, the location of documents in the list must be considered.

Rank	1. R	2. R	3. N	4. R	5. N	6. R	7. N	8. N	9. R	10. N
Recall	1/6	2/6		3/6		4/6			5/6	
Precision	1/1	2/2		¾		4/6			5/9	

Thus, the precision at 10 (P@10) is equal to 5/10, and the recall at 10 (R@10) is 5/6. Due to precision judging documents as binary values, with Average Precision (AP) the grades perfect, excellent, and good are mapped as relevant, and the grades fair and bad are mapped as non-relevant.

However, many different measures for evaluating the performance of information retrieval systems have been proposed, some of which are based on precision-recall metrics. Here we describe five precision-recall related metrics:

1- F-measure

This is the weighted harmonic mean of precision and recall. The traditional F-measure, or balanced F-score, was derived by van Rijsbergen (1979), and is computed as follows:

$$F = \frac{2.Precision\ Recall}{(Precision+Recall)}$$

This is also known as the F1 measure, because recall and precision are evenly weighted. Two other commonly used F measures are F2, which weights recall twice as much as precision, and F0.5, which weights precision twice as much as recall.

2- Average Precision

Precision and recall are single-value metrics that are based on the entire list of documents returned by the system. For systems that return a ranked sequence of documents, it is preferable to also consider the order in which the returned documents are presented. By computing precision and recall at every position in the ranked sequence of documents, we

can plot a precision-recall curve, that is, plot precision $p(r)$ as a function of recall r. Average precision computes the average value of $p(\mathrm{k})$ over the interval from $r = 1$ to $r = \mathrm{n}$:

$$\boldsymbol{AveP} = \sum_{\boldsymbol{k=1}}^{\boldsymbol{n}} \boldsymbol{P(k)\Delta r(k)}$$

where k is the rank in the sequence of retrieved documents, n is the number of retrieved documents, $P(k)$ is the precision at cut-off k in the list, and $\Delta r(k)$ is the change in recall from items $k - 1$ to k.

This finite sum is equivalent to:

$$AveP = \frac{\sum_{k=1}^{n}(P(k) * rel(k))}{number\ of\ relevant\ documents}$$

where $rel(k)$ is an indicator function equal to '1' if the item at rank k is a relevant document, and zero otherwise.

Some authors choose to interpolate the $p(r)$ function, in order to reduce the impact of 'wiggles' in the curve. For example, the PASCAL Visual Object Classes challenge (a benchmark for computer vision object detection) computes average precision by averaging the precision over a set of evenly spaced recall levels {0, 0.1, 0.2, ... 1.0}.

$$AveP = \frac{1}{11} \sum_{r \in \{0, 0.1, \ldots, 1.0\}} P_{interp}(r)$$

where $P_{interp}(r)$ is an interpolated precision that takes the maximum precision over all recalls greater than r. Average precision is sometimes referred to geometrically as the area under the Precision-Recall curve. Interpolation is considered useful when averaging over several queries, because it can be done at the same recall levels.

3- Precision at R (P@R)

Precision at **R** is the precision at a position R at the ranking list. This measure is highly correlated to Average Precision.

4- Mean Average Precision (MAP)

Mean average precision for a set of queries, is the mean of the average precision scores (uninterpolated) for each query.

$$MAP = \frac{\sum_{q=1}^{Q} AveP(q)}{Q}$$

where Q is the number of queries. Table below shows an example of computing MAP.

Document position in the Ranking list	Relevant documents	Documents found relevant	Precision
1	1	d_1	1/1 = 1
4	2	d_{20}	2/4 = 0.5
5	3	d_{14}	3/5 = 0.6
9	4	d_{11}	4/9 = 0.444
11	5	d_5	5/11 = 0.454
Not retrieved	6	d_3	0
Not retrieved	7	d_8	0
Not retrieved	8	d_9	0
Not retrieved	9	d_{13}	0
Not retrieved	10	d_{17}	0
Average Precision			0.2999

5- Rank Biased Precision (RBP)

The rank biased precision (RBP) metric, a precision-based metric recently proposed by Moffat et al. (2007 and 2008), incorporates a simple user model: the user's persistence in finding relevant documents. Less persistent users are likely to only look at a small number of results, while more persistent users will look deeper in the ranked list. The primary issue with this user model is the unlikelihood of real user browsing behavior being determined based entirely on user persistence, and also on the quality of the results in the ranked list, as illustrated in the example above. RBP is computed as follows:

$$RBP = (1 - p) \sum_{i=1}^{n} g_i \; p^{i-1}$$

where g_i indicates the degree of relevance of the document at rank *i*, and *p* is a parameter that models how persistent a user is when looking through the ranked list (Plachouras et al., 2005).

2.9.3.2.2 Discounted Cumulative Gain (DCG)

DCG (Järvelin and Kekäläinen, 2002) is an evaluation metric that is used specifically with search engines. It has become a popular measure for evaluating web search and related applications. One reason it is widely accepted for web search system evaluation is that it supports graded relevance. The DCG metric is based on two assumptions:

1- Highly relevant documents are more useful than marginally relevant documents.

2- The lower ranked relevant documents (i.e., those further down in the ranked list) are less useful for the user, since they are less likely to be examined.

The DCG metric assumes that users will browse to some position in the ranked list, according to a probability that only depends on the position. This leads to an evaluation that uses graded relevance as a measure of the usefulness, or gain, of examining documents. The gain is accumulated starting at the top of the ranking, and reduced (discounted) at the lower rankings. Thus, the DCG is the total gain accumulated at a particular rank k, and it is computed as:

$$DCG@k = \sum_{i=1}^{k} \frac{2^{g_i}-1}{\log(i+1)}$$

where g_i is the relevance grade of the document at rank *i*.

We can demonstrate this equation with an example. Consider we have 10 ranked documents, where each number is a relevance level on a scale of 4 to 0, corresponding to the previously mentioned description field evaluations (nav, key, high-relevant, relevant, and non-relevant). The ranked grades are: [3, 2, 3, 0, 0, 1, 2, 2, 3, 0].

The discount gain would be: [3, 2/1, 3/1.59, 0, 0, 1/2.59, 2/2.81, 2/3, 3/3.17, 0] =

[3, 2, 1.89, 0, 0, 0.39, 0.71, 0.67, 0.95, 0].

Thus, the DCG at each rank is formed by accumulating these numbers:

[3, 5, 6.89, 6.89, 6.89, 7.28, 7.99, 8.66, 9.61, 9.61].

The perfect value from different lists of rankings would have gain values at each rank: [3, 3, 3, 2, 2, 2, 1, 0, 0, 0], and the ideal DCG is:

[3, 6, 7.89, 8.89, 9.75, 10.52, 10.88, 10.88, 10.88, 10.88]

Normalized discounted cumulative gain (nDCG for adhoc and αnDCG for diversity task), measures the performance of a recommendation system based on the graded relevance of the recommended entities. It varies from 0.0 to 1.0, with 1.0 representing the ideal ranking of the entities. This metric is commonly used in information retrieval and to evaluate the performance of web search engines. To normalize the previous values, the actual DCG values above must be divided by the ideal DCG values:

[1, 0.83, 0.87, 0.76, 0.71, 0.69, 0.73, 0.8, 0.88, 0.88]

$nDCG = \frac{DCG_N}{iDCG_N}$; where $iDCG_N$ is the ideal DCG value for that query.

2.9.3.2.3 Reciprocal Rank (RR)

The reciprocal rank measure has typically been used for web applications with a single relevant document. It is defined as the reciprocal of the rank at which the first relevant document is retrieved. The mean reciprocal rank (MRR) is the average of the reciprocal ranks over a set of queries. For example, if the top five documents retrieved for a certain query were distributed as N, R, N, N, N, where N is a non-relevant document and R is a relevant document, the reciprocal rank would be ½=0.5. If more relevant documents were retrieved, as in the ranking X, R, X, R, X, the reciprocal rank would still be 0.5. The reciprocal rank is very sensitive to the position. It falls from 1 to 0.5 for rank 1 to 2, and from 0.33 to 0.2 for relevant documents at position 3 to 5. For example, X, X, X, X, R would have a reciprocal rank of 1/5=0.2. The MRR for these two ranking lists would be (0.5+0.2)/2=0.35.

2.9.3.2.4 Expected Reciprocal Rank (ERR)

Recent research regarding modeling user click behavior has revealed that the position-based browsing assumption that underlies DCG is invalid. Studies have shown that the likelihood of a user examining the document at rank *i*, is dependent on how satisfied the user was with previously observed documents in the ranked list. This type of user model is known as the cascade model (Plachouras et al., 2005).

In reality, the probability of a user browsing to some position in a ranked list depends on many factors other than position. A serious issue with DCG is the assumption that the usefulness of a document at rank *i* is independent of the usefulness of documents at lower ranks than *i*.

To demonstrate this more specifically, consider we have five judgment scales: perfect, excellent, good, fair and bad. And suppose we are evaluating two ranked lists, the first with 20 good documents, and the second with one perfect document and 19 bad documents. Which ranked list is better? Most DCG settings would indicate that the first list is best.

However, the single perfect document in the second ranked list could completely satisfy the user's information need, so they would observe the first document, be satisfied, and stop browsing without seeing the bad document. In addition, the first ranked list could require the user to spend much more time and effort to satisfy their information need, as each good document might only partially satisfy the need. Thus, because it could satisfy their information need with the least amount of effort, users might actually prefer the second ranked list over the first.

More precisely, DCG does not consider the likelihood of a user examining the document at rank *i*, which means this document is independent of the documents previously observed in the ranked list.

Also, ERR pays attention to the maximum likelihood on the click logs, computed as follows:

$ERR = \sum_{r=1}^{n} \frac{1}{r} P$ (user stops at position r).

where *n* is the number of documents in the ranking. The probability of the user stopping at position *r* is given by the definition of the cascade model:

$$ERR = \sum_{r=1}^{n} \frac{1}{r} \prod_{i=1}^{r-1}(1 - R_i)R_r$$

where Ri is the probability that the document satisfies the user. The Ri values can be estimated by the maximum likelihood on the click logs. Compared to position-based metrics such as DCG and RBP, for which the discount depends only the position, the discount in ERR depends on the relevance of the documents above it (Plachouras et al., 2005).

2.9.3.2.5 Intent Aware Precision

For a given topic, the intent-aware precision (P-IA) measure treats each subtopic as a distinct interpretation of the associated query (Clarke et al., 2009). Standard evaluation measures are computed separately with respect to each interpretation. A weighted average is then computed across the various interpretations to give intent-aware versions of the standard measures. A probability is attached to each interpretation. More specifically, intent-aware computes precision at retrieval depth k using the following procedure. Assume there are M topics. Let N_t, $1 \leq t \leq M$ be

the number of subtopics associated with topic number *t*. Let j_t(i, j) = 1 if the document returned for topic *t* at depth *j* is judged relevant to subtopic *i* of topic *t*; otherwise, let j_t(i, j) = 0. We then define intent-aware precision at retrieval depth *k* as:

$$\text{Precision\_IA} = \frac{1}{M}\sum_{t=1}^{M}\frac{1}{N_t}\sum_{i=1}^{N_t}\frac{1}{k}\sum_{j=1}^{k} j_t(i,j)$$

2.9.4 Related Works from TREC Web Track

The challenge models are deployed as groups from various companies and organizations, and have been used at TREC for several years. A total of 12 groups joined in 2012, less than the 16 groups in 2011 and a considerable decrease from the 20 groups in 2009 and 2010. The substantial change in the number of models each year indicates the difficulty of the Web track. Web indexing and searching is a fairly challenging task, and not considered a personal effort. As demonstrated at TREC, many models are programmed by groups of several people belonging to different organizations. The approaches used high performance hardware to improve the relevancy of the results. The most relevant approaches used clusters or grids of several computers in distributed environments. For example: Microsoft's team (Craswell et al. 2009) used the DryadLINQ[18] data-parallel processing platform (Yu et al. 2008) and the Scalable Hyperlink Store (Vogt and Cottrell, 1999) on a cluster of 220 machines; the University of Maryland and University of Southern California model (Elsayed et al., 2010) used a web-scale retrieval engine (Ivory[19]) built around a cluster-based computing environment such as 'Hadoop' or 'MapReduce' (Hiemstra and Hauff, 2010); the Chinese Academy of Sciences model (Li et al., 2010) used Golaxy[20], a high performance distributed search platform deployed over several servers; the Carnegie Mellon University model (Nguyen and Callan, 2010) used the Lemur Project's search engine[21]; the University of Amsterdam model (Kamps et al., 2010) used the Indri[22] search system for indexing; the University of Twente model (Hiemstra and Hauff, 2010) used Hadoop-MapReduce framework[23], which is also used by Google for experimental evaluations; the University of Delaware model, in cooperation with Yahoo, (Zheng et al., 2010)

[18] DryadLINQ: A System for General-Purpose Distributed Data-Parallel Computing Using a High-Level Language
[19] http://lintool.github.com/Ivory/
[20] Golaxy Search Engine - http://www.golaxy.cn
[21] http://www.lemurproject.org/lemur/
[22] http://www.lemurproject.org/indri/
[23] http://hadoop.apache.org/

used the scalable greedy framework run over several servers; the University of Glasgow model (McCreadie et al., 2010) used Terrier framework[24] for adhoc tasks and 'xQuAD' framework for diversity tasks; and, the University of Waterloo model (Smucker, 2010) used Indri framework in the SHARCNET, and Compute/Calcul Canada[25] in their experiments. Evaluation section in each chapter compares the results of these models with our model.

2.10 Parameter Setting

Nearly every ranking algorithm has parameters that can be tuned to improve the accuracy of the results. BM25, for example, uses three parameters for term weighting and query likelihood. Ranking algorithms for web search can have hundreds of parameters that provide the weights for the associated features (Vogt and Cottrell, 1999). The values of these parameters can have a major impact on retrieval effectiveness, and the values that provide the best effectiveness for one application may not be appropriate for another, or even for a different document collection. Choosing the correct parameter value is not only important to the performance of a search engine when it is deployed, it is also important for comparing the effectiveness of two retrieval algorithms. An algorithm that has had its parameters tuned for optimal performance for the test collection may appear to be much more effective than it actually is when compared to a baseline algorithm with poor parameter values. The appropriate method of setting parameters to maximize effectiveness and allow fair comparisons of algorithms is to use both training and testing sets of data. The training set is used to learn the best parameter values, and the testing set to validate the parameter values and compare ranking algorithms. The sets have separate queries and relevance judgments. In TREC experiments, for example, the training set is usually based on queries and relevance judgments from previous years.

Given a training set of data, there are a number of techniques to find the best parameter settings for a particular effectiveness measure, the most common being to explore the space of possible parameter values. This requires a large number of retrieval runs with small variations in parameter values. Although this can be computationally infeasible for a large number of parameters, it will find the parameter settings that give the best effectiveness for any given effectiveness measure (Vogt and Cottrell, 1999). We used only one parameter value in our

[24] http://terrier.org
[25] http://www.sharcnet.ca

experimental setting, which helped our system provide the best results for training and testing topics.

2.11 Stop Words and Stemming

Both stop words and stemming are common techniques to improve document representation in Web information retrieval. Removing stop words improves the retrieval speed, reduces the size of the index, and is proven to increase retrieval effectiveness. Stop words refer to words used in documents that convey little or no meaning and are used frequently, or words that fail to discriminate between documents. According to the Brown corpus[26], the most commonly used words in written English are: 'the', 'of', 'and', 'to', 'a', 'in', 'that', 'is', 'was', and 'he'. These ten words make up 25% of written text; the top 100 words make up 40% (Liddy, 2007). These words are so common that they do not help discriminate relevant documents from non-relevant documents, and they increase the storage requirements of an information retrieval system by adding index terms for each document. Such words are generally listed in a stop list, indicating that they should be left out and not indexed when they appear in the source text. Prior works on developing and testing stop lists indicate that they do not degrade the effectiveness of information retrieval, and in some cases they improve it (Ballesteros, 2011). A document-to-document similarity calculation identifies words that are not differentiators for query results. This approach removed 75% of the terms in the collection, but it is very computationally intensive, and impractical for large collections. The most commonly-used stop list is the manually-generated list used in the SMART system[27]. Researchers Lo et al. (2005) proposed different techniques to automatically generate a list of stop words, by determining how informative a term is using the Kullback-Leibler divergence measure. In our experiments, we used stop words in one method, and removed them in another method.

Stemming is a technique that conflates word variants into one entry in the information retrieval index. For example, if endings such as -s, -es, -ing, -ly are removed from terms, the terms: compute, computes, computer, and computing are indexed as the same term. This is very helpful to widen a search beyond the specific term variants found in the query. Sometimes there

[26] W. Nelson Francis and H. Kucera, Frequency Analysis of English Usage: Lexicon and Grammar (Houghton Mifflin, 1982): p. 465.
[27] ftp://ftp.cs.cornell.edu/pub/smart/

is confusion between ambiguous terms and unrelated terms; for example, if ‘train’ referred to a connected series of vehicles and ‘training’ to acquisition of knowledge, they are joined erroneously by stemming. There are a number of techniques that deal with word sense disambiguation in information retrieval (Cutting et al., 1992), as well as several stemmer algorithms. The first published stemmer was written by Lovins (1968), and the most popular stemmer is the Porter Stemmer[28].

[28] http://snowball.tartarus.org/algorithms/porter/stemmer.html

Chapter 3

Indexing Web Pages

Using Document-Tags

3.1 Introduction

An inverted index table based on document-tags is different than the standard approach of building an inverted index, which is based on collecting data from the text content of a webpage (the visible information). Web documents often have a structure that combines different topics in one document, with each section presenting a specific topic. Traditional indexing methods use terms from different sections to compute term occurrences, but these approaches do not take into account the fact that terms corresponding to different sections belong to different topics. For example, if a section contains the words 'Diana Ross' and another section contains 'Prince Charles', this page is irrelevant to the query 'Princess Diana', even though earlier versions of traditional search engines would rank it relevant. Some web documents might contain very little text, or perhaps only images or animations, and some might only contain links to redirect the pages to other pages; in such cases, it is difficult to build the inverted index with existing approaches.

Our idea is to generate the inverted index based on data collected from different tags and hidden attributes in web pages; that is, we partition the pages into different sections, depending on the available tags. Web designers hide information for different reasons; to describe some fields as private, for example, or to show particular information when certain components on the web page are active. Other information is used to define the attributes, or focus the document on a specific topic. In our approach, we use all these special tags/attributes and document URLs to compose the vector space for each document, and to build the inverted index to achieve improved search results.

In this section, we describe our first indexing framework, and detail how it indexes data from the meta-content and other html tags of web pages. Meta-data is often used by an author to describe the main topic or purpose of a web page. The index terms are usually collected from the visible data, but meta-data is preferable when:

- the web pages contain only multimedia data, such as images or animations;
- the web pages contain only URL links and minimal text data (e.g., home pages);
- the web pages contain several different topics;

- the documents are repetitive;
- we want to reduce the size of the inverted index; and,
- the weighting of a web page depends on specific zones in the document.

Our system was indexed with the data available in the following tags: 'URL', 'Title', 'Meta-Keywords and Meta-Description', 'Heading', and 'Image'.

We also presented a new approach that adds leverage to standard retrieval techniques, by using only meta-data to produce a fast web search engine and a small index table. The only thing that motivated our system to use the selected tags was that meta-data and other attributes were typically available in the main pages of websites. Since our index was based on the subset B collection, which involved the top rooted pages of websites, we assumed that all data available in our dataset are relevant for the selected tags.

In our experiment, we used the 50 million documents from the ClueWeb09 corpus, and 50 training queries from the TREC 2009 web track, as well as the 50 testing queries from the TREC 2010 web track (see Appendix II for a list of these queries). In the TREC Web Track ad-hoc task 2010, our framework achieved reasonable results, and was published in the TREC 2010 proceedings (Al-akashi and Inkpen, 2010).

3.2 Documents Indexing

The ClueWeb09 collection[29] contains two kinds of documents: Wikipedia and other webpages. Therefore, we used two types of indexes and a different indexing strategy for each. In this section, we discuss these types in detail.

3.2.1 Wikipedia Collection Indexing

Wikipedia English is an important part of ClueWeb09 (Category A & B). It represents a spam free collection of web-pages with dedicated descriptions of approximately 3,500,000 concepts or articles, distributed over five million documents. While it is not possible to find every article a user might ask for in Wikipedia, there is a very high probability that the most potentially popular queries can be answered by ranking articles in this repository. As the Wikipedia portion of ClueWeb09 is a collection of html pages, it only needs basic filtering to serve as a web repository. And since each article in Wikipedia deals with one topic, we indexed only the title

[29] http://lemurproject.org/clueweb09.php

section rather than the entire content of the documents, and ignored non-article pages such as lists, files, disambiguation pages and category pages. The index is structured as a vector space file rendered in a hash table where each key is the title of an article, while the content of each key was given by TREC id and URL for the corresponding article. Table 3.1 shows an example of our Wikipedia collection index.

Title	**TREC id**	**URLs**
smith_haven_mall	02-00-00003	Smith_Haven_Mall
oriol_salvia	02-06-18190	Oriol_Salvia
francis_adams translator	02-14-19165	Francis_Adams_(translator)
university of waterloo	02-23-14321	University of Waterloo
francis_burns_footballer	02-14-19188	Francis_Burns_(footballer)
smith_island_washington	02-00-00005	Smith_Island_(Washington)
bill_clinton	02-16-01521	Bill_clinton

Table 3.1 Wikipedia Collection Index

3.2.2 Web Collection Indexing

Another possible approach to find relevant results for a particular query is to index the main portion of the ClueWeb09 web collection (excluding the Wikipedia part). Generally, web pages are comprised of different tags/attributes, and each attribute is used for a specific purpose; to archive the data, for example, or to help a search engine distinguish which category a page belongs to. Other attributes provide simple descriptions for web pages. Our approach assigned each attribute to a specific weight, with the highest weight for the URL and the lowest for the image attribute. We used five attributes: URL, Title, heading, keywords-description and image. As an example, assume we need to compose the vector space representation of each attribute for the web page represented by the following URL:

‘*ca.news.yahoo.com/nphotos/slideshow/ss/events/us/080318_obama’*

The first step is to segment the document into the following attributes:

1- URL: The URL link is the address of the web page. It is often composed of tokens that refer to important words that are used frequently in the document, and which tend to reflect the topic of the web page.

2- Title: Most web documents contain valuable information in the title. The title of the above example is **'**U.S. President Barack Obama - Yahoo! Canada News Photos'**.**

3- Heading: Web documents often contain titles or sub-titles that are presented in large font and bolded by applying the 'heading' attribute. Most documents that focus on specific subjects must contain a title as a heading. For our example, the heading here is 'U.S. President Barack Obama'.

4- Meta-Keywords and Meta-Description: Sometimes documents do not contain the title attribute, or the text of the title does not cover the main topic of the document. Instead, the meta-keyword attribute contains all the keywords that are related to the main topic of the document. As well, the meta-description attribute can contain a full description of the document. Thus, these two attributes support topic finding in the document. In our example, the document contains the following description attributes and keywords:

***Description* = '**U.S. President Barack Obama on Yahoo! Canada News Photos'.

***Keywords* = '**photos, Yahoo! Canada News'.

5- IMG attribute[30]: If the document contains images, flash animations and/or movies, the 'alt' attribute provides information to identify these items. Our system stripped off 'alt' attributes with a string of less than three terms, because these attributes are often not related to the main topic (i.e., attributes such as 'email me' and 'click here' are not relevant). In our example, the web page contains the following 'alt' attributes to describe the available images:

alt 1	US President Barack Obama speaks before signing the Small Business ...
alt 2	LOS ANGELES, Calif. -- In a surprising move, Cindy McCain, wife ...
alt 3	A policeman keeps watch at Heathrow airport. Bahrain on Tuesday ...
alt 4	British Member of Parliament George Galloway gives an interview ...
alt 5	US President Barack Obama speaks before signing the Small Business ...
alt 6	US President Barack Obama leaves the Oval Office at the White ...
alt 7	US President Barack Obama prepares to leave the Oval Office ...
alt 8	FILE - In this June 26, 2010 file, White House Chief of Staff ...

[30] Some image tags replaced 'alt' by 'title' attributes.

Overall, if the page contains many images and little text, traditional search engines do not have enough text and find it difficult to index such pages. To weight each topic in a document separately, our approach aggregated the previous tags in a vector of blocks:

$$[\text{URL} \mid \text{Title} \mid \text{Heading} \mid \text{Meta-Keywords} \mid \text{Meta-Description} \mid \text{Alt}_1, .., \text{Alt}_8]$$

To do this, we first performed html filtration to remove the html tags, unicodes and numerical symbols, stop words, and all terms with the lengths of less than three characters.

Finally, we integrated the following two indexes. The first index is generated at the indexing time and deals with each document as a vector of content; whereas the second index is a kind of forward index that is generated in search time and reranks the documents retrieved from the first index more precisely, based on its zones, in order to take into account proximity search issues.

1- **Global Content Index (Inverted Index):** Each document in our corpus was indexed by using the Apache Lucene open source library.
2- **Local Content Index (Forward Index):** Our approach builds another index that processes each vector separately based on its local content for each zone. This means that the second index boosts the first index by posting the relevant vector to the memory to allow search in the vector tags. This increases the accuracy of document retrieval and it also allows the feedback about the relevance of the documents in the initial set of results from the inverted index to be enhanced based on this forward index. The retrieved results tend to have high accuracy, because the system takes into account which zones/attributes of the documents contain the query words. Figure 3.1 shows the structure of our inverted and forward index.

As explained in this section, we present the pseudo code for building the inverted index for indexing the documents using Lucene, as follows[31].

Initialize: Indexing

Wikipedia Pages:

Loop for all documents in Wikipedia collection

Index the main title using Lucene

End of loop for all documents in Wikipedia

[31] The forward index is built at the search time during the ranking of the documents, as shown on page 66.

Other Pages:

Loop for each document (D) in collection

Combine URL+Title+Heading+Keywords+Description+Alt(s) in vector 'V'

Index vector V using Lucene

End of loop for each document (D) in collection

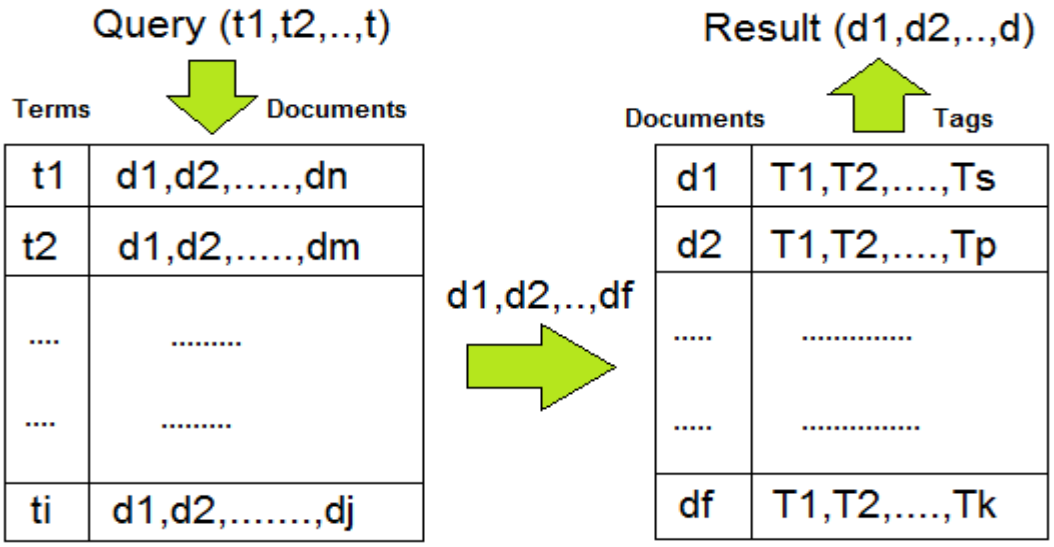


Figure 3.1 Inverted and Forward Index Tables

3.3 Document Weighting

For each query, we used an Apache query processing framework to retrieve the initial ranked list of documents from the inverted index; then the documents were loaded into the memory to make another iteration of scoring. Relevance feedback does not necessarily happen due to user feedback; the system could generate feedback to enhance the first results retrieved from the inverted index, by using the forward index. In this case, weighting can be based on whether or not query terms are available in a single attribute in a document.

Before applying document weighting, we should determine if all attributes in a document are equally important. Experimentally, we proposed giving each attribute a different weighting score. We used the parametric *weights:* $P(a_1) = 3$, $P(a_2) = 4$, $P(a_3) = 4$, $P(a_4) = 5$ *and* $P(a_5) = 5$ *to* the URL, title, heading, keywords, and description attributes, respectively. This corresponds to a query in which matching the headings or titles is more relevant than keywords

and descriptions, because keywords and descriptions expand the topic of the document, while the URL's contribution is most relevant.

Given a query *q* and a document *d*, our system assigns the pair (*q*,*d*) a score in the range [0-1] by computing a linear combination of the *attribute scores*, with each attribute in the document contributing a Boolean value.

To do so, our system partitioned the query string into terms and assigned parametric values to their locations ($P(q_i)$ in the next formulas). For example, if the query length is one term, the parametric location weight is 1; if it is two terms, the parametric location weight is 0.5 for each term, since we consider the two terms to be equally important. Likewise, if the query length is three terms, the parametric weights are 0.5, 0.3, and 0.2. Finally, if it is four terms, the parametric location weights are 0.4, 0.3, 0.2 and 0.1 (if the query has more than four terms, the other terms are ignored[32]). The similarity coefficient score of a document '*d'* for a query '*q'* can be computed as:

$$SIM(d,q) = W(att) + W(alts) \quad (1)$$

where *SIM(d,q)* is a similarity coefficient in the range (0-1), $W(att)$ is the total weight of all attributes except 'alt', and *W(alt)* is the total weight of '*alt*' attributes, computed as follows:

$$W(att) = \sum_{i=1}^{n} \frac{\sum_{j=1}^{m} P(q_j)}{P(a_i)} \quad (2)$$

where $0 \leq W(att) \leq 0.78$

n = number of attributes in the document, except '*alt*';
m = number of terms in the query;
q_j = parametric weight for the query term *j* (if query term *j* appears in the zone *i*); and,
a_i = parametric weight for the attribute *i* (if the query term *j* appears in the attribute *i*).

For example, if all query terms are located in all the attributes, the weighted attribute $W(att)$ is:

[32] These values are the approximating the importance of the terms in a query, based on the flowing schema: the first term is better than the second term, the second term is better that the third term, and so on.

$W(att) = 1/3 + 1/4 + 1/5 = 0.78$

$$W(alts) = C * \log\left(\sum_{j=1}^{m} 5 * P(q_j)\right) \quad (3)$$

where $0 \leq W(alts) \leq 0.22$

m = number of alt attributes in the document;
$P(q_j)$ = parametric weight for the query *j* (if the query appears in that attribute); and,
C = normalized factor equal to 0.22.

Query terms are often located in specific attributes of a matching document. For more contrast, let us consider we have five documents with six attributes (Heading and Title are grouped together, as Keywords and Description), and a query such as '*President Barack Obama'*. Assume that the documents have the following distribution of query term occurrences in the attributes, shown in Table 3.2. A value of '1' means that the query '*President Barack Obama*' completely matches either attribute H1 or the Title for document D2, as well as the URL attribute for d3, and the Keywords or Description attribute for d4. A value of '2' means that the query '*President Barack Obama*' completely matches two attributes of 'Alt'. Similarly, the value '0.7' means that the attribute H1 or the Title of document D5 only matches the string '*President Obama*', which is equal to the parametric distribution 0.5+0.2.

DOC ID	URL	MAX[H1 or Title]	MAX[Keywords or Description]	Alt
D1	0	0	0	2
D2	0	1	0	0
D3	1	0	0	0
D4	0	0	1	0
D5	0	0.7	0	0

Table 3.2 Example of five documents with their weighted distributions in different attributes

To compute the weight for each document *d*, the similarity between the query vector (*q)* and the document vector (*d)* is the inner product $\bar{V}(q).\bar{v}(d)$, which is the sum of the weights of the query terms matching all attributes. Hence, the mapping between web documents and query is as shown in Table 3.3.

DOC ID	D1	D2	D3	D4	D5
SIM(Q,D)	0.22*log(5*2)=0.22	1/4=0.25	1/3=0.33	1/5=0.20	0.7/4=0.175

Table 3.3 Similarity coefficients between documents and the query

Thus, the final ordered list of documents for this query is (D3, D2, D1, D4 and D5).

The idea behind using the 'OR' operator, is that most terms from 'H1' are often shared with the 'Title' terms, and vice versa. Similarly, most terms of the 'keywords' attribute are shared with the 'description' attribute, and vice versa. From this, our system selects the maximum term matches between the query and either attribute.

Briefly, based on the above discussion, we present the algorithm that calculates the ranking of each document for a query, as follows (in pseudo-code):

```
Initialize: assign document 'D' the weight from Lucene.
    Loop for each Tag 'T' in 'D'
        Loop for each term 't' in query 'Q'
            If term 't' in Tag 'T' then
                    SUM=accumulate the parametric term 't' weight for Tag 'T'
        End of loop on all terms in (Q)
            If (Tag!='alt') Divide the SUM by the parametric field weight of Tag 'T'=+R
    End of loop on all Tags in 'D'
Assign ranking of document 'D' as SUM of weights R
```

3.4 Evaluation

In this section, we present the experimental results for the approach presented above, for the adhoc and diversity tasks on the TREC Web Track benchmarks. In 2010, TREC track organizers and experts assessed our model (denoted DFalah2010 in the next tables). The results are shown in Tables 3.4 and 3.5. We present the results for the queries that obtained the best values, and the mean score of the 36 test queries, using MAP (Mean Average Precision), P@k (precision at top k result), and other measures required by the organizers.

We also compare our results to the average mean of the TREC 2010 Web Track results of all 88 models submitted to TREC for the test queries (36). The results of these comparisons are shown in Table 3.6. Figure 3.2 shows our result for each query (topic). Figure 3.3 shows our results for the adhoc and diversity tasks.

DFalah2010	**MAP**	**P@5**	**P@10**	**P@20**	**MAP-IA**	**αDCG@5**	**αDCG@10**	**αDCG@20**
Best topic	0.046	0.600	0.800	0.500	0.044	0.593209	0.585290	0.668684
Mean over all topics	0.011	0.155	0.133	0.119	0.008	0.158280	0.178036	0.213085

Table 3.4 The best and mean results for the 36 testing queries - TREC 2010

DFalah2010	**αnDCG@5**	**αnDCG@10**	**αnDCG@20**	**P-IA@5**	**P-IA@10**	**P-IA@20**
Best topic	0.751802	0.684204	0.672716	0.333333	0.225000	0.150000
Mean over all topics	0.190764	0.205070	0.243339	0.071944	0.051396	0.044931

Table 3.5 (continued) The best and mean results for the 36 testing queries - TREC 2010

Model-ID	**αnDCG@10**	**P-IA@10**
DFalah	0.205	0.051
Average over 88 models	0.213	0.083

Table 3.6 Average mean results (88 models/36 test queries) and our submission

Summary Statistics	
Run ID:	DFalah2010
Task :	adhoc
Category:	B
External resources used:	(A) no additional resources
Number of Topics:	36

Adhoc measures		Diversity measures	
Retrieved	20515	α-nDCG@10	0.2051
Relevant	4289	α-nDCG@20	0.2433
Relevant retrieved	264	ERR-IA@10	0.1668
Prec@10	0.1333	P-IA@10	0.0513
Prec@20	0.1194	P-IA@20	0.0449
MAP	0.0110	MAP-IA	0.0082
NDCG@20	0.0674	NRBP	0.1589
ERR@20	0.0643	ERR-IA@20	0.1778

Figure 3.2 Web Track 2010 adhoc and diversity results

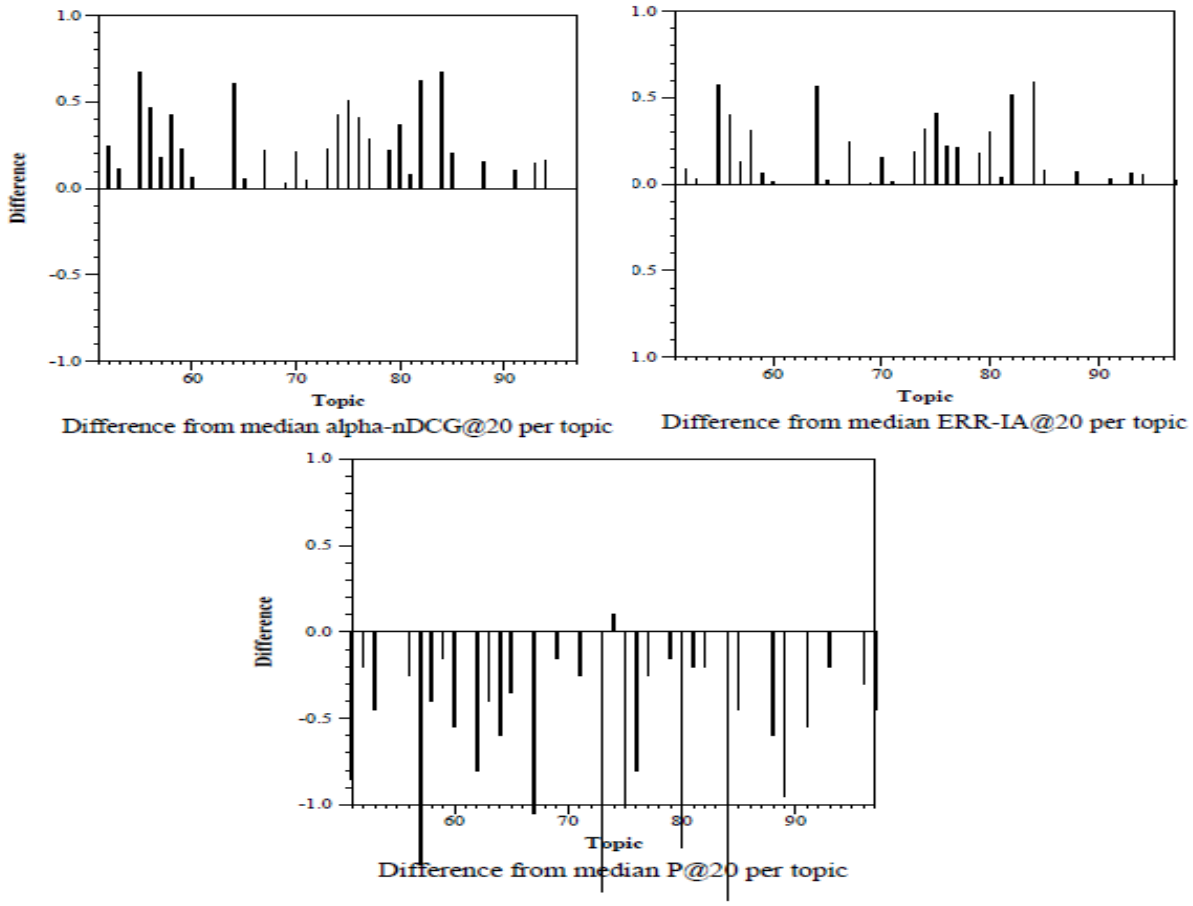


Figure 3.3 Web Track 2010 adhoc result for each query (topic)

3.5 Discussion

This chapter describes our first approach and experiments with the ad-hoc and diversity tasks of web indexing. We indexed most of the meta-data that was available as non-visible attributes (excluding the heading). Our approach assigned different weights to each attribute, unlike the traditional models that typically use only the visible web data. Even though using a linear combination of multiple field weights is fairly standard in Web search (Robertson et al., 2004), the combination of query field weights in document field weights is potentially new in our Web search engine. As we mentioned, the selected attributes are preferable when the speed is important, and when the size of the inverted index is large.

Our first approach achieved satisfactory and reasonable results. Despite the fact that this approach had two advantages (small index size and fast retrieving results), experiments (the

TREC results) showed that improvements are needed. We believe that the results of our first approach ranked below the mean average of other approaches, for the following reasons:

1- Spam filtering was not used.
2- The duplicated documents were removed from the final ranking list.
3- The documents in Wikipedia collection were indexed by their titles only.
4- Documents content were not indexed.
5- Overall, ~66% of the documents' contents were not indexed.

Our experiment with the testing queries showed that the percentage of pages that contain the selected attributes was approximately 34%. However, many different attributes were used, most of which encode details that did not identify the page content. We used the presence of one or more of the keywords or description tags as evidence that effort went into describing the content of the sites. Though the selected attributes had more impact in most documents, we noticed that isolating attributes from their document content resulted in most documents were not being indexed. Likewise, we found that 34% of home pages contain such attributes in their content, and we also observed great diversity in the Web meta-tags that were found, due to the lack of usage standardization. We also found that spam documents had high impact on the subset B collection.

Overall, we determined that the current approach was not suitable for queries that match in the document content. Some of selected tags; e.g. 'Keywords, and 'Description', are merely available on home pages, and other pages in the same domain were not indexed; whereas other selected tags, such as: 'URL', 'Titles', and 'Heading', are still important because they are available in each Web page in the collection. Therefore, in the next chapter, we explore retaining the current solution, but migrating to another framework that can refine and address the drawbacks.

Chapter 4

Indexing Web Pages

Using Collective Knowledge

4.1 Introduction

Indexing is crucial to finding relevant information on the Web. Various indexing methods are used in a wide range of applications, including finding Home pages, locating entities and Web page classification. New, highly-scalable indexing algorithms are needed, due to the estimated one billion pages currently accessible on the web. Previous work classified web document indexing into two types, word-based and phrase-based (Huang et al., 2007).

In word-based indexing single words are used to build an index table, and to find a set of relevant pages according to certain computations. To determine key-phrases that are important for each topic, a set of articles can be assembled from a particular dictionary. This procedure has been used recently for document clustering, entity finding and document classification in small collections, but it has not been applied to large scale webpage indexing.

Moreover, the current approach of word-based indexing does not scale well for phrasal queries, because the word positions must be recorded which requires significant storage space. However, indexing without considering word location for proximity searching can cause two documents to seem similar if there are words in common, even though they have different topics.

Term frequency is important for determining document topics, but not for all document configurations because documents could contain different topics in different parts. Only a few systems address this (e.g., Singhal et al., (2000) used a sliding window for each topic). Our approach in the previous chapter using meta-data did not capture the topic of most documents, because it did not consider the document content. In this chapter we further develop our previous model by grabbing non-relevant attributes, and using only the content attributes URLs, Titles, and Headings to capture the phrases and/or words that relate to a particular topic from the document's content; such that, the relevant keywords in a document should be shared between the document content and the nominated attributes. In order to test the search model, we evaluated our approach using the queries and topics made available by NIST in 2011 and 2012 as part of TREC web search track. This proposed approach has been published in (Al-akashi and Inkpen, 2011) as part of the TREC 2011 proceedings and in (Al-akashi and Inkpen, 2012).

4.2 Building the Index

In this section, we describe the idea behind our indexing approach for handling Web content by using phrases and concepts that are knowledgeably structured in the Wikipedia repository. The proposed approach is robust, and it allows the retrieval of documents relevant to a specific topic. And the proposed inverted index is structured as a centralized hash search tree, which overcomes the problem of using hash tables to index a very large collection of web documents. Currently, hash tables can be used for indexing these large collections only in a distributed environment, such as MapReduce (Hiemstra and Hauff, 2010). However, a centralized index can cause slow query response time unless it is used on a new platform, while the proposed index, though centralized, still provides fast indexing and retrieval of relevant pages.

The idea for an index that stores data efficiently was inspired by the block-oriented storage contexts of file systems; such as B-tree (Becker et al., 1996), which avoid the rebalancing issues of some indexing trees (e.g., binary trees). The encoding algorithm that generated the vocabulary term for drawing the path from the root to each leaf was used a Hash Algorithm. Each leaf in the index holds a table of all indexed documents related to a particular topic, and the label (path) should match a search query for that table. We used three types of tables for each concept or vocabulary term, and each table holds similar types of documents structured as a set of vectors with a certain number of dimensions. The three categories include a repository for all Wikipedia documents, a repository for all home pages and a repository for other web documents.

4.2.1 Assembling the Titles for the Index Nodes

Wikipedia turns out to offer a good structure for describing the topics; for example, abbreviations and bolded terms are always important in the content of a document. Not all pages in Wikipedia are necessary for indexing, e.g., ‘Disambiguation’ and ‘File’ pages were removed from consideration in our model, and non-textual pages and non-article pages were also stripped off. Overall, the following terms from the remaining Wikipedia articles were extracted:

- all abbreviation terms;
- all bold terms or phrases available in the head of the content;
- all phrases in the headings;
- all titles of pages (titles can be different than headings);
- all terms or phrases located between brackets, or split by punctuation characters; and,

- all phrases and terms split by conjunctions, such as ‘or’ and ‘and’.

Stop words were removed from the content in a pre-processing step. We then obtained three types of terms: single terms or abbreviations (e.g., ‘AVP’ or ‘Ottawa’); two-word phrases (e.g., ‘Martha Stewart’); and longer key-phrases (e.g., ‘President United States’). These terms were initially used for building the index nodes and labeling them by titles accordingly.

4.2.2 Vector Generation

In this section, we describe our representation for each table in the index that is based on the vector space model. Similar to the model in the previous chapter, each table is treated as n-dimensional vector, where each dimension represents a specific type of data. In this model, we added more functions in each vector with dimension ‘i’ assigned to function ‘i’. As discussed, we used three types of tables, each organized with a similar number of dimensions:

$T(W) = \{D_1, D_2, .., D_m\}$; where D_i i=1..*m*, represents one of the *m* Wikipedia documents indexed in table *W*;

$T(H) = \{D_1, D_2, .., D_n\}$; where D_i, i=1..*n*, represents one of the *n* homepage documents indexed in table *H*; and,

$T(O) = \{D_1, D_2, .., D_k\}$; where D_i, i = 1..*k*, represents *k* other documents indexed in Table *O* (i.e., all but the Wikipedia and homepage documents).

Each leaf in the index is a class that holds these three tables, and its name matches the name of a Wikipedia article. Some operations were performed on the names of the articles before using them to assign titles to the index nodes or leaves; for example, all stop words, symbols and numbers were stripped off.

4.3 Indexing the Collection

As mentioned, we categorized the documents in our repository index into three classes[33] (i.e., Wikipedia articles, home pages and other documents), and each class holds a particular set of documents. In this section, we describe the indexing algorithm we used for each category or class.

[33] A class is a set of index tables generated for one type of data. The class is distributed in all leaves.

4.3.1 Wikipedia Index Class

Wikipedia is a collective knowledge source, with approximately five million English language articles. This collection is used by many researchers. For example, Itakura et al. (2011) proposed a method for computing the anchor density and identifying the topical links within existing articles based on a specific anchor density threshold. Generally, the data in each article is structured into several fields, and it can have a relationship with other articles via tags or links to expand a certain topic. Each article has a unique vocabulary name identifier (ID), and Wikipedia sometimes uses different faceted vocabulary terms to describe the same article. We extracted the important key phrases from each article, and used them as setup information to rank each leaf in the index. Each document in the Wikipedia repository version in the ClueWeb09 collection was scanned and composited as the following normalized vector.

- Word occurrences (frequencies): To determine which term is significant, we used a specific threshold value (15) to select the best terms, and disregarded those that were lower than the threshold[34]. Likewise, to determine which document was significant, we used a specific threshold impact value (2.5) and excluded documents which were lower than the threshold. The impact value was computed by summing the cosine similarities of the significant terms. The cosine similarity was computed between a vector that represents the frequency of each significant term, and a vector that represents the frequencies of content terms (see Module 4 below).
- Outgoing-link occurrences (frequencies): Outgoing-links are all links that point internally[35] to other Wikipedia articles. If an article frequently points to other articles at more than one position in the content, we assumed that all the articles are related or similar in topic.
- All external links.

Html source codes, ads, navigation links and stop words were removed, and thereby the outputs were transformed and represented as vectors in the following structure:

$$D^W = \{D^{URL}, D^{ID}, D^{lf}, D^{eURL}, D^{tf}\}$$

where D^W is a Wikipedia vector for document D, D^{URL} is a document URL, D^{ID} is a TREC identifier (TREC-ID), D^{lf} represents the outgoing-link frequencies (for links that were frequently

[34] We chose these threshold values based on experiments on training data.
[35] 'internally' means within the boundaries of the Wikipedia corpus.

repeated at more positions in the content), D^{eURL} refers to all external links, and D^{tf} represents all terms that occurred with high frequency. Thus, all documents in the Wikipedia repository are transformed into several tables of vectors available in the leaves of the index.

4.3.2 Home Pages Index Class

Using the content of a webpage for home page finding is problematic for several reasons. The home page is usually the first page of a site, and it often contains mainly navigational links for the sitemap. Some potentially useful evidence for home page finding is query-dependent, including the presence of query words in the document text (which refer to anchor texts) or in the document's URL. It is recognized that full-text relevance ranking is not particularly effective for home page finding (Craswell et al., 2010). Other potentially useful evidence is query-independent, and this was demonstrated in the TREC-2001 home page finding task (Craswell et al., 2010). The best result was submitted by Westerveld et al. from UTwente/TNO (Craswell and Hawking, 2003), who used the URLs of pages as evidence. In addition, URLs sometimes use shortcuts or abbreviated terms to represent underlying meanings beyond the domain name (e.g., 'nist.com'). We used different representations when we processed the URLs for home-pages; that is, two basic methods for home-page finding. The first method is standard, and uses the structure of the URL, while the second is suitable for most abbreviated and embedded terms.

4.3.2.1 Processing URL Structures

Consistent with the general structure of URLs, we stripped off all the symbols and numbers, as well as the trailing terms (e.g., index, default, welcome). We then classified them into three categories depending on the location of the term in the URL, if:

- a term is located in the main domain section (e.g., 'www.ottawa.ca/'), it is 'main'.
- a term is mixed or embedded with other terms, such as 'air france' in 'airfrance.ca', it is 'main'.
- a term is represented by a shortcut or an abbreviation (e.g., 'www.uwaterloo.ca'), it is 'main'.
- a term is located in the sub-domain section, such as 'trec' in 'trec.nist.com/', it is 'sub'.
- a URL ends with the document name preceded by a symbol (e.g., ~), such as 'www.uottawa.ca/~cadams', it is 'tail'.

Thus, there are only five possible evidence types in the URL forms. We used different scores for each status, and assigned a ranking value of 1, 2 or 3 for a term that is located in the main-domain, the sub-domain or the document-name respectively[36].

Home-page finding methods require finding equivalent pages by converting hyperlinks to a canonical form. For example: 'http://bmo.com', 'http://www.bmo.com/', 'http://www.bmo.com:80/', 'http://www.bmo.com/index.htm/', 'http://www.bmo.com/welcome/', 'http://www.bmo.com/default', and 'http://www.bmo.com/ (language code)/ index.html', should be all represented as 'www.bmo.com/'.

4.3.2.2 Embedded Keyword Extraction

URL keyword and terminology extraction is a challenging task (Craswell and Hawking, 2003). Researchers have employed different algorithms, including statistical 'n-grams' and natural language processing methods, to tokenize and analyze URL terms and extract keywords that can be used to index the document's content. In addition to web page popularity that was computed by ALEXA[37], we used query log files to identify the original keywords behind the embedded keywords in domain names. Alexa.com computes traffic for all popular search engines. To exploit the information that exists in Alexa network, our approach posted HTTP requests for each main domain name available in the collection, to discover the underlying meaning behind the name (the domain names were extracted by the previous method). The downloaded information was available as a list of frequent phrases (queries). Then, our approach built a table of vectors, and each vector contained the domain name and its frequent queries. We used the term frequency (*tf*) to compute the occurrence of frequent queries in the table, which contained the important queries for each site accessed by users. For example, 'airfrance', 'uottawa' and 'nist' were defined in log files as the queries: 'Air France', 'University of Ottawa' and 'National Institute of Standards and Technology', respectively.

We tested our home page finding method on TREC 2011 web track topics, and achieved high precision of (P@10=1.0) for some queries, such as 'jax chemical company'.

[36]We assume the main domain keywords are better than the subdomain keywords; and, likewise, the subdomain keywords are better than the document words, because each document is part of a subdomain, and each subdomain is part of a main domain.
[37] http://www.alexa.com

4.3.3 Other Collection Indexing

Web pages are typically indexed by their content, but not all pages are suitable for indexing in this manner, including multimedia pages that contain videos, sound or images. As well, some web pages contain content such as programming code which is not usable for indexing. We used two types of index structures: word based and key-phrase based.

4.3.3.1 Word Based Index Class

The topic of a document is often conveyed by words located in the meta content, e.g., the URL, title, and heading attributes, and usually at least one term is shared by the meta content and the document content. If we represent the shared terms by a short vector and the document content by another vector, it is possible to compute the similarity and impact of those terms in the document content. We used the content from three meta attributes (‘title’, ‘headings’ and ‘URL’) and we combined these attributes together, because (i) we assume that not all documents contain relevant terms in one of these fields, and (ii) usually keywords in the title, headings and URL are complementary to each other. If the heading ‘h1’ is not available, or is similar to the title, we chose an alternative, such as ‘h2’ or ‘h3’. A very short content attribute might not contain enough information, while a long content attribute could contain unnecessary or redundant information. In addition, to provide comprehensive indexing it is necessary to index the main content. Our approach used this content from the selected attributes to trigger the topic of a document, and the important keywords in the document content. The keywords from selected attributes create a path to at least one query term, and the system accesses the index for other query terms.

Two documents from different sites might have content with different impacts, even if they have similar term frequencies in the content. A document with shorter content than others that shares similar term frequencies with the content of meta-terms should be judged more relevant. Similarly, a document with content that is highly similar to its meta-attributes should also be judged more relevant. Our model measures the closeness of document content to its meta-content attributes, as well as the effect of term impacts on documents that share similar term frequencies in meta-terms. It does this by computing the cosine similarity (Singhal, A., 2011) between the first vector that represents the frequencies of all terms in the document content, and the second

vector that represents the subsequent term frequency of each meta-term with complementary '0' values[38].

$$\text{Similarity(M,D)} = \text{Impact(M,D)} = \frac{M^2}{\sqrt{M^2} * \sqrt{\sum_{i=1}^{n} tf(D)_i^2}} \quad (4)$$

where: M is the term frequency of meta-term *M* in the content of document *D*;

n is the number of terms in document *D*; and,

$tf(D)_i$ is the frequency of term i in document *D*.

To compute the similarity measure (the impact) of each term in the meta-content, we processed the document content, as follows:

- Stripping off the html codes from the content of the document;
- Removing stop words, symbols and numbers;
- Removing stemming characters from each term; and,
- Computing the occurrence *tf*, of each term in the document content.

Once the *tf* value of each term was computed, the impact of every term in the meta-content was found using equation (4). Each meta-term was assigned to its cosine similarity measure with the document content. However, as mentioned, not all terms in the content are available in the meta content, and finding all the query terms in the meta-content is unusual. Therefore, it is impractical to rank web documents by their meta-content only; we also need to add the term frequencies of the document's content. To reduce storage requirements, we ignored all the terms that only occurred once. In addition to the docID and the termID, each term in meta-content has a vector with a certain dimension, including the cosine measure and the significant term frequencies. Document relevance was computed by summing the impact of the first query term that was available in the meta-content, and the percentage of frequencies of other terms from the query in the content; otherwise, only term frequencies were considered. Queries occasionally contain digits when searching for more precise results (e.g., 'hp mini 2140'), and we addressed this by adding an extra dimension to the vector for the document's title. Hence, the meta-content

[38] Term impact is different from tf-idf; in which, inverse document frequency computes the weight of term with respect of how important a term is to a document in a collection or corpus.

of ‘n’ terms was broken down to ‘n’ vectors, and each vector was transmitted to a corresponding node in the index, as shown below:

{termID}{SIM(Term)}{<t_1, f><t_2, f><t_3, f>… <t_n, f>}{Title}{docID}

where termID is a term’s hash key that was generated using the same algorithm that generated the hash key for each node in the index.

Consider an example of three documents, as shown in Figure (4.1). To generate the first vectors for all three documents, we combined the URL, title, and heading (h1) in one vector for each document. The stop words, numbers, symbols, unicodes, characters and duplicated words were removed:

1- ‘dealbook nytimes martha stewart imclone ImCloneNYTimes’.

2- ‘baltimoresun business bal imclone special Martha Stewart baltimoresun’.

3- ‘economics about cs finance insider trading Martha’.

To compute the impact of each term in its content, we first compute the term frequency for each term in the document. Then, we use formula (4) to compute the impact of each term in the meta content. Let’s consider that the term frequencies for three documents are distributed as follow:

Document1	Imclone	Stewart	drug	martha	Nytimes
	6	5	2	3	2

Document2	baltimoresun	imclone	martha	stewart	Vacation	systems	stock	Friend
	2	6	4	6	2	2	2	3

Document3	insider	Trading	Martha	stewart	news	company	Drug
	4	4	5	4	2	2	2

Table 4.1 Documents terms and their frequencies

<URL>http://dealbook.nytimes.com/2008/07/31/martha-stewart-imclone-and-that-60/ </URL>
<title>Martha Stewart, ImClone and That $60 - NYTimes.com</title>
<h1>Martha Stewart, ImClone and That $60</h1>
If only Martha Stewart had hung in there.
More than six years after Ms. Stewart famously sold nearly 4,000 shares of ImClone Systems for $58 apiece — leading to a media-saturated trial that ended in her conviction on charges of conspiracy and lying to investigators — ImClone's stock has climbed back above the $60 mark that was a major point of controversy in her trial.
For those who may have forgotten, Ms. Stewart's stock sale on Dec. 27, 2001, came just one day before the Food and Drug Administration said it would not review ImClone's application for its cancer drug, Erbitux.

<URL>http://www.baltimoresun.com/business/bal-imclone,0,4902408.special</URL>
<title>Martha Stewart-ImClone - baltimoresun.com</title>
<h1>Martha Stewart-ImClone Vacation</h1>
 Stewart billed firm for luxury vacation
Martha Stewart billed her company for the luxurious vacation she took just after selling her ImClone Systems Inc. stock in 2001, a Stewart business manager testified Tuesday.
Friend says Stewart knew CEO was selling
Three days after dumping her ImClone Systems stock, Martha Stewart confided to a longtime friend that she knew ImClone CEO Sam Waksal had tried to sell his own shares, the friend testified Thursday

<URL>http://economics.about.com/cs/finance/a/insider_trading.htm</URL>
<title>Insider Trading: What did Martha Do?</title>
<h1>Insider Trading</h1>
Martha Stewart has been in the news for several months because the U.S. Securities and Exchange Commission believes that Martha Stewart was told by her friend Sam Waksal that his company ImClone's cancer drug had been rejected by the Food and Drug Administration before this information was made public. This rejection was a huge blow to his company and the price of its stock went down dramatically. However, Martha Stewart wasn't financially hurt because she had her broker sell her 4000 shares before this news was made public. If this is true, and it should be noted it hasn't been proven yet, then Martha Stewart is guilty of insider trading.

Figure 4.1 An example of three documents

The meta-terms available in each document are:

1- 'imclone', 'stewart', and 'martha'.

2- 'martha', 'stewart', 'imclone', and 'vacation'.

3- 'insider', 'trading', and 'martha'.

The impact of each term in its document's content is computed by formula (4), as shown below:

Document1: imclone - V1 = 6 0 0 0 0, V2= 6 5 2 3 2 I(imclone)= 0.67
stewart - V1 = 0 5 0 0 0, V2= 6 5 2 3 2 I(stewart)= 0.56
martha - V1 = 0 0 0 3 0, V2= 6 5 2 3 2 I(martha)=0.33

Document2:	martha - V1 = 0 0 4 0 0 0 0 0, V2= 2 6 4 6 2 2 2 3 I(martha) =0.37 stewart - V1 = 0 0 0 6 0 0 0 0, V2=2 6 4 6 2 2 2 3 I(stewart) =0.56 imclone - V1 = 0 6 0 0 0 0 0 0, V2=2 6 4 6 2 2 2 3 I(imclome) =0.56 vacation - V1 = 0 0 0 0 2 0 0 0, V2=2 6 4 6 2 2 2 3 I(vacation) =0.18
Document3:	insider - V1 = 4 0 0 0 0 0 0 0, V2 = 4 4 5 4 2 2 2 I(insider)=0.43 trading - V1 = 0 4 0 0 0 0 0 0, V2 = 4 4 5 4 2 2 2 I(trading)=0.43 martha - V1 = 0 0 5 0 0 0 0 0, V2 = 4 4 5 4 2 2 2 I(martha)=0.54

Finally, each term is mapped to the corresponding node in the index, with the other information mentioned previously.

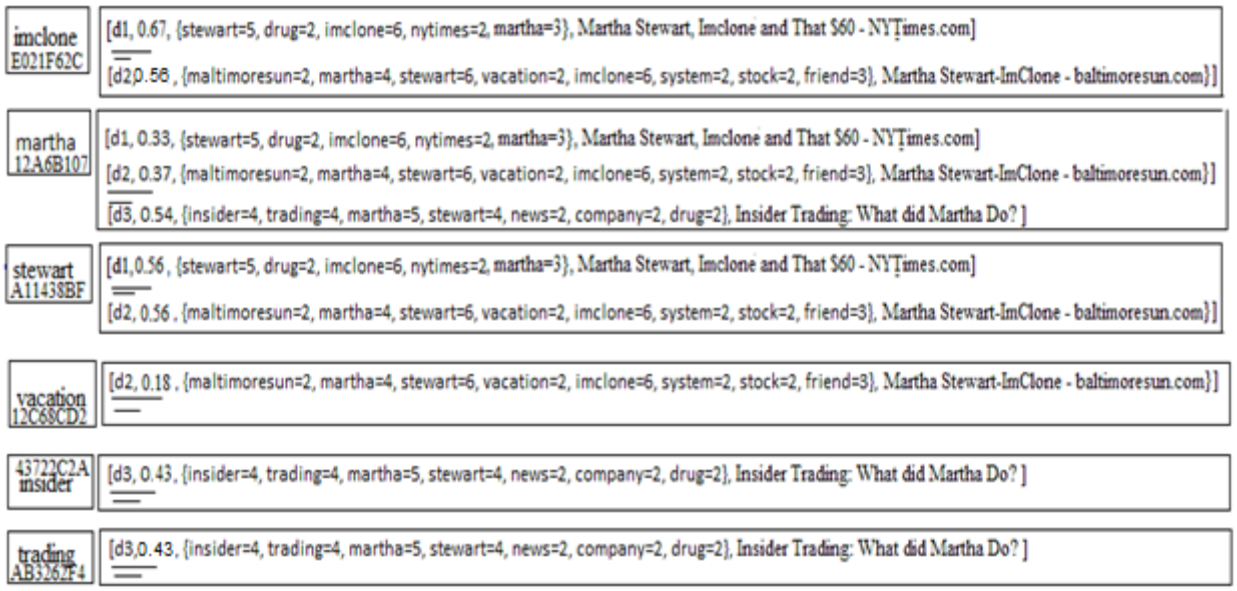


Figure 4.2 An example of word-based index

4.3.3.2 Key-Phrase Based Index Class

Though our method did not use computation for terms that occur only once, the terms can still be used for phrasal queries. Some documents are based on a fixed order of sequence terms, which might occur only once, or could be repeated in the document's content. Terms do not need to occur at more than one position in the content (e.g., 'map of brazil' could be located once at one position in a document), hence the occurrence of terms in other positions is not important to the

document's relevance (if they occur in some sub-topics). However, our method used the key-phrase index to compute term frequencies for all the terms in the content. For example, the query 'martha stewart and imclone' requires computing the proximity search for all terms, as computing both the term frequency for the term 'imclone', and the key-phrase frequency for the phrase 'martha stewart', is important to computing the document's relevance. To compute the key-phrase frequency, we first stripped off all stop words, symbols, characters and single letters from the document content. We then computed the frequency of single terms and double contiguous terms, and the length (i.e., 3, 4, etc.) to where the terms occurred together frequently. Then, for each key-phrase, we composed a vector of fixed dimension, as shown below:

{phrase ID, f}{<t1, f><t2, f><t3, f>…. <tn,f>}{Title}{docID}.

where phrase ID is a phrase key generated by using the same algorithm that generated the hash key for each node in the index. Finally, each key phrase is mapped to the index table.

As explained in this section, we present the pseudo code for indexing the documents using three index classes, as follows.

Initialize: Indexing

Wikipedia Pages:

Loop for all documents in Wikipedia collection

Compute term occurrence, outgoing links, and external links

If (term occurrence=threshold) then Index document in Class (W)

End of loop for all documents in Wikipedia

Home Pages:

Loop for all documents (D) in collection

Index the main domain (URL) in Class (H).

Parsing URL by ALEXA

Index the result in Class (H)

Other Pages:

Combine URL+Title+Heading in vector 'V'

Remove duplication from 'V'

Compute *tf* for each term (T) in document (D)

Loop for all terms (t) in ‘V’

Compute the impact of each term (t)

Build a node in index if ‘t’ within threshold

End of loop on all terms (t) in V

End of loop on all documents (D) in collection

4.4 Query Processing

Query processing is an important step, as it includes detecting the type of query, query normalization and query expansion. Generally, our system categorized queries into three document targets according to the type of matching between the queries and the target results:

- **Title:** means the query points to the relevant pages that contain all query terms positioned in the contiguous area as full key-phrases (e.g., ‘map of brazil’).
- **Domain:** means the query points to the relevant documents that are located in a particular site or domain (e.g., ‘jax chemical company’).
- **Frequency:** means the query points to unknown documents that were judged using the frequencies of query terms in the document (e.g., ‘fact of uranus’).

Each query was processed according to the three types of index classes mentioned in the previous section. We used overlapped term positions and the priority factor for each term in the query. However, the following criteria were used for processing a query:

- If the query length was one term, searching was done in two indexes (the home-page index and the word-based index), because a one-term query should refer to a term in the main domain. For example, ‘uOttawa’ or the term should be frequent in a document’s content, whether it was a home page or not, such as ‘afghanistan’.
- If the query length was more than one term, searching was done in all three indexes (the home-index, the word-based index and the key-phrase based index), because two terms or more should refer to the phrase available in a particular site (e.g., ‘american military university’, which is a home page). Likewise, two terms or more should be searched in the word-based index where one of the query terms matches any term in the key of the index that was represented by content attributes, while other terms match the terms in the index

content (e.g., ‘va dmv registration’). As well, the query should match the key-phrase index (e.g., ‘Ritz Carlton Lake Las Vegas’).

- If the query involved a prepositional or conjunctional term, the single term that occurred before or after these must be distinguished and weighted more than the other terms in the query. Searching occurs in the word-based index in which the distinguished term matches any key in the word-based index, while other terms match the content body of that key. For example, the query ‘**uplift** at yellowstone national park’ or ‘martha stewart and **imclone**’ in which ‘uplift’ and ‘imclone’ match the keys of the word-based index, and the remaining terms match the content of that key.

4.4.1 Query Expansion and Normalization

Search engines use query expansion to increase the quality of search results. It is assumed that users do not always formulate search queries using the most effective terms. The goal of query expansion is to increase recall without overly decreasing the precision, by including result pages that are more relevant (higher quality), or at least equally relevant. The current commercial search engines use word frequency (tf/idf) to assist with ranking. By ranking the occurrences of the user entered words and synonyms, documents with a higher density (i.e., high frequency and close proximity) tend to migrate upward in the search results, leading to a higher quality of search results near the top of the final ranked list.

The trade-off between precision and recall is one of the problems of query expansion. To improve retrieval performance, we applied query expansion for queries that were classified as Wikipedia articles, and not classified as home pages. We used anchor terms or phrases that frequently occurred in each article to expand the query topic (these anchor terms and phrases were previously indexed when the Wikipedia documents were indexed). Query expansion and normalization did improve the performance of some queries. For example, the query ‘all men are created equal’ achieved higher precision (P@5=0.8 and P@10=0.7), because the query was expanded with the phrases ‘Gettysburg Address’ and ‘Declaration of Independence’, which frequently occur in the Wikipedia article ‘all men are created equal’. In this way, the system was able to retrieve relevant documents that could not be retrieved without the query expansion step. This was because some documents were entitled ‘Gettysburg Address’ and ‘All men are created equal’, and their contents are linked to each other. Hence, these documents could be accessed by

normalizing the queries by their related topical phrases. The Wikipedia thesaurus composed by our system described the indexing of vocabularies that were used for query expansion, and included information about the synonyms or phrases for related topics. Topically it is important, because Wikipedia topics are manually joined using the terms in the thesaurus, and the vocabulary tags that link topics in Wikipedia are carefully chosen and subject to quality control (the vocabulary used in Wikipedia is called a controlled vocabulary). Using the composed thesaurus, the system could determine which keyword or phrase could be used to expand the initial query. Table 4.2 shows a sample of four initial queries and their expansions selected from our query expansion table.

Main Query	Expansion Query
lymphoma in dogs	lymphoid leukemias, Tumor, non-Hodgkin lymphoma, Working Formulation
dangers of asbestos	Crocidolite, asbestos cement, asbestosis, asbestiform
von willebrand disease	coagulation, von Willebrand factor, nosebleeds, asymptomatic, platelet
all men are created equal	Gettysburg Address, Declaration of Independence, Thomas Jefferson, Declaration of Sentiments

Table 4.2 Wikipedia titles (left) and their related topics (right)

4.4.2 Document Ranking

In this section, we explain our custom model for ranking web pages. As mentioned, documents are ranked based on the length of the query, as shown below:

1- One term length query: If a query term matches any key in the home-page index, the content of the key will be at the top of the ranked list. Then the documents retrieved from the word-based index will follow (the ranked documents were retrieved based on their term impacts).

2- Two terms or more length query: First, if the query terms match any key in the phrase-based index, the content of the key will be at the top of the ranked list. If the content contains a URL referring to the main domain, the results will be ranked high; otherwise the results of the content will be ranked based on the impact of the phrase. Second, if any of the

overlapped terms in the query match any key in the word-based index, the content of the key that is a term impact will be used to rank the document. Thus, summing the term impacts for all query terms will rank the document. Figure 4.3 shows an example of the query 'martha stewart' ranking two documents.

3- If a query contains conjunction or preposition term, searching will take place in the word-based index, and the prominent term will be evaluated by term impact. The remaining terms will be evaluated by their frequencies (e.g., '*martha stewart and imclone*'). We used this strategy because this type of query is focused on a specific term, 'imclone', rather than the other terms '*Martha' and 'stewart*'. Our approach uses four conjunctions ('and', 'on', 'for', and 'at') for determining the prominent query terms, when a single term is located before or after the conjunction. For this kind of query, we used the following formula to rank the retrieved documents:

$$Rank(D,Q) = Impact(t) + \left(\frac{\sum_{i=1}^{j} tf_i}{100}\right) \quad (5)$$

where t is the prominent query term (e.g., '*imlcone'*) in document *D* (derived from equation 4 above), tf_i is the term frequency of query *Q* for term *i* in document *D*, and *j* is a query length.

Our system summed the weights of all query terms, because 'imclone' is not the only prominent term in the query; other terms are also distinct. By applying formula (5) in our example, we can compute the rank of D1 and D2 based on the query '*martha stewart and imclone*' (D3 is ignored because the term 'imclone' is not available).

Rank (D1, Q) = 0.67 + (5 + 6 + 3) / 100 = 0.67 + 0.11 = 0.78

Rank (D2, Q) = 0.37 + (4 + 6 + 6) / 100 = 0.37 + 0.16 = 0.53

Likewise, if no prominent terms are available in a query (e.g., 'martha stewart'), the impacts of the two terms were summed, as shown below:

$$Rank(D,Q) = \sum_{j=1}^{n} Impact(t_j) \quad (6)$$

where t is a query term's impact in a document *D* (derived from equation 4 above), and *n* is a query length *(Q).*

Figure 4.3 An example of document ranking

Based on the method above, we present the algorithm (pseudo code) for calculating the ranking of each document for a query, as follows.

Initialize: Searching

Parse query (Q) to determine the type
If (Q) contains conjunctional term
Determine the prominent term (t)
Loop for all documents (D) in node (t)
Assign rank (R) to document (D) using formula 5
End of loop for all documents
Else loop for all terms (t) in query (Q)
Loop for all documents (D) in node (t)
Accumulate the term impact for each term (t) in (D) += R
End of loop for all documents
End of loop for all terms
Assign rank (R) to document (D) using formula 6

4.5 Spam Documents Filtering

The ClueWeb09 collection contains many spam documents. Cormack et al. (2010) studied the spam filtering in ClueWeb09, and determined that the spam filtering could significantly improve the performance of a system. Our system filtered out the spam documents that would decrease the retrieval quality. Thus, computing term frequency and term impact can detect spam documents, because spammers generally use many junk words that affect the impact of document terms. Our system used an impact threshold between 0.9 and 3.0 to find the best documents. If prominent terms had totaled impacts higher than 3.0, the document was flagged as spam, and if the total impact was lower than 0.9, it was ranked as junk. Otherwise, the document was ranked relevant, as shown in figure 4.4.

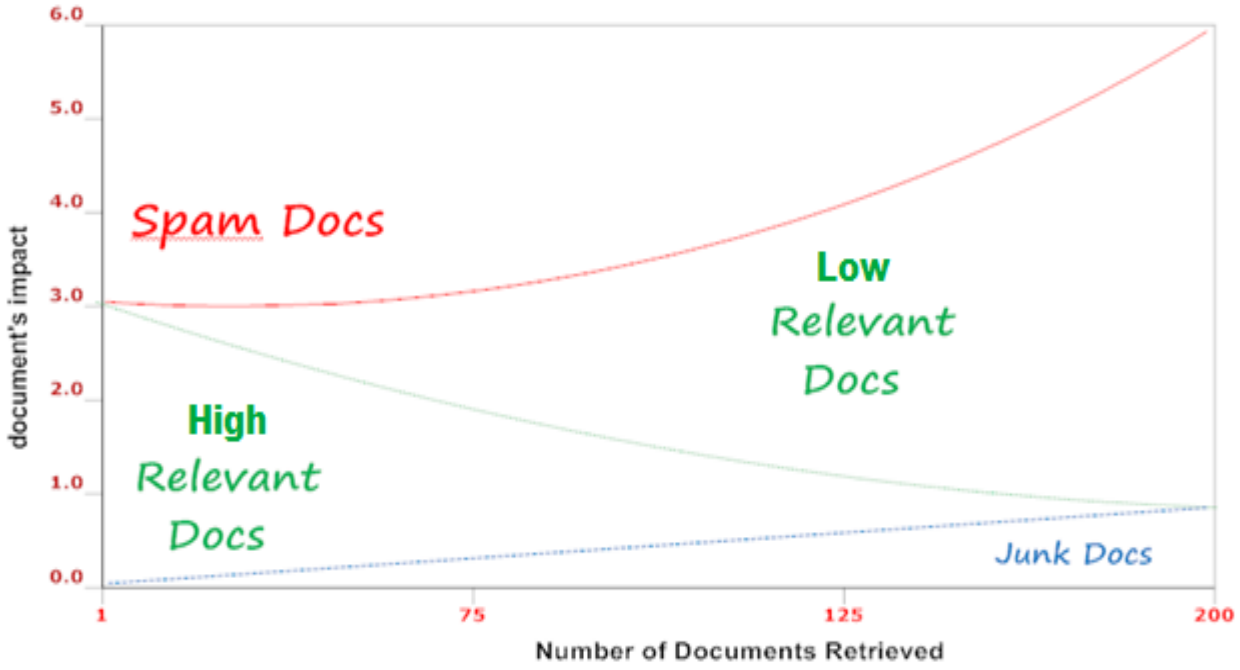


Figure 4.4: Document Categories vs. Term Impacts

4.6 Enhancing the Ranking Results

Traditional methods for ranking Web documents are not ideal in terms of search engine optimization (SEO). Smoothing ranking lists and changing document factors can provide end-users with more helpful and accurate search results. In this model, we used two strategies based on human references to improve our rankings.

4.6.1 Using Site Reputation

There is a vast amount of data available on the web, but not all of it is reliable and valuable. Many sites have untrue content that could be ranked high by our model (in the previous steps). Informational queries for a diversity task, for example, are always looking for broad, reliable, and valuable information, which is usually available on sites that can be trusted. Summarization, site reputation as local and global ranking is important for our relevancy algorithm. We used these factors to enhance the ranking algorithm by filtering out the poor sites, and we exploited the information from Alexa's web analytics by crawling and computing the reputation of domains in our dataset. In our previous example, the site traffics are (D1= 32, D2=1484 and D3=40), and we enhanced the final ranking list by using the following module:

$$Rank(D,Q) = \alpha / log(\beta) \quad (7)$$

where α is the value from module (5 or 6), and β is equal to the site-reputation (traffic).

The final ranks of the documents in our example 'martha stewart' are:

Rank (D1, Q) = 0.89 / log(32) = 0.59
Rank (D2, Q) = 0.93 / log(1484) = 0.29

4.6.2 Using Wikipedia

As mentioned earlier in this chapter, documents that are classified as home pages or archived in Wikipedia articles are considered relevant. Consequently, our system re-ranks the list by moving these relevant documents up so they are at the top of the final ranked list. Human references are robust arguments that can cause bias in the ranking of some documents. Since we previously indexed all the external documents that were archived for each Wikipedia article, the ranking algorithm increases the ranks of documents with links available in the external-links node of the Wikipedia repository index, which boosts them to the top of the ranked list. Some of the Wikipedia references could be targeted by spam links; but in our experiments, this issue rarely occurred because the use of Wikipedia references was limited to our initial results, which proved to be spam free. Even though some links in Wikipedia are marked as 'nofollow' for crawlers, they could be targeted by spammers.

4.7 Evaluation

In this section, we present the experimental results of the method described in this chapter, for the adhoc and diversity tasks on the TREC Web Track benchmarks. The TREC track organizers assessed our model during the evaluation campaign TREC 2011.

As mentioned before, we report the standard evaluation metrics used in Web information retrieval. The primary effectiveness measure for the adhoc task is the ***expected reciprocal rank*** of the first k documents retrieved (ERR@k) (Chapelle et al., 2010) and (Bruce et al., 2010). We also report NDCG, and standard binary measures including mean average precision (MAP) and precision at rank k (P@k). The primary effectiveness measure for the diversity task was a variant of the ***intent-aware expected reciprocal rank*** (ERR-IA). We also report a number of other intent-aware measures found in the literature, including αnDCG@k (Discount Cumulative Gain), NRBP (Rank-biased Precision) and MAP-IA.

The results of our TREC 2011 submission, as assessed by the TREC Web track organizers, are shown in Tables 4.3 and 4.4, and Figure 4.5. We also show our results in comparison to the average of all 62 models submitted by all participants for the 50 test queries (see Appendix II).

αnDCG@10	**αnDCG@20**	**ERR-IA@10**	**P-IA@10**	**P-IA@20**
0.4380	0.4675	0.3578	0.2414	0.2098
MAP-IA	**NRBP**	**ERR-IA@20**	**strec@10**	**strec@20**
0.0685	0.3207	0.3670	0.6731	0.7155

Table 4.3 Our diversity task results for the 50 test queries - TREC 2011

NDCG@20	**ERR@20**	**P@10**	**P@20**	**MAP**
0.1743	0.09639	0.2909	0.2636	0.0838

Table 4.4 Our adhoc task results for the 50 test queries - TREC 2011

Our model achieved good results for most queries, though for some the precision of the retrieved document list was zero, because the relevance judgments contained only documents selected from other parts of the collection (Category 'A'); our method could not model the meaning of these queries. In some cases, though the retrieved documents seemed relevant; but they were not, as the relevance judgment file holds the best results selected from Category A. It

is difficult to capture all relevant documents that satisfy all users' needs in one relevance judgment file; users could have different points of view at different moments.

TREC compared our model with other sixty two models on the same ad-hoc and diversity retrieval tasks for the same dataset. Some models used the whole data collection (Category A), and it is unreasonable to compare our results to those results because some documents that were in the expected solution could not be retrieved by our approach, since they were not in the reduced dataset (subset B). Table 4.5 presents the comparative results according to the Web track overview paper (Clarke et al., 2011). However, we tested our approach by using the testing queries in 2012 with different parameter settings. As we said, the parameter value was selected according to the number of results in the final list.

For the adhoc task, the general comparison criteria used were based on ERR values for all the participants, regardless if they used collection set A or subset B. Table 4.7 shows the comparison between our model and the mean average over all models. Table 4.8 shows our position in the top eight approaches in TREC 2011. Figure 4.5 shows our results for the diversity task for each topic. Figure 4.6 shows other metrics for the adhoc and diversity tasks.

Model	**ERR@20**	**nDCG@20**	**P@20**	**MAP**
Name not Disclosed	0.131	0.233	0.298	0.110
Univ. of Ottawa	0.122	0.204	0.275	0.079
Univ. of Amsterdam	0.119	0.202	0.273	0.085

Table 4.5 Comparison of the ad-hoc retrieval task results for the testing queries 2011 for two top systems from TREC 2011 on subset (B), and for our system submitted to TREC 2011.

Model	**ERR-IA@20**	**α-nDCG@20**	**NRBP**
Univ. of Amsterdam (Kamps)	0.438	0.522	0.407
Univ. of Ottawa	0.405	0.502	0.370
Univ. of Delaware (Fang)	0.375	0.458	0.340

Table 4.6 Comparison of the diversity retrieval task results for the testing queries 2011 for two top systems from TREC 2011 on subset (B), and for our system submitted to TREC 2011.

Group	ERR@20	nDCG@20
University of Ottawa	0.122	0.204
Average over 36 models	0.106	0.187

Table 4.7 Average mean results (36 models/50 test queries) and our submission

Group	Run	Cat.	ERR@20	nDCG@20	P@20	MAP
Chines Acad. of Science (ICT)	ICTNET11AADR3	A	0.157	0.283	0.345	0.175
University of Glasgow (Terrier)	uogTrA45Vm	A	0.149	0.305	0.347	0.203
University of Waterloo (MDS)	UMatMDSqlt	A	0.144	0.242	0.307	0.135
Microsoft Research	msrsv2011a3	A	0.143	0.273	0.327	0.172
Srchvrs	srchvrs11b	B	0.131	0.233	0.298	0.110
University of Ottawa	DFalah11	B	0.122	0.204	0.275	0.079
University of Amsterdam	UAmsM705tiLS	B	0.119	0.202	0.273	0.085
University of Waterloo (Clarke)	uwBAadhoc	A	0.119	0.172	0.230	0.079

Table 4.8 Top 2011 adhoc task results ordered by ERR@20 for the best eight of 36 models

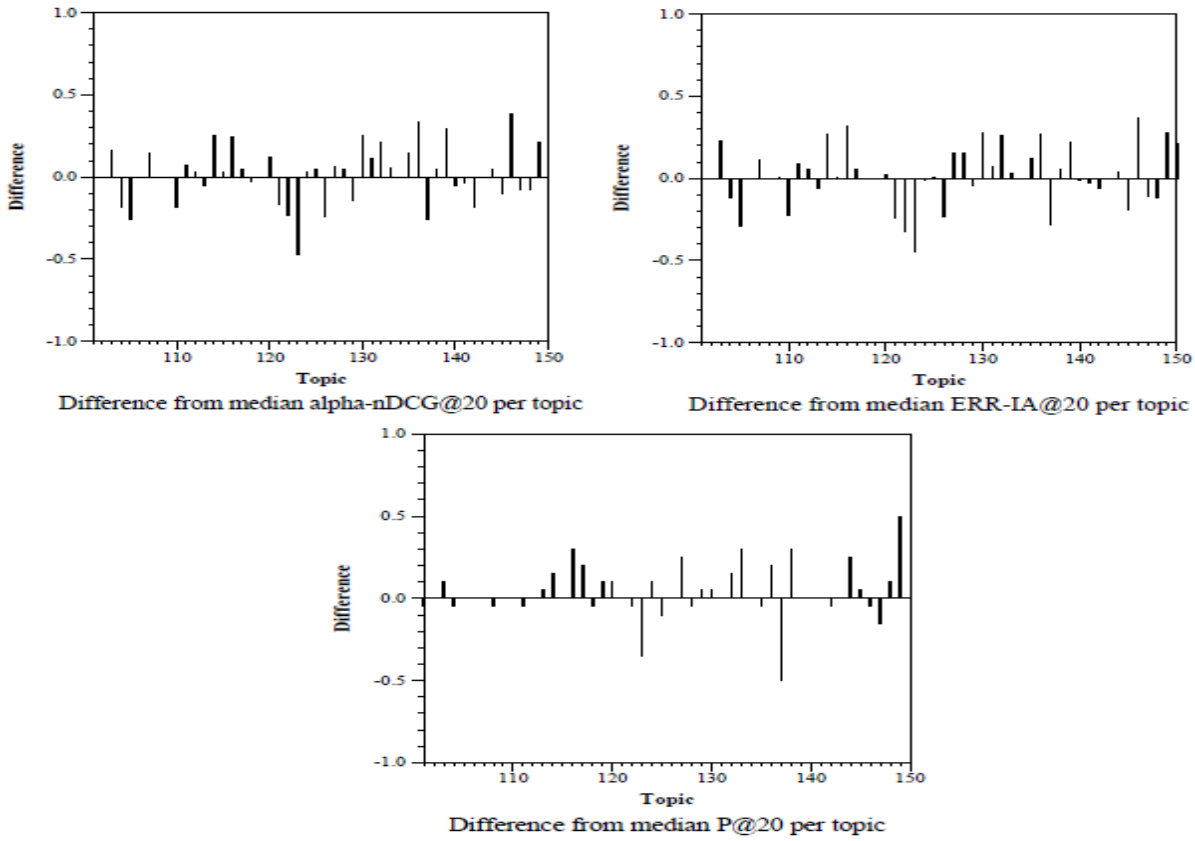


Figure 4.5 Our Web Track 2011 diversity task results for each query

Summary Statistics	
Run ID:	DFalah11
Task :	adhoc
Category:	B
External resources used:	(B) generally available resources
Number of Topics:	50

Adhoc measures		Diversity measures	
Retrieved	6547	α-nDCG@10	0.4728
Relevant	3157	α-nDCG@20	0.5021
Relevant retrieved	577	ERR-IA@10	0.3976
Prec@10	0.2960	P-IA@10	0.2546
Prec@20	0.2750	P-IA@20	0.2199
MAP	0.0794	MAP-IA	0.0727
NDCG@20	0.2044	NRBP	0.3700
ERR@20	0.1219	ERR-IA@20	0.4058

Figure 4.6 Web Track 2011 adhoc and diversity results

4.8 Discussion

Although meta content and document content provide very useful information for web searching, current web search engines approaches and our approach from the previous chapter did not show how to use the meta content for indexing web content. There are several promising techniques for using meta content to extract structured data from unstructured web content. For example, coupling the structured data extracted by analyzing URLs and titles into fragments, which allows search engines to exploit the structured data as they are querying web pages. Wikipedia data is another example. If the Wikipedia pages mention a book by its ISBN number, the links under the ISBN number could reference the book in the books index, and use it to call APIs to obtain it. More sophisticated methods for extracting references to named entities and factual assertions could also be applied.

Another approach is to capture structured data before it enters the system. The easiest way to do this is to give users the tools to structure their data based on the query structures, such as techniques to add structured fields and create hierarchies of query types.

In this chapter, we proposed a custom structured-content-based indexing approach and ranking module that uses query structures and Wikipedia knowledge to overcome insufficiently clear document topics, by discovering a web page's plausible missing information from its similar web pages. Our model provides a variety of analytic capabilities, including concept extraction, concept correlation, spam filtering, and term-to-document similarity.

In this approach, we improved several aspects of the version of our system discussed in the previous chapter. We kept stop words in the key-phrase index, which allowed us to successfully

process queries such as ‘to be or not to be, that is the question’. The conjunctional and prepositional terms also helped our approach to separate the prominent terms in some queries. For example, in ‘Martha Stewart and imclone’ and ‘earn money at home’, the prominent terms are: ‘imclone’ and ‘home’, respectively.

Our model is an improvement over the model in the previous chapter, despite the fact that many topics in the 2011 test data were underspecified (e.g., multi-faceted). This new model reflects the distinction of terms (term impact) better than the model that used term frequencies. Specifically, meta content terms reflect the topic of web documents stronger than other terms in the same documents.

We addressed the issues in the first model, by reducing the number of attributes/tags in meta data. In the previous model, we used the meta data in the ‘URL’, ‘titles’, ‘keywords’, ‘description’ and ‘alt’, but we did not exploit the content of documents. We showed that data from selected tags has little impact on capturing document topics, unless they are addressed differently as in our second approach in this chapter. In this model, we replaced the tags ‘keywords’, ‘description’ and ‘alt’ by document content because they did not make a significant impact. We noticed that the keywords in the URLs and titles were able to capture the topic of document contents. Wikipedia is a strong knowledge dictionary, and a database of quality web collections that are categorized topically. However, our experiments showed that keywords in URLs and titles can trigger and leverage the topics of documents, if they engage with Wikipedia knowledge appropriately. Only the index of the previous model suffered from spam, indicating that the contents of documents were highly targeted by spammers.

In the next chapter, we experiment with using more knowledge from Wikipedia, to close the gaps for queries that ranked low, and enhance the retrieval results in order to improve our final approach.

Chapter 5

Query Structure-Based Web Page Indexing

5.1 Introduction

Many existing information retrieval models do not explicitly take information about query word associations into account. As well, numerous existing retrieval approaches, including those using tf/idf and others using the unigram language modeling framework, do not clearly take information about term dependencies in queries located in the relevant documents into account. Term dependencies are the relationships between words in related contents. Existing information retrieval models that explicitly consider term associations, predominantly access information about first order relationships between words located in epic structures known as syntagmas (Esko, 2009). Syntagmatic relationships are formed between words that co-occur near each other in web content with likelihood greater than chance. For example, in ‘gs pay rate’ the term ‘gs’ would have syntagmatic relationships with ‘pay’ and ‘rate’, assuming they co-occur with ‘gs’ more than by chance. Another example is ‘brooks brothers clearance’ in which the term ‘clearance’ would also have associations with ‘brooks’ and ‘brothers’.

The combining of words in syntagmatic relationships give words their meaning. The association between words is a ‘paradigm’ if they can replace each other in a sentence without affecting the meaning of the sentence. Typical examples are synonyms such as ‘paralegal’ and ‘legal writer’, or related verbs like ‘becoming’ and ‘how to become’. Earlier, we stated the importance of using Wikipedia in web information retrieval. Although the associations and relationships between words are typically well expressed in a set of web articles, web search models are unable to find the associations between words unless they use external references such as Wikipedia, or a set of well-known topical queries collected from log files of popular search engines.

As well, within the structural queries for Web content there is a specific type that makes relationships between words focus on particular sites, known as home pages. The association between words is required to process the query and find the relevant pages. If a site focuses on a particular distribution of words, all the pages in that site will be relevant, but the degree of relevancy will vary from one page to another in website. Traditional search engines retrieve

pages from the relevant site randomly, or from the anchor texts that are available in the home page only.

Our approach makes use of the relationships between query words. This is achieved by using a collection of Wikipedia titles, and a set of well-known log queries provided by TREC and available in the 'Million Query Track'[39] data, to build our index. We used these resources to investigate different structures and associations between query words and web content.

Our collection was indexed by this model, and it did not use stemming and stop word removal. In order to experiment with this new search model, we evaluated it using the queries/topics made available by NIST in 2012 as part of TREC Web search track. Our approach achieved statistically significant improvements in adhoc and diversity retrieval tasks, and is among the top systems. The method we proposed in this chapter has been published in (Al-akashi and Inkpen, 2012) as part of the TREC 2012 Proceedings, and in (Al-akashi and Inkpen, 2014).

The rest of this chapter describes the components of our approach. Section 5.2 discusses our new model, and Section 5.3 describes the query processing steps. Section 5.4 examines query expansion, while Section 5.5 explains how our approach ranked the final results. Section 5.6 shows the evaluation and the comparisons with other approaches. Finally, Section 5.7 presents our discussion regarding this approach.

5.2 Our Indexing Approach

Indexing is critical to finding relevant information on the Web. Various indexing methods are used in a wide range of applications, including Home-page finding, Entity finding, and Web page classification.

In order to improve search efficiency, using word associations and query structures in the indexing and searching processes can be helpful for the following reasons:

- Phrases express structured concepts closer than individual words do.
- Phrases have a smaller degree of ambiguity than single words. While two words can be ambiguous, their combination is not, since each word creates a context for the unambiguous interpretation of the other.

[39] http://trec.nist.gov/data/million.query09.html

- ❖ Phrases increase the efficacy of proximity search in high-precision search engines.
- ❖ Documents that contain phrases should be ranked higher than documents that contain their constituent words in unrelated contexts.

To find key-phrases that are important for each topic, a set of articles can be assembled from particular dictionaries. This was recently used for document clustering, entity finding or document classification in small collections of documents, but it has not been used in large scale webpage indexing. Bahle et al. (2001) showed that phrasal queries can be rapidly evaluated using next-word indexes, but the indexes are twice as large as conventional inverted files. They combined the use of an auxiliary next-word index and a conventional inverted file, and their space overhead was only 10% of the size of the inverted file.

Overall, we built an index that handles phrases and concepts of different lengths using two references: Wikipedia articles and Million Query Track[40] data. Our index is structured as a centralized search file, and the key idea is to store data efficiently, inspired by the block-oriented storage contexts known as B-trees (Becker et al., 1996). We wanted to avoid the rebalancing issues of some indexing trees, such as binary trees; our index uses sub-trees in a fixed length. The first level of each internal node is labeled with three letters from the first term (single word or phrase), whereas the second level is labeled by a full phrase or term. Each class has a collection of documents that represent the documents indexed around the term or phrase node. This means that each leaf node contains several vectors, and each vector represents the document that is relevant to the index term or phrase. The depth from the root to each leaf corresponds to the term or phrase. We used a pool of five index classes to automatically build the index, and each class stores a particular type of indexed data, as listed below and shown in Figure 5.1:

- **Wikipedia Class:** The index class that contains three sub-classes: Common-Tags, Terms' Impact and CRC-Dictionary. These sub-classes hold the indexed Wikipedia documents from the ClueWeb09 collection.
- **Home-Page Class:** The index class that contains two sub-classes: Domain-Name and Wikipedia-External-Links. These sub-classes hold all the indexed home pages from the ClueWeb09 collection.

[40] http://trec.nist.gov/data/million.query.html

- **Document-Title Class:** The index class that contains all documents that were indexed using the phrases from their titles.
- **Terms-Combination Class:** The index class with nodes that were labeled with the keywords selected from the URLs and the document titles. The content of these nodes hold the vectors for significant terms[41].
- **Topical-Keywords Class:** The index class that holds all other documents except Wikipedia pages and home pages. The documents in this class were indexed based on our collective phrases from Wikipedia titles and the One Million Query Track at TREC. This class also holds the domain names.

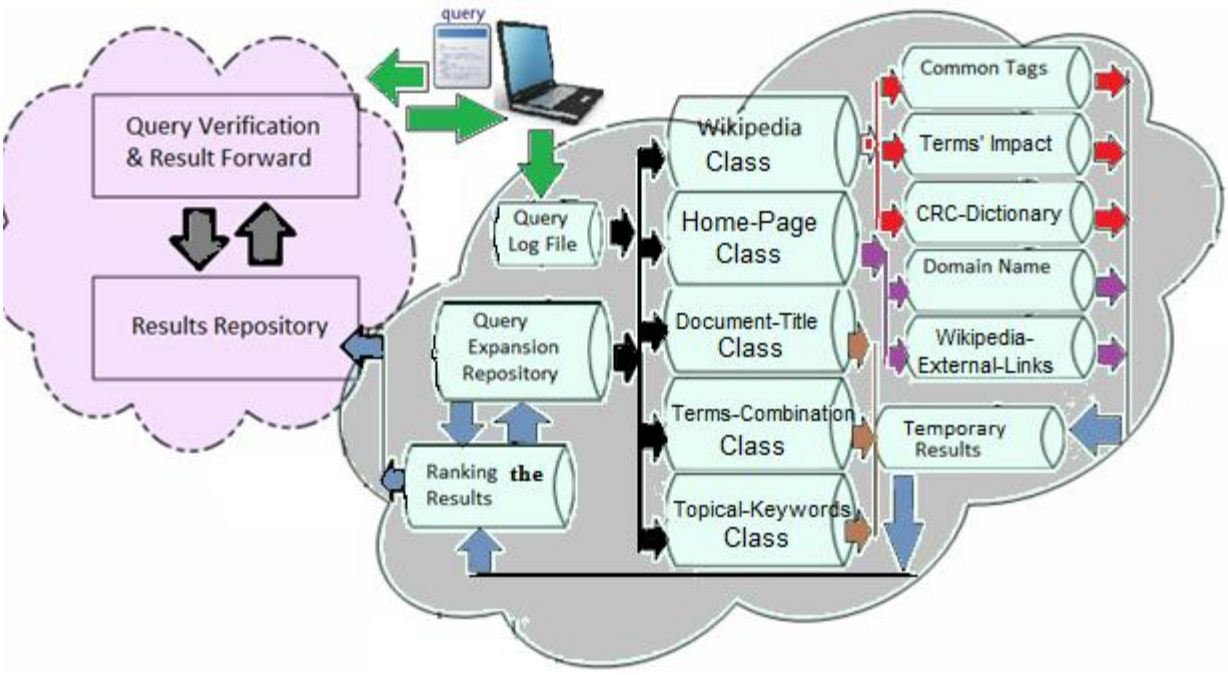


Figure 5.1 High Level Architecture of Our Index Structure

5.2.1 Wikipedia Index Class

Wikipedia is a collective knowledge reference of approximately five million pages, representing about three million English articles. The data in each article is structured into several fields, and it can have relationships with other articles through tags or links to expand certain topics. Each

[41] This class is similar to our approach in the previous chapter.

page has a unique vocabulary identifier. Wikipedia sometimes uses different faceted vocabulary names to describe the same article. To efficiently index Wikipedia documents, we used three methods to group and cluster similar articles, in order to reduce the index size and the time required for indexing and searching. The first method is based on the first paragraph of the document content, which means that the documents that share significant terms (impacts) in their contents are grouped together. The second method is based on the document titles, which means that the CRC code (see Section 3.1.3) for the first paragraph of each document's content is computed, then the documents that share the same CRC values are grouped together. The third method is based on the document tags, which means that the documents that share similar tags are grouped together in one cluster. The second method is essential for retrieving the initial results from the Wikipedia index, whereas the two other methods are responsible for retrieving the corresponding (similar) documents. This means that the three methods are synchronized and working cooperatively, and before indexing the document by the first method it will check whether the similar document was indexed by methods two and three. If so, it only adds the document's identifiers to the indexes of method 1 and 2, and avoids adding them to the index of method 3.

5.2.1.1 Terms-Impact Subclass

As we discussed, not all Wikipedia articles are real articles. Our investigation found that approximately 50% of articles that are very short contain only a few words (e.g., definition and description). As well, some articles cover topics more comprehensively than others.

Term frequency is important for detecting the topic of documents, but we need to look at different document configurations to better measure the importance of the terms. For example, two documents with different lengths may have different impacts on their topics, even if they are equal in term frequency. Singhal (1997) proposed measuring the importance of the terms by different weights and document length normalization when computing the likelihood of document relevancy. He also claimed that the use of cosine normalization in the presence of the inverse document frequency is not advisable in large document collections. Karbasi and Boughanem (2006) reported that document length normalization should be used with an effective level of term frequency in large collections. Our module from the Chapter 4 proposed another faceted term weighting method, as term frequency alone was not enough when computing the

topics in large collections. The process we used to calculate term impacts proved to be effective in our experiments. Therefore, we propose using term impact, rather than only term frequency.

Term impact is computed as the cosine similarity between two vectors, as in the previous model. The first vector represents the frequency of any term in the document, and the second represents the frequency of all terms in the same document. For example, consider the following document's content and term frequencies: Julie[2], loves[3], Linda[5], than[6] and Jane[8]. To compute the impact of the term 'Jane', the first vector is formed as [0, 0, 0, 0, 8], whereas the second vector is formed as [2, 3, 5, 6, 8]. Thus, the impact value is the cosine similarity between the vectors, which is 0.68.

To avoid short and non-topical documents, we selected documents with at least one term with a frequency higher than the threshold (15 in this case), because short documents do not need to be retrieved according to the expected solution of the training queries. After computing the weight/impact value for each term in each document, a maximum of three terms with the highest impact weights were selected as representative of their documents. Overall, each document that focuses on a suitable topic should have a total impact between 2.5 and 5.5 for all terms higher than the threshold. Thus, whenever a document has a term frequency higher than 15, it will be highly relevant to that term. We ignored documents that did not have a term frequency higher than 15. This range of values (thresholds) was computed and selected experimentally. The candidate terms (three representatives) were used to build and label the index nodes (if they were not built previously), while the document identifiers were stored in the content of the corresponding nodes. Therefore, each node in the index could store a cluster of documents that have equal or close term impacts (i.e., all terms with impacts in the mentioned range).

More specifically, one term impact for each document is not enough to represent the document topic, because not all terms cover a clear topic. This issue was addressed by combining more than one term impact for each article to determine its topic. The following example shows martha's node, where each vector refers to a document that was created from the value at the beginning; that is, the impact of the term 'martha' in that document, the document identifier (TREC id), other document's terms with their impacts, and the document title.

- 0.31|doc id| clueweb09-enwp02-00-11320 |prison,0.27 | stewart,0.40 | ImClone_stock_trading_case
- 0.32|doc id| clueweb09-enwp02-00-11326 | ray,0.32 | The_Honeymoon_Killers
- 0.32|doc id| clueweb09-enwp02-27-14284 | raye,0.37 | Martha_Raye
- 0.31|doc id| clueweb09-enwp01-25-14293 | jefferson,0.30 | wayles,0.27 | Martha_Wayles_Skelton_Jefferson

- - -

- - -

- 0.28|doc id| clueweb09-enwp01-06-06111| stewart,0.28|Jennifer_Hutt

Usually, two or more terms in the same class cover groups of similar topics. Therefore, we applied a technique of the ***Self-Organizing Map Algorithm (SOM)*** (Vesanto and Alhoniemi, 2000) and (Kohonen, 2001) to combine the similar topics, which is a neural network paradigm proposed by Kohonen[42] (Kohonen, 1991), known as the Kohonen map or topographical map. SOM has been used as a troublesome clustering approach in many applications, and several researchers have used it to discover patterns and clusters in Web pages (WEBSOM), or even in textual documents (Guthikonda, 2007), (Kaski et al., 2004) and (Kohonen et al., 1991). Each node in a given layer of the network is linked to all the nodes in the upper and/or lower layer. Nodes that are close by total weight are going to interact differently than nodes that are far apart.

For example, if x1 and x2 are input vectors and t1 and t2 are the locations of the corresponding output nodes, then t1 and t2 should be close together if x1 and x2 are similar. Neurons tend to cluster in groups a network, and the connections within a group are much greater than connections with the neurons outside the group. Kohonen's network tries to imitate this in a simple manner. The SOM that we adapted to the web page retrieval task is a two-layer neural network that accepts N-dimensional input patterns, and maps them to a lattice of output neurons that represent the feature space. The output map is typically two dimensions of neurons, and all input neurons are fully connected to all output neurons.

SOM is an essential aspect of our technique for grouping Wikipedia articles. Other methods, are complementary to this method (see 5.2.1.2 and 5.2.1.3), because, as mentioned, they are very precise with the degree of similarity when grouping the documents (based on constant value). Our method tends to group the documents even if the term occurrences between them are different. As an example of how we used SOM, assume a query ‘martha stewart living’ that attempts to cluster 725 documents available in the class ‘martha’ into sub-topics, based on the query terms and vector terms. The documents that match all query terms are extracted from the index class to represent the value of the neurons. The SOM maps the input documents in the neural network into two dimensional spaces (x, y) with a grid length of 10 neurons[43].

[42] http://page.mi.fu-berlin.de/rojas/neural/chapter/K15.pdf

[43] The value (10) is the side length of the output grid. This value is proposed to get the results clustered between (0 and 10); the output grid is 10X10 (we can use other values).

In Table 5.1 below, the value at each field (columns 2, 3, and 4) is the impact of the corresponding term in its document. It is a challenge for SOM to reduce data with more than two dimensions (i.e., 'martha', 'stewart' and 'living') down to two dimensions (column 5). We do this by automatically identifying the two features (column 5) that will have the greatest differentiating effect. The input data and output results for the above example are also shown in Table 5.1.

1. Document CRC	2. Martha	3. Stewart	4. Living	5. Neuron (SOM)
d1	0.460	0.501	0.299	0,0
d2	0.460	0.501	0.299	0,0
d3	0.313	0.235	0.102	4,2
d4	0.476	0.277	0.150	2,4
d5	0.460	0.501	0.299	0,0
d6	0.395	0.387	0.294	0,2
d7	0.299	0.363	0.210	4,0
d8	0.460	0.501	0.299	0,0
d9	0.476	0.277	0.150	2,4
d10	0.455	0.500	0.299	1,0
d11	0.270	0.201	0.114	4,4
d12	0.313	0.404	0.102	4,0

Table 5.1 Document identifiers (left column), term impacts (mid), and SOM values (right)

Thus, clusters are ranked with respect to their sizes (number of documents), as shown below:

1- [d1 , d2 , d5 , d8 |0,0] 2- [d4 , d9 |2,4] 3- [d12 , d7 |4,0] 4- [d11|4,4] 5- [d3 |4,2]

6- [d6 |0,2] 7- [d10|1,0] .

Figure 5.2 shows a low level representation of the Term Impact Index for two query examples: 'martha stewart' and 'imclone stock'.

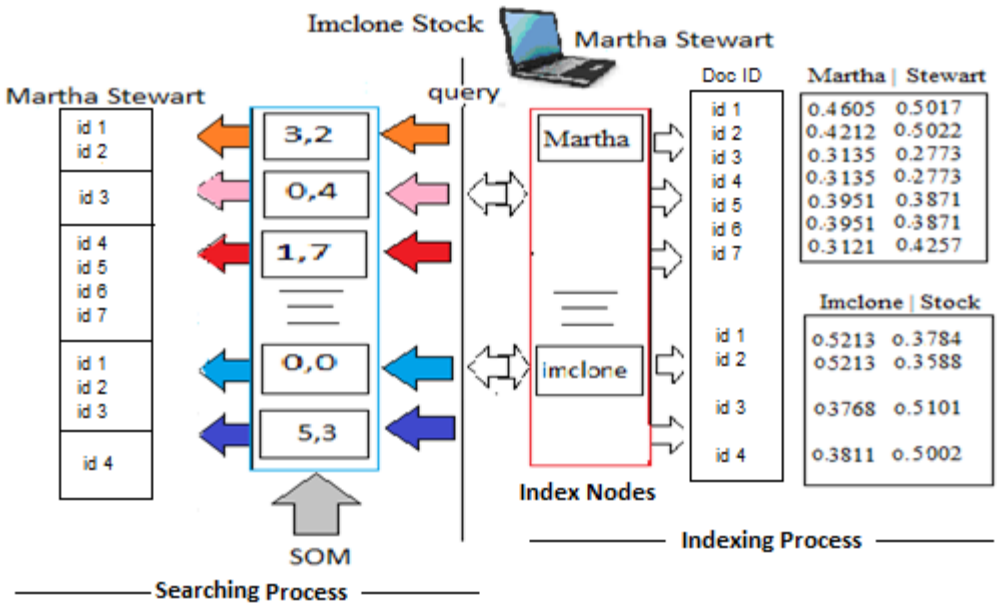


Figure 5.2 Low level representation of the Term Impact Index Structure for two queries

5.2.1.2 CRC-Dictionary Subclass

A cyclic redundancy check (CRC)[44] is an algorithm used to detect highly-similarity text. We applied this technique to find duplication in Wikipedia documents. Generally, Wikipedia articles that are very similar repeat the first paragraphs; other paragraphs may or may not be repeated. We believe documents that share first paragraphs share similar topics, and our approach used the CRC function with a fixed-length of 8 bits (a hexadecimal number) to detect documents that share the same paragraphs[45]. While the document indexing in method 1 above is in processing, the process synchronously scans the document's content and generates the CRC value only for the header paragraph. The CRC value will be added to the index above for each document to make a pointer to this index class (in the searching process). Our system then built a dictionary (hash-table) with the generated CRC values representing the keys, and the value of each key holding the document TREC identifier (the dictionary was built if it wasn't created before; otherwise, the document is added to the content of the similar key). By scanning all the Wikipedia documents, those that share the same CRC values were aggregated by the content of

[44] http://google.com/patents/US5898836
[45] The value '16' was experimentally selected because it is enough to encode and differentiate all 50 million paragraphs.

the keys. Finally, the dictionary was transferred to the index in the physical disk, as a set of vectors.

$$CRC=\sum_{i=1}^{m}\sum_{j=1}^{n} D_i^j \quad (7)$$

where *m* is the number of keys in the index class *CRC*, *n* is the number of documents that belong to the key *i*, D_i^j is a document *j* in the index key *i*, D was indexed as: D= {URL, Title, TREC-id}. TREC-id is the document's name in the collection.

In fact, this method supports the Term-Impact approach to finding similar documents. In traditional search engines, displaying similar documents to users is not preferred, but in the TREC data the duplication is needed because similar documents expect different TREC IDs as solutions. Our model uses this method for two purposes:

1- To find similar documents, if a document was retrieved by the Term-Impact method, those with the same CRC (for TREC submissions only) were also retrieved. This method helps our search engine to remove duplication in the Wikipedia documents retrieved in the final ranked list.
2- To help method 1 avoids indexing similar documents if they were indexed before.

Consider an article 'Civil Rights Movement' that provides a CRC value for the header paragraph of E035F2BC. The documents that have similar content to the article Civil Rights Movement should produce the same CRC value of E035F2BC. This also means that the content of the dictionary key E035F2BC should accumulate the articles that share the same content. The following example below shows two dictionary keys with their documents.

E035F2BC | $d_1,\ldots,d_n$
21FF6C2A | $d_1,\ldots,d_m$

To retrieve answers to the query 'civil rights movement', the relevant documents represented by CRCs are retrieved from the index above (method 1), and used as the keys to retrieve documents from the dictionary content that are under the same key. This method is important in reducing the index of method 1. Figure 5.3 below shows the structure of our index, using the query 'lipoma' as an example.

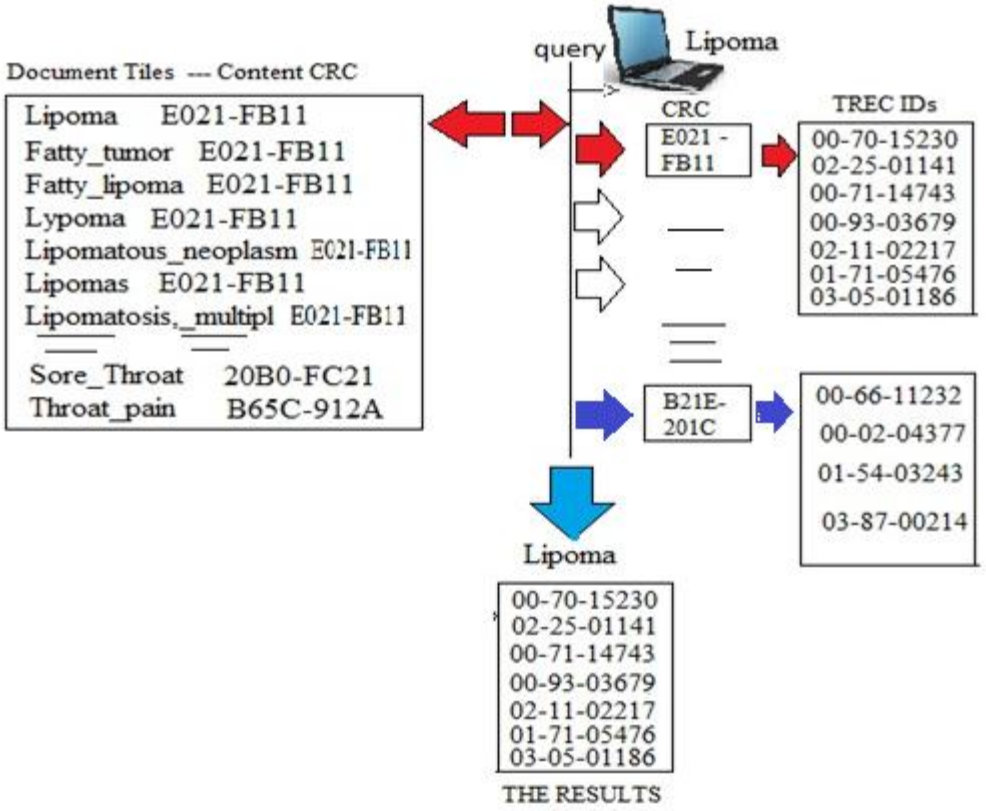


Figure 5.3 Example of low level representation of the CRC Index Structure

5.2.1.3 Common-Tags Subclass

Each document in Wikipedia has a unique vocabulary name and identifier, and Wikipedia articles can use different vocabulary terms to describe the same article. This means that some articles are repeated many times with different vocabulary names and TREC identifiers in the Wikipedia repository (i.e., they share the same tags or bold text). In our proposed approach, these articles have similar topics, thus we scanned the content of each article for vocabulary terms and significant tags. Overall, each article title and its significant tags were used to create and title the index nodes, whereas the content of each node accumulates the document identifiers (as encoded by TREC). Hence, the index class contains vectors that represent documents.

To optimize this process, we built a custom hash table and mapped it to the index pool. The content of each article was scanned for the vocabulary terms and significant tags that were used to title the keys of table, whereas the content of each key accumulated the document identifiers through scanning the content of all documents. Consider the article ‘Ficus aurea’, the table contains the combined-key ‘Ficus aurea’, ‘golden fig’, ‘focus venusta’ and ‘florida strangler fig’,

while the content accumulates the TREC identifiers of documents that share these keys. Figure 5.4 shows the structure of the *common tags index class*. It is worth mentioning that this method finds all articles with exactly the same tags; thus the names of tags in these articles must coincide with the query terms.

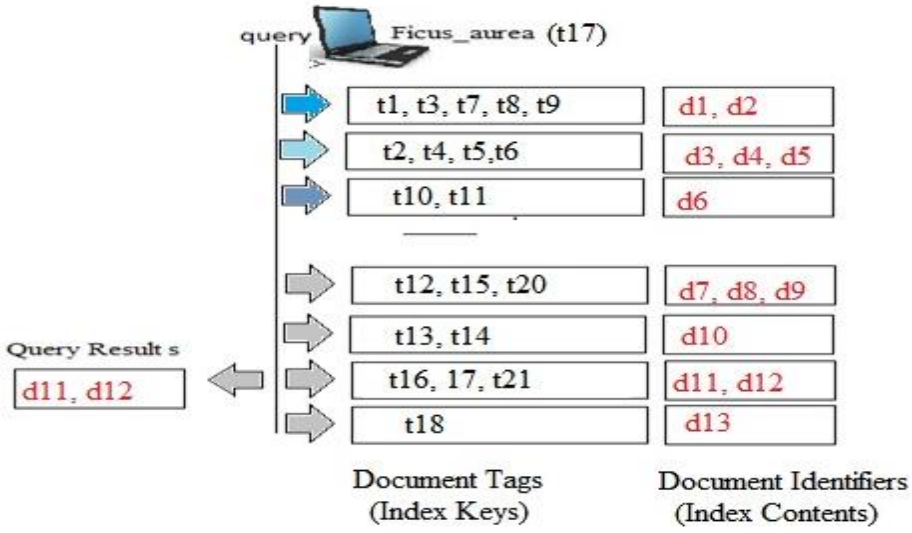


Figure 5.4 Example of low level representation of Common Tags Index Structure

5.2.2 Home Pages Index Class

Home page finding is a difficult task, and it is recognized that full-text relevance indexing is not a particularly effective way to accomplish it, as demonstrated in the TREC-2001 home page finding task. The best system using URL evidence to predict the home pages was proposed by Craswell and Hawking in 2010. However, though a home page is not restricted to the home domains, it should still be the first page on the final ranked list. Since the first page takes more effort to process, we used two methods to predict the home pages. The first is based on domain names, and the second is based on Wikipedia articles. The domain names are assigned a higher priority in the results list.

5.2.2.1 Domain Names Subclass

Query-independent evidence and URL-types have proven useful for home page finding, and URL-types offered a better than random improvement on an anchor text baseline (query-dependent). In realistic settings, using URL-types achieved consistent gains (Upstill et al., 2003). Given only the URLs of web pages, researchers showed that documents can be identified by their topics (Cafarella et al., 2007). Domain names and URLs are very important for home page

finding (Baykan et al., 2009). For each domain, the home page is defined by the shortest URL, and is often the first page in a hierarchical site. We used two phases to aggregate sparse documents. The first phase indexed the documents that belong to the same domain in the same index node, and the second phase sorted and generated a tree of URLs, in alphabetic order.

The documents with the shortest URLs were classified as the home pages, and we created a node in the domain name class for each home page. The name of the node was a combination of the domain name and document title, and the content of the node holds the TREC id for the home page and all documents that belong to that domain (URLs and titles of documents were also indexed). We linked the titles of the documents to the domain names as some domain names involve abbreviations. For example, the title '*civil rights movement*' is referred to the domain name by '*crmvet*'. Thus, the index node could be accessed by either query string.

$$DNC=\sum_{i=1}^{n} K(D_i) \quad (8)$$

where *n* is the number of domain names in the index class, *K* is a node in the index class for domain name *i*, and *D* is the list of documents that belong to domain *i*. *D* was indexed as: *D= {URL, Title, TREC-id}*

Usually the domain names include different extensions (e.g., '.com', '.org', '.gov', '.edu', '.net', '.ca', '.info'), or are predefined with or without 'www'. Our system built the index nodes from the domain names, regardless of the type of extension. The contents of the nodes contain the document ids and their URLs. However, we assigned a different ranked value for each extension; that is, the priority order is: '.com', '.gov', '.org', '.edu', '.net', '.info', and so on.

Consider two examples of queries selected from TREC 2012: 'arkansas' and 'barbados'. The first frame below is an example of an index node 'arkansas'; whereas the second frame is an example of an index node 'barbados'.

```
6.000|clueweb09-en0004-74-26852|http://www.arkansas.com/
6.000|clueweb09-en0000-37-09961|http://arkansas.com/
5.000|clueweb09-en0004-80-17694|http://www.arkansas.gov/
5.000|clueweb09-en0000-90-13464|http://arkansas.gov/
```

6.000|clueweb09-en0007-88-32012|http://www.barbados.com.au/
5.000|clueweb09-en0005-26-06844|http://www.barbados.gov.bb/
4.000|clueweb09-en0000-16-19445|http://barbados.org/index.html
4.000|clueweb09-en0000-16-19227|http://barbados.org/
4.000|clueweb09-en0005-27-26737|http://www.barbados.org/
4.000|clueweb09-en0000-16-19234|http://barbados.org/./index.html

Unintentionally, in some cases the same URLs were crawled twice or more, which means that some documents are repeated in the TREC collection, with different TREC identifiers. Our approach indexed all main domains, whether they were repeated or not. We recognize that this is not the preferred approach in a traditional search engine in terms of SEO. Typically, a single query term is not always on the top of the list to represent the home page, but in our perspective, we assume one term is more important for determining user's intention; therefore our approach located the top locations on the ranked list for home pages, if they exist.

5.2.2.2 Wikipedia External-Links Subclass

Wikipedia is often a good reference for most home pages. Researchers have used the external links in the Wikipedia repository for home-page finding, and they potentially work better than searching anchor texts for the same task (Craswell and Hawking, 2003). Other researchers used Wikipedia for entity finding (Kamps et al., 2010) and (Serdyukovand and Vries, 2009).

Many official home pages are referenced by Wikipedia writers in the external links section. Since Wikipedia is a large portion of the ClueWeb09 corpus, our model was designed to index all external anchor texts and their URLs. Wikipedia uses the terms 'Official', 'Website' and 'Home page' to represent the external home pages for related articles. Our model used regular expressions to parse these terms, and to extract the corresponding URLs and anchors. Consequently, each home page was used to create a node in the Wikipedia external links class. The nodes were labeled by the titles of the referred home pages. The content of each node holds the home page vectors, and each vector is represented by the anchor text, URL and TREC id.

5.2.3 Document's Title Index Class

The titles of documents can contain terms and phrases that are relevant for indexing, particularly documents with only a few words in their content that are suitable for indexing. The URLs also contain keywords relevant for indexing; however, these keywords are interconnected, which

makes them difficult to parse. For example, the home pages: ‘www.crmvet.org’ and ‘http://civilrights.org/‘ are defined in their titles by ‘civil rights movement’.

The phrases in the titles are often connected using conjunctional words (i.e., ‘or’, ‘and’, ‘at’, ‘in’, ‘on’, ‘by’, ‘with’, ‘from’ or ‘for’), or punctuation characters (i.e., ‘:’, ‘|’, ‘(‘, ‘)’, ‘-’, ‘,’ or ‘&’. Thus, to find the most important key phrases for documents, it is essential to segment the titles of the documents into phrases. We used these characters and function words to partition the titles into lists of terms and phrases. More specifically, the terms and phrases do not have equal importance, and some of the terms are more important than others. Also, the same phrase in two documents does not necessarily have the same importance. To address this, we used our module ‘4’ (see Section 4.3.3) to compute the impact of each segment (term or phrase) in its content. We also computed the similarity between the vectors represented by each segment separately, as well as the vector that represents the document content (the similarity was based on term frequency). Finally, the fragment title was used to create and name the class node, and the content of each node holds a set of vectors represented by the fragment impact (term or phrase), document id (TREC id) and URL.

Consider three documents with the following titles: “Civil Rights Movement - Period 1”, “Civil Rights Movement - Period 2” and “Civil Rights Movement - Period 3”. The fragment ‘civil rights movement’ is repeated in all three documents. The impact of the fragment in each of the documents was computed by equation 1.

Before extracting the items from the document titles, the index class combined the documents that share the same items, and built a node for each set of combined items. The two frames below show the nodes of ‘civil rights movement - period 1’ and ‘civil rights movement usa’ in the index class. Each vector in the index node refers to the phrase’s impact, document-TREC id, and URL, respectively, for each document that shares the same phrase.

0.537| TREC id | http://www.dipity.com/ the_civil_rights_movement_period_1

0.152| TREC id | http://www.dipity.com /the_civil_rights_movement_period_1/flip

0.251| TREC id | http://www.dipity.com/ /the_civil_rights_movement_period_1/list

0.511| TREC id | http://www.dipity.com//the_civil_rights_movement_period_1/map

0.421| TREC id | http://www.dipity.com/user /timeline/the_civil_rights_movement_period_1

0.711| TREC id | http://www.dipity.com/ashley/civil_rights_movement

0.221| TREC id | http://www.dipity.com/ashley/civil_rights_movement/map

0.031| TREC id | http://www.dipity.com/devinj/civil_rights_movement_in_usa

0.621| TREC id | http://www.dipity.com/devinj/civil_rights_movement_in_usa/flip

0.700| TREC id | http://www.dipity.com/mark1/civil_rights_movement_in_the_usa

0.001| TREC id | http://www.dipity.com/mark1/civil_rights_movement_in_the_usa/flip

0.558| TREC id | http://www.dipity.com/mark1/civil_rights_movement_in_the_usa/list

0.021| TREC id | http://www.dipity.com/wa5037/civil_rights_movement_in_the_usa

5.2.4 Terms Combination Index Class

Usually, a query is combined with keywords located in different positions in the documents. For example, some terms are located in the URLs, while the remaining terms are located in the document's content or title. The combination index class focuses on these types of queries. First, the frequency of each term in the document is computed, and the keywords of the URL and document title are combined and parsed in one vector to remove the repeated terms, and the normalized vector is split into terms. Then, the impact value is computed for each term (equation 4), and specified in a range (0.5 - 1.0) to represent the document (the best three values were selected). Finally, the best three terms are chosen to create and name the index nodes. The content of nodes is a set of vectors, and each vector is created from the impact value of a representative term and all terms with their frequencies in the document content (TREC id is also wrapped). Overall, this type of index is similar to the index in the previous chapter, in which our system used the same threshold value to distinguish the junk, spam and relevant documents (see figure 4.4).

For another example, consider the following documents selected from ClueWeb09 and relevant to the query 'gs pay rate', as shown in Figure 5.5.

Example 1: URL→http://www.opm.gov/oca/pay/HTML/02maxgs2.asp.

The term frequencies for this document were computed, and terms that occurred only once were disregarded. Then, the impact for each term in the URL and title was computed, and the index node for each term (i.e., 'opm', 'oca' and 'pay') was created. The content of node

represented the impact of the target term, the term frequencies for the terms ‘GS’, ‘pay’ and ‘rate’, and the document identifier.

Example 2: URL→ http://www.gspay.com/.

The same steps were applied to this document. To rank the document for the query ‘gs pay rate’, we used our previous Model 6 (see Section 4.4.2).

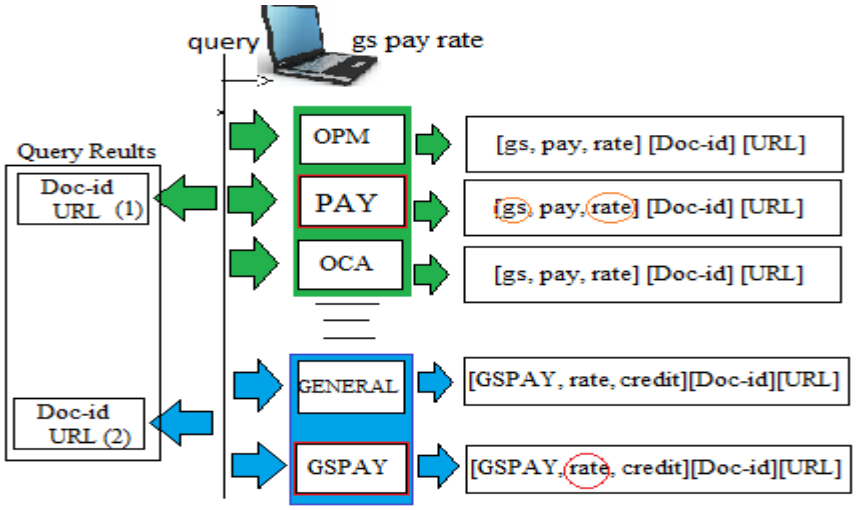


Figure 5.5 Terms Combination Index Structure

Generally, the matching mechanism for searching a query, such as ‘gs pay rate’, is as follows:

- ✓ Sequentially, the system matches each term in the query with the index nodes, and then matches the other terms in the query with the content of the matched node in the index. For example, a query term ‘pay’ matches a node ‘PAY’, and the remaining query terms ‘gs’ and ‘rate’ match the terms in the content.
- ✓ The mechanism of matching is based on combining the query terms; that is, combining two words to find the node and search for others in the content, or combining three words and searching for others, and so on. As a result, the query terms are matched with the combined-term node ‘gspay’, and the remaining term, ‘rate’, is matched with the content of the node.

5.2.5 Topical Keywords/Keyphrases Index Class

This type of indexing model addresses the drawbacks of traditional indexing models. Typically, the inverted index is two-level structure: the upper level holds all the index terms, and the lower level contains term occurrences. For huge datasets, such as web pages, this type of simple inverted file is not preferred; it takes quite some time to build, requires a lot of disk space, and is prone to issues such as results duplication.

Topical phrases are difficult to collect, and require a high level of parsing. Topical phrases are usually available in external resources, such as web-based encyclopedia. Our index class for this type was based on collecting phrases from the three following resources:

- Wikipedia titles: Since the Wikipedia collection is rich with marked phrases, our model collected all the titles from the Wikipedia articles with significant term impacts using 'Terms-Impact Class'. Documents that did not have representative weights in their terms were disregarded.
- Query logs: The query logs from the 'Million Query Track'[46] were collected, as they contain rich keywords about frequently used phrasal queries.
- The keywords in the main URL domains were also collected.

The gathered collection was combined into one list, with duplicated words removed. The final size for all elements in the list was approximately 7 million topical phrases and terms. In this class, we sought to index all the documents that focus on each element in the list. Our model used two phases to completely index the entire collection. The first phase detected the validity of the topics in each document in our corpus, and the second phase detected the weights (impacts) of keywords/keyphrases in the documents, and then in its domain, as explained below.

5.2.5.1 Phase 1 – Document Keywords/Keyphrases Validation

The first step in this phase was detecting the validity of the index terms in the documents, and assigning each document to a set of terms. We used three hash tables ($\partial 1$, $\partial 2$, and $\partial 3$) that functioned cooperatively. Briefly, table $\partial 1$ stores the elements and assigns the position value for each by its position in the table. Table $\partial 2$ tokenized the elements in table $\partial 1$ into keywords, and wrapped each single term with its position in table $\partial 1$. The terms of the first paragraph of each document in the collection were then parsed sequentially, and distributed in table $\partial 2$

[46] http://trec.nist.gov/data/million.query.html. Specifically, we used this list of queries as a dictionary / collection of phrases.

based on available single keywords. Table $\partial 3$ then checks whether the distributed terms in table $\partial 2$ have the same phrased positions in table $\partial 1$; if so, the documents will be aggregated in table $\partial 3$. Then, table $\partial 3$ aggregated the documents with the same key from table $\partial 1$, and the same position from table $\partial 2$. Assume we have the following five topical phrases:

1- 'angular cheilitis', 2- '403b', 3- 'civil rights movement', 4- 'angular velocity' and

5- 'q learning'.

First, the order of these keyphrases is assigned to the keys of hash table $\partial 1$, and then the keyphrases are assigned to the contents of keys, as shown below:

Keys	1	2	3	4	5
Content	angular cheilitis	403b	civil rights movement	angular velocity	q learning

Secondly, the content of table $\partial 1$ is tokenized into individual terms. Table $\partial 2$ then builds its keys from the individual terms. The content of each key involves two values: the location of the term in table $\partial 1$, and the length of query that holds that term.

Keys	Angular	cheilitis	403b	Civil	rights	movement	velocity	learning
Content	1,2 4,2	1,2	2,1	3,3	3,3	3,3	4,2	5,2

[angular, {(1,2) | (4,2)}], [cheilitis, (1,2)], [403b, (2,1)], [civil, (3,3)], [rights, (3,3)], [movement, (3,3)], [velocity, (4,2)], [learning, (5,2)]

As we tried to fix the drawbacks of our models in the previous chapters, we found that some queries were not located in the URL or title of the documents; they were located in the first paragraph, and may or may not repeat in other paragraphs. Thus, we added the attribute <p> for the first paragraph to the attributes (URLs and titles) that were used previously. The texts from these attributes were combined together in one vector (duplicated words, symbols and punctuation characters were removed). The generated vector was tokenized into single terms,

and distributed over the keys of a second dictionary $\partial 2$. The terms that did not match any keys in the table were discarded. More precisely, consider that we need to index the following four documents:

1- [document 1, URL, 'CareFair.com - Learning about Angular Cheilitis'].
2- [document 2, URL, '403b Savings Calculator - Free Online Retirement Calculators'].
3- [document 3, URL, 'Civil Rights Movement Veterans - CORE, NAACP, SCLC, SNCC'].
4- [document 4, URL, 'What is an American Civil Rights'].

As mentioned, the duplicated terms and punctuation characters were filtered out:

1- [document 1, URL, 'carefair com learning about angular cheilitis'].
2- [document 2, URL, '403b savings calculator free online retirement calculators'].
3- [document 3, URL, 'civil rights movement veterans core naacp sclc sncc'].
4- [document 4, URL, 'what is an american civil rights'].

The terms in each document are processed individually, such that if any term matches any key in the second table $\partial 2$, the system will copy the content of that key to the corresponding key in table $\partial 3$. The results are combined in the content of each key in table $\partial 3$.

As shown below, the document number represents the key of table $\partial 3$, and the values separated by the symbol '|' represent the content of table $\partial 2$ for those keys that matched the terms extracted from the document. This means that the values '5-2', '1-2 | 4-2' and '1-2' correspond to the terms 'learning', 'angular' and 'chelitis' respectively, for the first document.

[Document 1, {5,2 | (1,2 | 4,2)| 1,2}] [Document 2, {2,1}] [Document 3, {3,3 | 3,3 | 3,3}] [Document 4, {4,3 | 3,3 | 3,3}]

Thirdly, the content of hash table $\partial 3$ that separates each value by the symbol '|' was manipulated as one occurrence, For example, '1,2' is repeated twice in document 1, hence the value is transferred to a new form '{1,2→2}' as the location of term '2' in cell 1 of table $\partial 1$ and repeated '2', [location , length →frequency], as shown below:

Document-1 {5,2→1}{1,2→2}{4,2→1}, Document-2 {2,1→1}, Document-3 {3,3→3}, Document-4 {4,3→1}{3,3→2}.

Consequently, if the frequency of occurrence is equal to the length of the query, the full keyphrase is occurred in the document content:

Document-1, {1,2→2}, document-2, {2,1→1}, document-3, {3,3→3}.

The other occurrences are discarded because their length is not equal to the length of the keyphrases (e.g., document-4, {4,3→1}{3,3→2}).

Finally, the matching between the keys of table $\partial 1$ and the content of table $\partial 3$ is processed to determine the keyword/keyphrase names, rather than keyword/keyphrase locations. Hence, each document is assigned to a particular topic, as shown below:

[Document 1, URL, 'angular cheilitis']
[Document 2, URL, '403b']
[Document 3, URL, 'civil rights movement']

In this case, each document matches a particular topic in the index node. Thus, each node contains a set of documents that involve the same topic, as shown below:

['angular cheilitis', {doc_1, URL | .... |doc_n, URL}]
['403b', {doc_2, URL| ..... |doc_n, URL}]
.....
.....
['civil rights movement', {doc_3, URL| .... |doc_n, URL}]

5.2.5.2 Phase 2 - Weighting the Keywords/Keyphrases

Often, keywords/keyphrases that are available in the documents are not enough to distinguish their topics. For example, if a site focuses on the topic 'civil rights movement', all documents that belong to that site might contain the phrase 'civil rights movement'. Generally, researchers use the inverse document frequency (*idf*), which is based on the number of documents that contain a term in that site, to find terms with strong discriminating power. In our case, by processing all the documents in the collection, the content of table $\partial 3$ has already captured the documents that belong to the same site. Our method is different than the standard *idf*, as it uses the inverse document frequency based on the frequency of topical phrases, which is a more robust than one based on individual words. In this section, we focus on computing the impact of phrasal keywords in a document with respect to its domain.

1. Document-Topic Weighting

A long phrase could have different distribution in some documents. For example, the sequence of words '***to be or not to be that is the question***' has different distributions that should lead to different topics in some documents, as in "***To Be or Not to Be*** *a Church Mem**b**er:* ***That Is the***

Question *by Wayne Mack Paper*". If the phrase was available in the correct distribution (e.g., as consecutive words), the weight of the matching document was computed using the cosine similarity between two vectors. The first vector represents the phrase in table $\partial 3$, and the second vector represents the matching document's content. The result is stored as shown below:

[Document 1, URL, 'angular cheilitis', 0.371]
[Document 2, URL, '403b', 0.012]
[Document 3, URL, 'civil rights movement', 0.511]
…
[Document 10, URL, 'angular cheilitis', 0.252]
…
[Document 100, URL, 'angular cheilitis', 0.716]

2. Site-Topic Weighting

Finally, all the keywords that belong to the same domain (site) meet in the same keyphrases in dictionary $\partial 3$. Computing the inverse document frequency based on phrases reflects the topic of the document in the site better than *idf* based on single terms. *idf* computes the frequency in the whole site rather than in a specific sub-domain, though not all documents in the same domain make similar contributions to the topic. For example, the query 'University of Phoenix' refers to the domain 'phoenix.edu', but not all the documents in the domain 'phoenix' have equal relevancy to the 'phoenix' query. To address this, our model computed the impact of a document in its sub-domain or class with respect to the term frequency. First, the documents that belong to each keyword in dictionary $\partial 3$ were sorted alphabetically; thus each key in dictionary $\partial 3$ was structured hierarchically (main domain, sub-domains, classes and files). The size of a sub-directory (sub-tree) is important to computing the weight of the nested documents. We applied a top-down traversal algorithm[47] to compute the weights of the sub-trees.

Overall, each parent in the sub-tree counted the nested documents and summed their impacts. Then, the tree sorted its nested sub-trees (or documents, if no parents) with respect to the number of children (documents). Our model used an automatic cutoff value to optimally customize the content of sub-trees, with respect to the number of documents. The cutoff value was based on the number of documents in the sub-trees, as well as the contributions (impact) of the sub-trees. Initially, the elimination started at frequency 1, and increased until the number of documents in

[47] http://www.cs.umd.edu/~hjs/pubs/SametPAMI85b.pdf

the entire tree equaled 200. We chose the threshold value of 200 experimentally to maintain the balance between the precision and the recall value. However, the old tree was pruned and normalized to a new tree, and each parent had an optimal sub-tree size. The total impact weight of each sub-domain was computed as follows:

$$\text{Weight}(D, I) = \sum_j w_j / N \quad (9)$$

where *w* is the impact weight of document *j* in sub-tree *I*, and *N* is the number of documents in the sub-tree *I*.

The final tree was assigned to the index class, and the root of the tree (the main parent) was created and labeled as the main node in the index class. The children (documents) were bounded in the space of the vectors represented as: total impact value, document TREC-id and URL. Consider the following tree for the tree node 'hfma' that is relevant to the topic 'medical health care':

77 | clueweb09-en0007-02-21681|http://www.hfma.org | 0.712

…..

10 | clueweb09-en0007-02-21687|http://www.hfma.org/certification/ | 0.321

…..

12 | clueweb09-en0007-02-21697|http://www.hfma.org/events/ | 0.214

….

3 | clueweb09-en0007-02-21710|http://www.hfma.org/forums/ | 0.330

1 | clueweb09-en0007-02-21711|http://www.hfma.org/forums/five_keys_physician_hospital_relationships.htm | 0.330

1 | clueweb09-en0007-02-21712|http://www.hfma.org/forums/what_intensivist.htm | 0.330

11 | clueweb09-en0007-02-21713|http://www.hfma.org/hfm/ | 0.780

….

3 | clueweb09-en0007-02-21908|http://www.hfma.org/jobs/ | 0.712

….

21 | clueweb09-en0007-02-21912|http://www.hfma.org/library/ | 0.880

7 | clueweb09-en0007-02-21940|http://www.hfma.org/publications// | 0.413

….

The value at the beginning of each line refers to the number of documents in each sub-tree (one represents the leaf), while the value at the end of each line is the document-topic weight computed in the previous section.

Our system optimized the content of the sub-trees for the number of documents using an automatic cut-off value. The cut-off value is variable, and relies on the number of documents in each sub-class, which means the value is low if the sub-class holds a small number of indexed documents (children), and vice versa. However, the sub-trees that make minimal contributions were cut off from their parents. Figure 5.6 shows an example of how the website was structured, and how the weights of the sub-trees were computed.

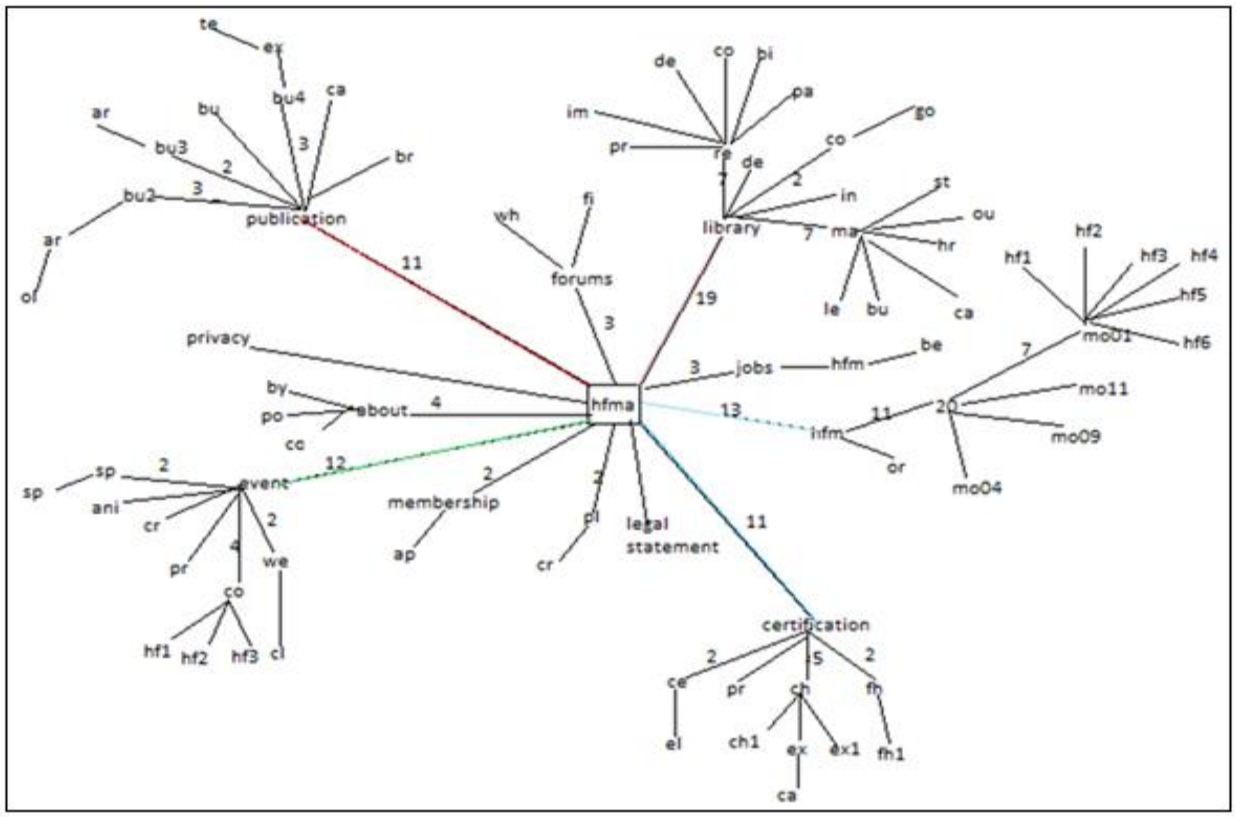


Figure 5.6 Documents, sub-directories, and sites weighting

Assume the number of documents in all sub-classes (sub-trees) exceeded 200; the system trims the sub-trees that were involved with three documents or less, as shown below:

77 | clueweb09-en0007-02-21681|http://www.hfma.org | 0.712
4 | clueweb09-en0007-02-21682|http://www.hfma.org/about/ | 0.330
10 | clueweb09-en0007-02-21687|http://www.hfma.org/certification/ | 0.321
4 | clueweb09-en0007-02-21690|http://www.hfma.org/certification/chfp/ | 0.712
12 | clueweb09-en0007-02-21697|http://www.hfma.org/events/ | 0.214
3 | clueweb09-en0007-02-21710|http://www.hfma.org/forums/ | 0.330
11 | clueweb09-en0007-02-21713|http://www.hfma.org/hfm/ | 0.780
3 | clueweb09-en0007-02-21908|http://www.hfma.org/jobs/ | 0.712
21 | clueweb09-en0007-02-21912|http://www.hfma.org/library/ | 0.880
7 | clueweb09-en0007-02-21917|http://www.hfma.org/library/management// | 0.220
7 | clueweb09-en0007-02-21925|http://www.hfma.org/library/revenue/ | 0.880
7 | clueweb09-en0007-02-21940|http://www.hfma.org/publications// | 0.413

After reducing the size of the tree based on the sub-classes, the total weight of each document is recomputed using the following formula:

*Rank (D) = (Site-Topic Weighting * Document-Topic Weighting)/100*

0.548 | clueweb09-en0007-02-21681|http://www.hfma.org
0.013| clueweb09-en0007-02-21682|http://www.hfma.org/about
0.032| clueweb09-en0007-02-21687|http://www.hfma.org/certification/
0.028 | clueweb09-en0007-02-21690|http://www.hfma.org/certification/chfp/
0.025 | clueweb09-en0007-02-21697|http://www.hfma.org/events/
0.009 | clueweb09-en0007-02-21710|http://www.hfma.org/forums/
0.085 | clueweb09-en0007-02-21713|http://www.hfma.org/hfm/
0.021 | clueweb09-en0007-02-21908|http://www.hfma.org/jobs/
0.184| clueweb09-en0007-02-21912|http://www.hfma.org/library/
0.015 | clueweb09-en0007-02-21917|http://www.hfma.org/library/management//
0.070 | clueweb09-en0007-02-21925|http://www.hfma.org/library/revenue/
0.028 | clueweb09-en0007-02-21940|http://www.hfma.org/publications//

As shown above, the sub-tree '*library*' is hosting three documents, which were also combined together in the sub-tree 'library'.

0.548 | clueweb09-en0007-02-21681|http://www.hfma.org

0.013| clueweb09-en0007-02-21682|http://www.hfma.org/about

0.032| clueweb09-en0007-02-21687|http://www.hfma.org/certification/

0.028 | clueweb09-en0007-02-21690|http://www.hfma.org/certification/chfp/

0.025 | clueweb09-en0007-02-21697|http://www.hfma.org/events/

0.009 | clueweb09-en0007-02-21710|http://www.hfma.org/forums/

0.085 | clueweb09-en0007-02-21713|http://www.hfma.org/hfm/

0.021 | clueweb09-en0007-02-21908|http://www.hfma.org/jobs/

0.269| clueweb09-en0007-02-21912|http://www.hfma.org/library/

0.028 | clueweb09-en0007-02-21940|http://www.hfma.org/publications//

In special cases, some leaves are connected directly to the root, without nested sub-trees (parents). This occurs when the parent is not relevant to the topic (i.e., it makes minimal contribution, or the document was not crawled so it is not available in our dataset), and the leaves have more topical contributions. As shown below, the home 'http://www.authorama.com/' is not available, and thus is not relevant.

clueweb09-en0005-29-01100|http://www.authorama.com/secret-garden-1.html|0.1690308

clueweb09-en0005-29-01101|http://www.authorama.com/secret-garden-10.html|0.4285714

clueweb09-en0005-29-01102|http://www.authorama.com/secret-garden-11.html|0.13484

clueweb09-en0005-29-01103|http://www.authorama.com/secret-garden-12.html|0.145865

clueweb09-en0005-29-01104|http://www.authorama.com/secret-garden-13.html|0.1524985

clueweb09-en0005-29-01105|http://www.authorama.com/secret-garden-14.html|0.1428571

clueweb09-en0005-29-01106|http://www.authorama.com/secret-garden-15.html|0.145865

clueweb09-en0005-29-01107|http://www.authorama.com/secret-garden-16.html|0.1324532

clueweb09-en0005-29-01108|http://www.authorama.com/secret-garden-2.html|0.140028

clueweb09-en0005-29-01109|http://www.authorama.com/secret-garden-7.html|0.2153874

To process this case, the leaf with the highest weight (the second level above) was selected to represent all documents in the class. The weight of the representative document is the total weight of all the documents, as shown below:

clueweb09-en0005-29-01101|http://www.authorama.com/secret-garden-10.html|2.3856

To summarize, the following two steps are used to determine rank r_i of a given document *i*:

1. For each terminal node *i* in the hierarchy of URLs, the frequency of the terms in the text is determined, and the topic-weight w_i for this node is equal to that frequency.
2. The frequencies are propagated recursively by the links in the hierarchy to the node with the highest weight:

$$r_i = r_i + \sum_j w_j^i \quad (10)$$

At this stage, the spam documents are recognized by checking the distribution of keywords in the sites. We found that the spam filtering technique used by Waterloo University (Cormack et al., 2010) is not accurate, for the following reasons:

1- Some pages that were ranked highly relevant by our model were ranked as spam by the Waterloo method. For example, 'http://hubpages.com/hub/403b-vs-401k' and 'http://www.money-zine.com/ financial-planning/retirement/roth-403b-plans/', which refer to sites ranked 'high popularity' according to 'ALEXA' for site popularity, were not flagged as spam in our method, but were with the Waterloo model.

2- If some documents are categorized as spam, all the documents that belong to that site must also be categorized as spam; thus the entire site should be marked as spam. We used our previous model to filter out documents that come from spam sites, using Alexa.net for site popularity.

As explained in this section, we present the pseudo code for indexing the documents using five index classes, as follows.

Initialize: Indexing

Wikipedia

Loop for each document 'D' in Wikipedia

L =title of document ‘D’

Accumulates L in DIC

Computes CRC for the first paragraph→ ‘C’

Collect the significant tags→’G’

If ‘C’ in CRC class index OR ‘G’ in Common-Tag class

Adds only doc id in CRC

Else Accumulate term impact in document ‘D’ using formula 4→T

If ‘T’ greater than threshold selects the best three terms→ ‘t’

Loop for each term ‘t’ in ‘T’

Builds index node ‘N’ in term-impact class for term ‘t’ and builds a vector impact of ‘t’, other terms impact, doc id, title

End of loop on each term

Adds doc id and ‘C’ in CRC class

Adds doc id and ‘G’ in Common-Tag class

Home page Parses document ‘D’ for home page ‘H’

if ‘H’ exist creates node ‘L’ in home page class and stores URL and doc ID

End of loop for each document ‘D’

Home page

Loop for each document ‘D’ in collection

Tokenizes URL to terms ‘T’

Extracts main domain ‘t’ from ‘T’

Accumulates ‘t’ in DIC

Creates node ‘N’ in Home Page class for ‘t’ in ‘T’ and build a vector {URL, Title, TREC-id}

Document’s Title Class

Segments title of ‘D’ by punctuations and conjunctions→S

Loop for each segment in ‘S ‘

Computes the impact ‘I’ using formula 4→V

If ‘I’ greater than threshold creates node ‘N’ in Document’s Title Class

Stores V, document id (TREC id) and URL

End of loop on each segment in title

Term Combination Class

Combines URL+TITLE→V

Removes duplication in V

Loop for each term ‘t’ in V

Computes impact ‘I’ for term ‘t’ in V

Accumulates ‘I’ in document ‘D’→SUM

If SUM greater than threshold builds index node ’N’ for the best three terms ‘t’ in V and builds a vector for term impact, doc id, url

Combines DIC with ‘Million Query Track’ log file in one list→’DIC’

Topical Keywords Class

Loop for all keywords ’T’ (term or phrase) in ‘DIC’

if ‘T’ in document ‘D’ then computes impact and assigns document ‘D’ in dictionary ‘R’ where Key=’T’ and Content= D[URL]+Impact

End of loop on all keywords

End of loop on all documents ‘D’ in collection

Loop for each Key ‘T’ in dictionary ‘R’

Sort dictionary content ‘R’ alphabetically

Loop for each D[URL] in dictionary content ‘R’

If D[URL] in D[URL+1] then

Impact(D[URL],T)=Impact(D[URL], T)+Impact(D[URL+1],T)

Remove D[URL+1]

Else Remove D[URL]

End of loop on each D(URL)

End of loop on each Key ‘T’

5.3 Query Processing and Document Ranking

Query processing is an essential application of a search engine (Wang et al., 2004). It includes detecting the type of query, query searching, query normalization and query expansion. Our approach first processes all the queries as one term. If the query length is more than one term, it removes the spaces between the terms and combines them into one term (with or without hyphens to replace the spaces). The query is then forwarded to the domain-name index class. If

the query matches any node in the index class, the home page is extracted and forwarded to the topical keyword-class index, in order to find the node of the extracted domain and the inherent documents that have good impacts on the terms. If the query does not match the nodes in the domain-name class, the original query is forwarded to the Wikipedia index class; and if the query matches any node in that class the home page is extracted and the domain name is forwarded to the Domain-Name index class to retrieve the relevant pages. Wikipedia articles often do not contain home pages, in which case the external pages in the Wikipedia class are flagged to be used for result enhancement later. Finally, if the query does not match any node in the Wikipedia index class, it is forwarded to the Term-Combination index class, and then to the Document-Title index class.

Some queries match only one index class, and others match more than one. The flowchart in Figure 5.7 shows the query processing stages (this algorithm is depicted in the flowchart rather than pseudo-code[48]).

We present example queries from the TREC Web Track 2012, and their corresponding index classes and subclasses types:

- The Domain-Name subclass: ‘arkansas’, ‘quitsmoking’, ‘newyork-hotels’.
- The Wikipedia-External-Links index subclass: ‘churchhill downs’, ‘indiana state fairgrounds’.
- The Document-Title index class: ‘becoming a paralegal’.
- The Terms-Combination index class: ‘gs pay rate’ in ‘www.gspay.com’ or ‘brooks brothers clearance’ that refers to the site ‘brooksbrothers.com’, where the terms ‘rate’ and ‘clearance’ are extracted from their contents respectively.

The Topical-Keywords index class: ‘black history’, ‘septic system design’, ‘dogs clean up bags’, ‘furniture for small spaces’

[48] The ranking value for each document is based on the priority of finding the document in the class.

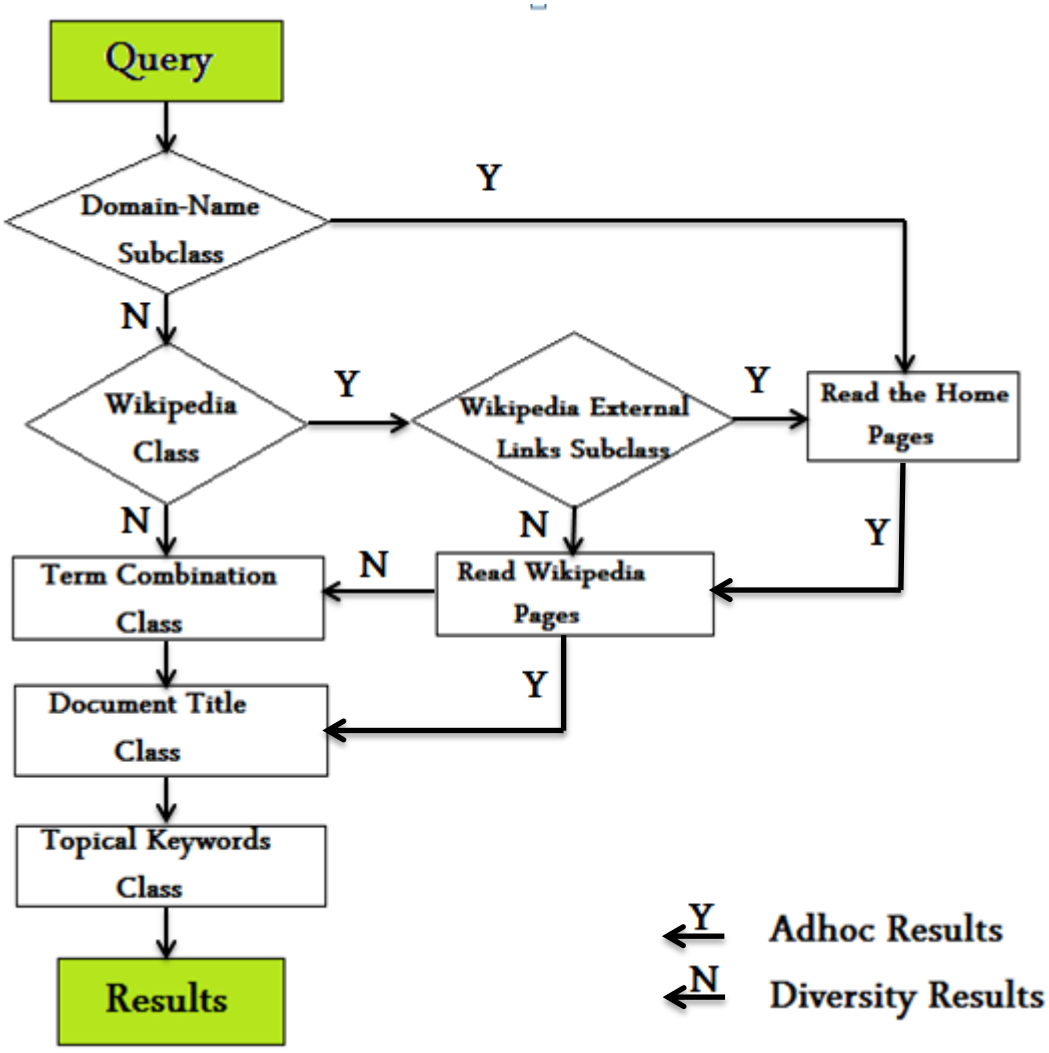


Figure 5.7 Query Processing Flowchart

Typically, each search engine has certain criteria for manipulating search results. There are two types of search preferences: user dependent and user independent. User dependent is when users add their own preferences to query results, and user independent means that search engines use their preferences to select a site over others. In our model, we redistributed the documents in the final list, to compromise for the preferences that satisfy user needs for both tasks (adhoc and diversity):

- Homepages (e.g., '.com', '.gov', '.org', '.edu').
- Wikipedia results with titles that match the query literally.

- Site preference, if it is one of the top 20 in the ranked list. We used the ‘about.com’ and ‘answers.com’ sites as user preferences, because relevant judgment considers them valuable informative sites in the diversity task.
- Top ten results that ranked high, regardless of the site type.
- Other Wikipedia results that ranked high based on their contents.

The example below shows a sample of the ordered results for the query ‘University of Phoenix’, selected from our results at the TREC 2012 web track:

```
0300.500|clueweb09-en0010-96-37426|http://www.phoenix.edu/
0300.500|clueweb09-en0002-37-25523|http://phoenix.edu/
0300.500|clueweb09-en0000-60-13706|http://axia.phoenix.edu/
0200.000|clueweb09-enwp02-07-02081|http://en.wikipedia.org/wiki/en:University_of_Phoenix
0200.000|clueweb09-enwp01-79-14175|http://en.wikipedia.org/wiki/University_of_Phoenix
0115.167|clueweb09-en0003-27-19431|http://phoenix.about.com/library/blseatingcardinals.htm
0072.193|clueweb09-en0004-83-07994|http://wiki.answers.com/q/what_are_some_....._the_university_of_phoenix
0012.920|clueweb09-en0000-49-20635|http://business.phoenix.edu/
0007.792|clueweb09-en0005-12-33913|http://www.classesandcareers.com/uni.....ity-of-phoenix-cleardegrees/
0007.672|clueweb09-en0007-27-03846|http://www.earnmydegree.com/online-.....llege/university-of-phoenix.html
0007.253|clueweb09-en0001-50-14713|http://education.stateuniversity.com/pages/2520/university-chicago.html
0007.037|clueweb09-en0003-21-24814|http://technology.phoenix.edu/
0005.500|clueweb09-en0002-35-32718|http://military.phoenix.edu/
0004.826|clueweb09-en0000-03-11218|http://artsandsciences.phoenix.edu/
0004.126|clueweb09-en0001-68-08416|http://education.phoenix.edu/
0003.671|clueweb09-en0008-22-23145|http://www.healthprofessions.com/
```

Search engines use query expansion to optimize the quality of search results. It is assumed that users do not always using the best terms when formulating search queries (Imran et al., 2009 and Billerbeck et al., 2004). User feedback is important for query expansion, and most search engines use this technique to expand the initial results and satisfy user needs. Studies show that observable behavior explicit feedback used by traditional search engines (e.g., the links a user clicks on it in the ranked results, the time a user spends reading a page, how a user reformulates a query), provide weak and noisy feedback information about which pages the user preferred in the returned list. Radlinski et al. (2008) proposed that none of the absolute usage metrics that were explored (e.g., number of clicks, frequency of query reformulations, abandonment) reliably reflected retrieval quality. Our approach uses implicit user’s feedback; that is, it computes the

behavior of some Wikipedia writers who adopted the preferences in the dynamic properties of the Wikipedia collection. Our approach uses two algorithms for query expansion: shared-links, and the manner of titling similar articles.

5.3.1 Query Expansion Using Shared-Links

We explained how Wikipedia articles are connected together to form an online resource. Wikipedia writers join similar or related articles using links. Similarly, Wikipedia articles can expand current articles to other articles by using shared links, known as in-links. Usually, the target articles also point back to the source articles.

If we assume an article (A) in Wikipedia has a link that points to article (B), and article (B) has a link that points back to article (A), then articles A and B are topically related. We indexed all incoming and outgoing links for each article by building a custom hash-table on the memory. The article names represent the keys of the table, and the outgoing links were stored in the contents of the corresponding keys. The index was then mapped to the physical disk.

Figure 5.8 shows an example of expanding the query 'Global Warming' by the terms 'Carbon Dioxide', 'Air Pollution', 'Greenhouse Gas' and 'Alternative Fuel'.

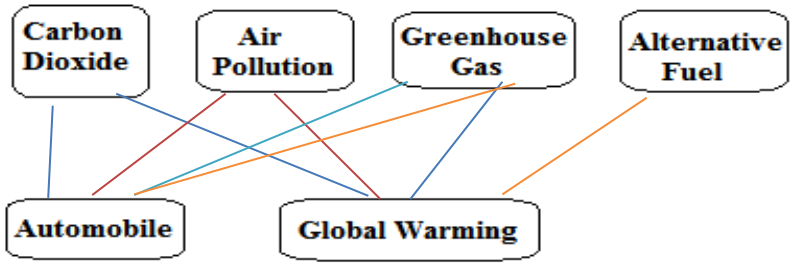


Figure 5.8 Global Warming and related articles

5.3.2 Query Expansion Using Titling Variation Aspect

Wikipedia writers often use different strategies to title articles, and highly similar articles can have different titles. We used these variations in our model to expand queries, which means if the initial ranked list contains results from Wikipedia, the title of the article will be used to collect the other corresponding titles. For example, an article 'Lipoma' is titled by Wikipedia writers as: 'Fatty Tumor', 'Fatty Lipoma', 'Lypoma', 'Lipomatous Neoplasm', 'Lipomas' and 'Lipomatosis', because the content of these articles is similar. These variations were collected by

our system during the indexing of the Wikipedia articles (see Section 5.2). Each title in Wikipedia was a key in the dictionary, while the expansion phrases were stored in the content of that key.

Hence, if a user queried the system and the system retrieved a Wikipedia document relevant to the query, then the title of the document will be used to access the key of the dictionary, in order to retrieve its content.

However, query expansion and reformulation in our model was not used for all queries; only when the initial ranking list was short (i.e., less than 200 pages), and the initial query retrieves at least one Wikipedia document from the index. For example, the initial result list for the query 'angular chelitis' was too short, and the expansion to 'angular stomatitis' increased the number of results retrieved by our model.

5.4 Evaluation

In 2012, TREC Web track organizers and assessors tested our final model for the two tasks of adhoc and diversity (Clarke et al., 2012). Again, for some queries our precision was zero, because many documents retrieved for these queries were ranked as spam in the TREC relevance judgments. Specifically, the relevance judgments automatically filtered out spam documents using the Uwaterloo IR system (Cormack et al., 2010). We believe that some of the documents that were ranked as spam in the TREC collection were not actually spam, since they were ranked as having high popularity (good reputation) according to ALEXA[49]. Some other relevant documents were missed by our system because they were from the set A of the ClueWeb09 collection that we did not use.

We present our results compared to all 28 models for the adhoc task and 20 models for the diversity task used in the test queries in TREC 2012 (the 50 testing queries are shown in the Appendix II). Tables 5.2 and 5.3 list the results of different metrics for both the diversity and adhoc metrics averaged over the 50 test queries.

[49]http://www.alexa.com/

Group	ERR@20	nDCG@20	P@20	MAP
Univ. of Ottawa	0.299	0.214	0.405	0.120
Name not disclosed	0.290	0.167	0.305	0.117
Univ. of Twente	0.219	0.113	0.221	0.061

Table 5.2 Comparison of the ad-hoc retrieval task results for the testing queries 2012 for two top systems from TREC 2012 on subset (B), and for our system submitted to TREC 2012.

Group	ERR-IA@20	α-nDCG@20	NRBP
Univ. of Ottawa	0.431	0.525	0.394
Univ. of Twente	0.405	0.508	0.357
Univ. of Delaware (Fang)	0.300	0.420	0.241

Table 5.3 Comparison of the diversity retrieval task results for the testing queries 2012 for two top systems from TREC 2012 on subset (B), and for our system submitted to TREC 2012.

Table 5.4 and 5.5 show our positions in the top eight models in TREC 2012 ordered by ERR score, regardless of which collection was used, A or B. We were in third position for the adhoc task and second for the diversity task; but if only the systems that used subset B are considered, we achieved the best results for both tasks. Methodology, the criterions nDCG and α-nDCG are similar; but they named in each task differently. Table 5.6 and 5.7 show the comparison between our model and all other models for two tasks, Figure 5.9 shows our results for each topic in the diversity task, and figure 5.10 shows other metrics for the adhoc and diversity tasks.

Group	Run	Cat	Type	ERR@20	nDCG@20	P@20	MAP
uogTr	uogTrA44xi	A	auto	0.313	0.238	0.453	0.212
srchvrs	srchvrs12c09	A	auto	0.305	0.176	0.315	0.126
uottawa	DFalah121A	B	auto	0.299	0.214	0.405	0.120
QUT_Para	QUTparaBline	B	auto	0.290	0.167	0.305	0.117
utwente	utw2012fc1	B	auto	0.219	0.113	0.221	0.061
ICTNET	ICTNET12ADR2	A	auto	0.215	0.110	0.257	0.078
IRRA	irra12c	B	auto	0.173	0.143	0.367	0.153
qutir12	qutwb	B	auto	0.166	0.146	0.308	0.131

Table 5.4 Top adhoc task results ordered by ERR@20 for the best model over 48 models

Group	Run	Cat	Type	ERR-IA@20	α-nDCG@20	NRBP
uogTr	uogTrA44xu	A	auto	0.505	0.606	0.463
uottawa	DFalah121D	B	auto	0.431	0.525	0.394
utwente	utw2012c1	B	auto	0.405	0.508	0.357
srchvrs	srchvrs12c00	A	auto	0.386	0.485	0.340
ICTNET	ICTNET12DVR1	A	auto	0.326	0.422	0.280
udel	autoSTA	A	auto	0.325	0.419	0.282
LIA	lcm4res	A	auto	0.318	0.424	0.268
udel_fang	UDInfoDivSt	B	auto	0.300	0.420	0.241

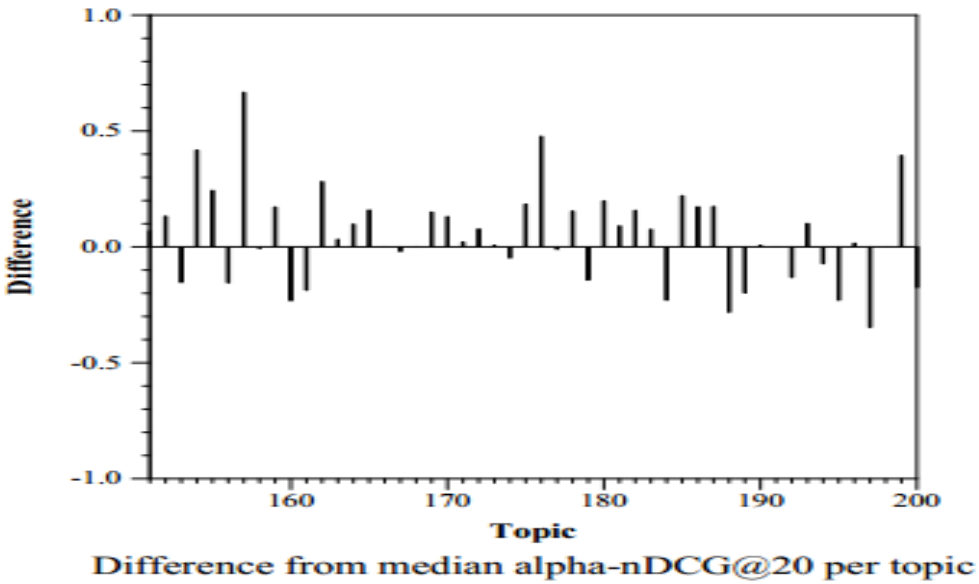
Table 5.5 Top diversity task results ordered by ERR-IA@20 for the best of 48 models

Model	**ERR@20**	**nDCG@20**
University of Ottawa	0.299	0.214
Average over all models	0.213	0.102

Table 5.6 Average mean results (28 models/50 test queries) and our submission for adhoc task

Model	**ERR-IA@20**	**α-nDCG@20**
University of Ottawa	0.431	0.525
Average over all models	0.143	0.437

Table 5.7 Average mean results (20 models/50 test queries) and our submission for diversity task



Difference from median alpha-nDCG@20 per topic

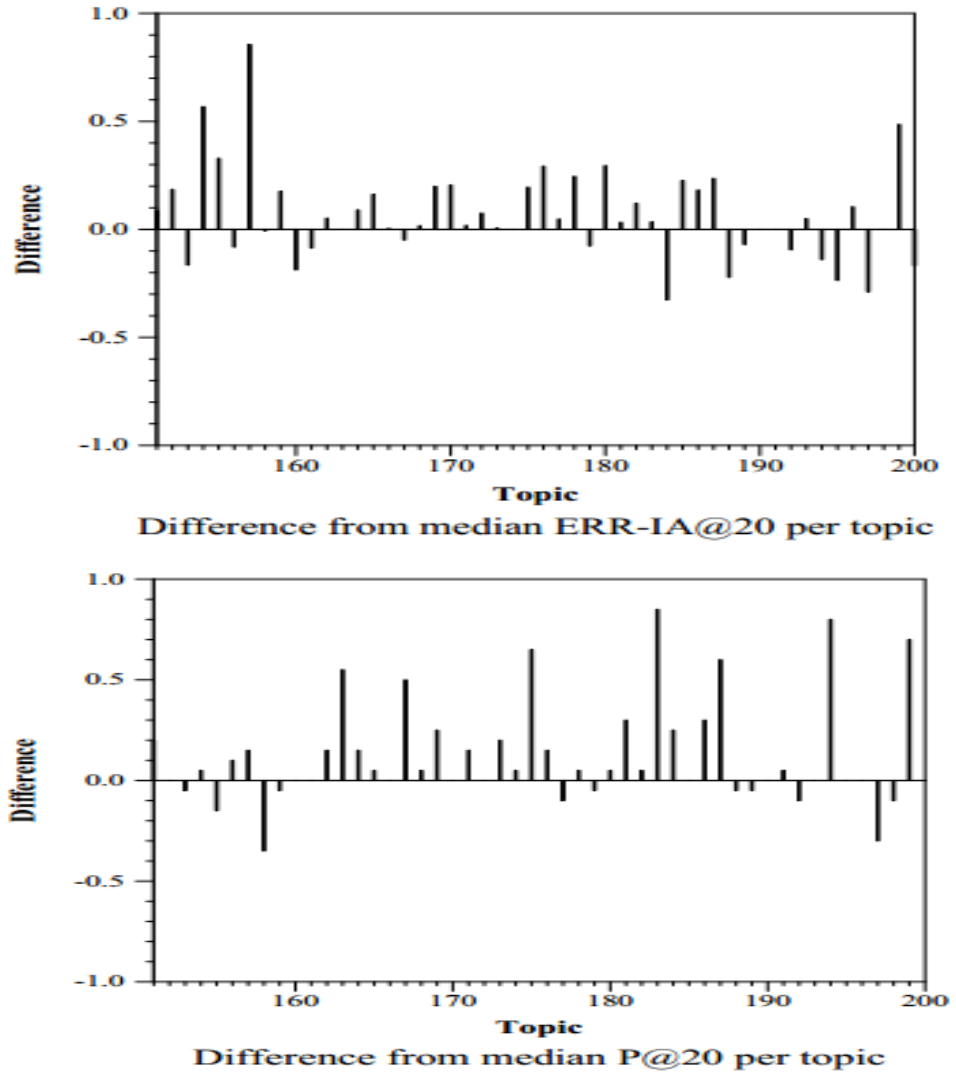


Figure 5.9 Our Web Track 2012 diversity task results for each query

Summary Statistics	
Run ID:	DFalah121D
Task :	diversity
Run type :	automatic
Document collection category:	B
External resources used:	no additional resources
Number of topics:	50

Adhoc measures		Diversity measures	
Retrieved	13109	α-nDCG@10	0.4910
Relevant	3523	α-nDCG@20	0.5253
Relevant retrieved	802	ERR-IA@10	0.4222
Prec@10	0.4420	P-IA@10	0.3414
Prec@20	0.4050	P-IA@20	0.3177
MAP	0.1203	MAP-IA	0.1052
NDCG@20	0.2135	NRBP	0.3940
ERR@20	0.2992	ERR-IA@20	0.4315

Figure 5.10 Web Track 2012 adhoc and diversity results

All the models submitted for the adhoc and diversity tasks were evaluated according to the judging criteria of both tasks, including approaches that were not submitted for both tasks. This method of judging allows TREC to make direct comparisons between models optimized for the two tasks, which helps determine if various judging criteria and evaluation measures identify genuine differences. For example, Figure 5.10 presents a scatter plot comparing the performance of the models under ERR@20 and ERR-IA@20, the primary effectiveness measures for the adhoc and diversity tasks, respectively. Though the values are correlated, there are clear differences in the relative performance of the approaches under the two measures. Table 5.8 shows the adhoc results (nDCG and ERR) for each query (the query string for each topic number is stated in the Appendix II).

Model ID	**Topics**	**nDCG@20**	**ERR@20**
DFalah120D	151	0.22955	0.39361
DFalah120D	152	0.00000	0.00000
DFalah120D	153	0.08275	0.10021
DFalah120D	154	0.10298	0.28497
DFalah120D	155	0.21067	0.41412
DFalah120D	156	0.08611	0.11084
DFalah120D	157	0.05937	0.06146
DFalah120D	158	0.32675	0.33867
DFalah120D	159	0.51948	0.96808
DFalah120D	160	0.00000	0.00000
DFalah120D	161	0.00000	0.00000
DFalah120D	162	0.16466	0.09836
DFalah120D	163	0.63615	0.19782
DFalah120D	164	0.34942	0.94775
DFalah120D	165	0.02823	0.04861
DFalah120D	166	0.00000	0.00000
DFalah120D	167	0.09613	0.09279
DFalah120D	168	0.52165	0.96808
DFalah120D	169	0.54711	0.24389
DFalah120D	170	0.07545	0.10471
DFalah120D	171	0.28692	0.38342
DFalah120D	172	0.06756	0.09157
DFalah120D	173	0.24769	0.20116
DFalah120D	174	0.45431	0.96808

DFalah120D	175	0.24657	0.38656
DFalah120D	176	0.03225	0.09302
DFalah120D	177	0.22445	0.05043
DFalah120D	178	0.20987	0.16400
DFalah120D	179	0.13742	0.08799
DFalah120D	180	0.04214	0.09410
DFalah120D	181	0.00000	0.00000
DFalah120D	182	0.53050	0.59760
DFalah120D	183	0.11062	0.12120
DFalah120D	184	0.49436	0.24674
DFalah120D	185	0.05297	0.08456
DFalah120D	186	0.65770	0.37750
DFalah120D	187	0.12228	0.15549
DFalah120D	188	0.00000	0.00000
DFalah120D	189	0.00000	0.00000
DFalah120D	190	0.00401	0.00417
DFalah120D	191	0.27695	0.94757
DFalah120D	192	0.19223	0.26049
DFalah120D	193	0.13347	0.09869
DFalah120D	194	0.16381	0.12120
DFalah120D	195	0.00000	0.00000
DFalah120D	196	0.74016	0.48905
DFalah120D	197	0.01717	0.03867
DFalah120D	198	0.14005	0.13648
DFalah120D	199	0.27827	0.38656
DFalah120D	200	0.21007	0.15089
DFalah120D	**Average of all topics**	**0.2022**	**0.24222**

Table 5.8 Our system adhoc task results for each query

As mentioned, this model used five classes that work cooperatively for each topic. We wanted to do more analysis regarding the contribution of each index class. We ran additional experiments that used one class at a time. Activating one class and deactivating other classes will clearly show the contribution of the used class. We combined the Wikipedia and Home-page classes together for the following reasons: (1) these classes are interconnected and they work

cooperatively; therefore deactivating Wikipedia and activating Home-page class will cause some home-pages to not be retrieved, (2) for some queries, both the Home-page and Wikipedia classes return results are less than 10, if they are used separately; as a consequence, the trec-eval tool for evaluation results is not working. However, using other classes without Wikipedia will affect their results because the query expansion process will be eliminated, too. The home pages indexed from Wikipedia are far more relevant than those from other resources that might also have contained some homepages. With respect to adhoc and diversity tasks, the charts show the significant changes when we use one individual class at a time; the Wikipedia and the Home-page classes bring the highest contribution; for the following reasons: (1) they are relevant for most query types and for both tasks (adhoc and diversity); therefore most query results are retrieved from those classes, (2) query expansion in our approach is based on Wikipedia; therefore eliminating the Wikipedia class will cause some documents to not be retrieved. This situation is proved when we use the other classes individually. The Document-Title class has the least contribution, because the Topical-Keywords and Terms-Combination classes are using document's title, internally. However, using all classes does not mean a simple combination of individual classes' results, because some results are shared in more than one class (without duplicating any information); hence, using all classes means combining all results in one list without keeping duplications. Figures 5.11 and 5.12 illustrate the basic comparison, and the impact of each index class on the retrieval performance.

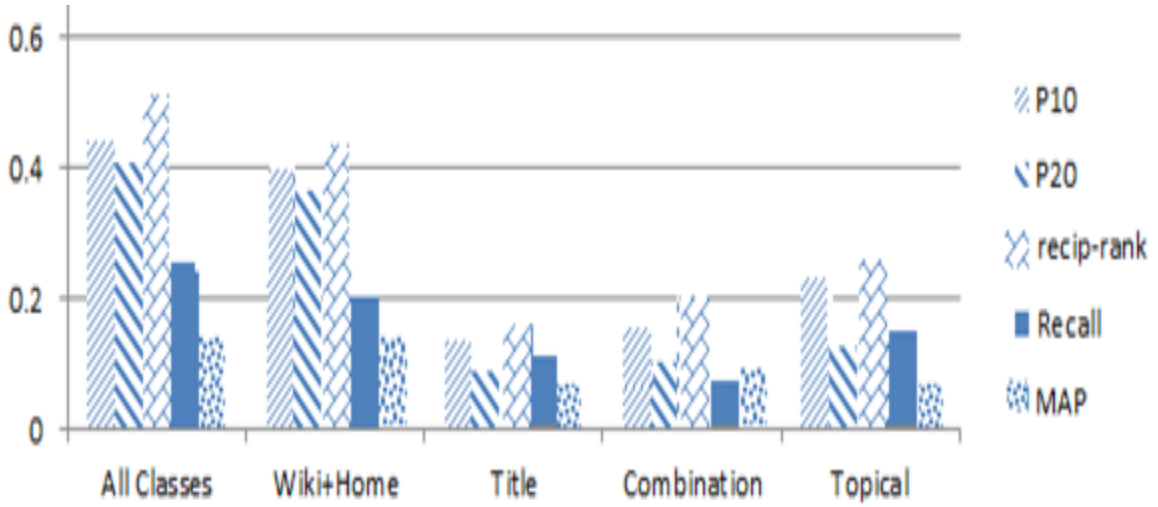


Figure 5.11 Impact of each index class based on the adhoc metrics

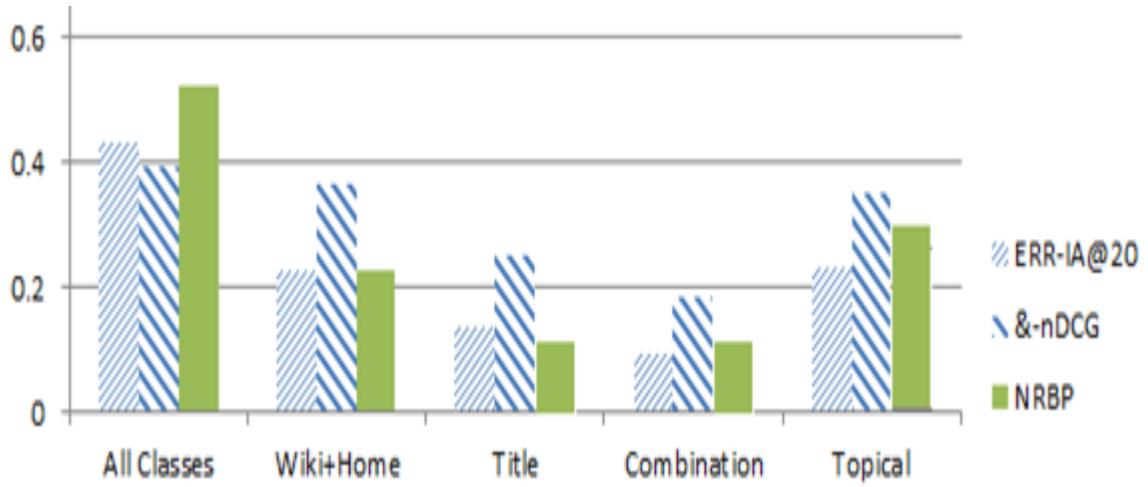


Figure 5.12 Impact of each index class based on the diversity metrics

5.5 Performance Evaluation

As discussed, our approach was not based on a currently available framework; instead, we built an entirely new model that is robust, and applicable for different kinds of queries and both types of tasks. Throughout the development, we monitored the progress using various performance metrics, some of which were discussed previously. Other metrics that are related to the speed and complexity factors are examined in this section.

We have run our system on many queries, including the ones from the TREC log file[50]. Our system tested multiple implementations and is currently using different types of data, including offline data provided by TREC, and online data indexed in real time and adapted from social sites such as Twitter and Facebook. We believe that search engines need to work with both types of indexes, as some users are looking for archived data and others want online data, such as events or news. Overall, this combination of indexes gives our approach high scalability, and it can easily model user needs. Multiple indexes allow our approach to be more flexible and reliable when responding to the different query types. In terms of efficiency, figures 5.13 and 5.14 show the latest results we have achieved with the current approach, with respect to the indexing time and query response time. The size of our index is 80 Gigabyte for as compared to the original collection size of 1 Terabyte. As shown in the figure 5.13, the indexing time complexity increases during processing, because the index is initially empty and the indexer will only access (write) the disk once, at the end. The writing also depends on the size of the

[50] http://trec.nist.gov/data/million.query.html

documents that have been indexed. The process takes somewhat longer during indexing, since the indexer must access the disk twice (read and write/replace), and the operation is appended (i.e., read the old node, append the new data and replace the old node). Informally, this means that for large enough input sizes, the space complexity increases linearly, *O(n)*, with the size of the input. Whilst the time complexity for indexing, searching, and insertion is a tight bound *O(log(n))*.

Our approach can index a corpus of over 50 million documents, but this is dependent on the available hardware. For example, we used a 3.00 GHz Quad Core CPU with 16.0 GB Memory to build the index. Likewise, query response time is variable and decreases from query to query, because the response time depends on the size of the index node accessed for reading, and on the type of query (diverse queries require a little more time than adhoc). A simple query (one word) increases the diversity of the searched topics and has high complexity; likewise, a longer query may increase the complexity if the query terms have different distributions in the documents. To reduce the complexity of search in long queries, most search engines, similarly to our approach, use Boolean searching to reduce the probability of term distribution on the document content. Finding consecutive and proximity terms for a query in the document is a challenging task. Implementation of Boolean search may have many positive features that overcome the shortcoming for the long queries. Given the manner in which most people search and based on the results of most popular search engines, a reasonable course of action can conjecture that the ranking algorithms of search engines adhere to the following rule: place at the top of the results list those documents that contain all the query terms and that have all the query terms near each other. With such a ranking algorithm, the use of advanced query syntax will have little impact on the results at the top of the list. Many other methods are used to reduce the complexity for long queries. The user interface, for instance, is important to tackle this problem. A sophisticated user interface with advanced Boolean operators, for example, may be discouraging for users. Query expansion is also worth using for reducing this complexity, formulating the user query in a suitable manner and using resources for some queries, as we did by using Wikipedia. These approaches can help to provide the relevant results regardless the degree of complexity. In short, the worst-case performance to find a query in the index is *O(N)* (linear time); that is, proportional to the length of the index being searched; and generally, the search query is sub-linear; not every

entry in the index needs be checked. Table 5.9 shows the relation between the number of results and the length of the query in our approach.

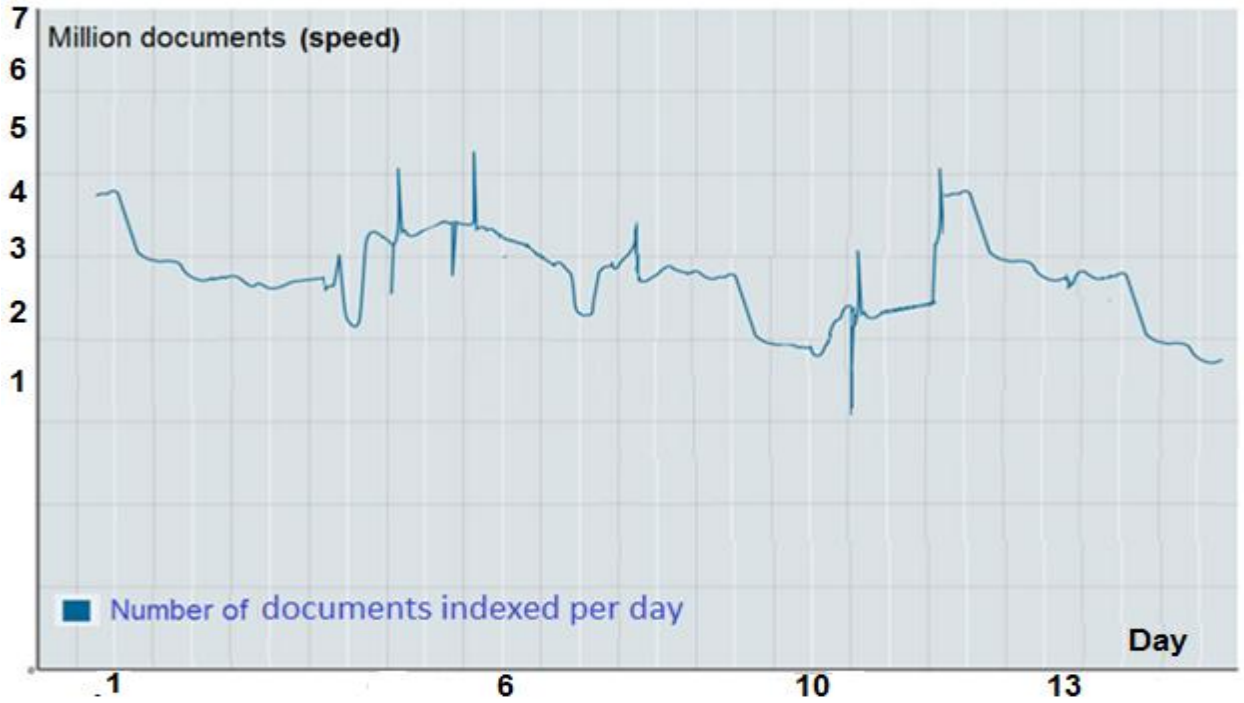


Figure 5.13: Full document content indexing over time

Figure 5.14: Query processing response time

Query Length	Average number of Results
1	>200
2	>100
3	>75
4	>50
5	<50
6	<20
7	<10

Table 5.9 Query length vs. number of results

5.6 Discussion

Web and IR systems currently model a user's information needs via the query. Most Web and traditional search IR engines follow a statistical query term and document term comparison. The premise of this analysis is if one can correctly model the query, it would be a major step to correctly modeling a user's information need. While current WIR systems have modeled document structure aspects, they focused on document characteristics, not on the query structures. In future work, more investigation could try to address questions such as: Is there a linguistic component to WIR research? Is there a linguistic identification for query structure in WIR? There appears to be a basic syntactic structure to queries, and user modeling should take the syntax and semantic of the query into account.

Our model combines content-based web indexing and structure-based queries, similar to how an SQL database functions. The queries combine conditions on text patterns in document content, with meta-data patterns described in URLs and titles.

We proposed to model the web as a relational database composed of two relations: Document and Query. The Document relation has three tuples for each document in the web, represented by: URL, titles and document content, while the Query relation has five tuples for each Query in each document in the web, represented by: URL, title, content, URL-title and URL-title-content. This relational abstraction of the web allows us to use a query language similar to SQL to pose the queries.

The score-based query structure is a useful framework for analyzing document retrieval, with respect to document structure. We investigated the composition of different implementations of

structured retrieval aspects, including computation, combination and propagation of query term scores for complex structured query formulations. Furthermore, we illustrated how state-of-the-art retrieval models can benefit from the formulation of structured queries with such a flexible and structured retrieval framework.

We evaluated our structured-query based approach on TREC testing queries in 2012, using the same test collection we used with our previous models. The effectiveness improves when using structured queries that utilize document structure. Furthermore, the selectiveness of structured queries is comparable or better than using all the document structures in the index classes.

In this chapter, we extended our custom approach of Chapter 4, to close the gaps and address the pitfalls that degraded our results in the chapter 4. We exploited the structure of the queries, by using the offline Wikipedia collection included in the ClueWeb09 data, in order to find related entities in the larger Web. It became evident that using category information to retrieve entities in Wikipedia is a significant improvement over the baseline full-text search in Wikipedia; though it should be noted that the performance is weak. Locating the web homepages of entities found in Wikipedia by following the external links on the Wikipedia page is better than searching an anchor text index represented in URLs. In fact, we heavily used the Wikipedia crawled corpus and the information was driven from data which was not vulnerable in the case of a Wikipedia shutdown. Our approach used Wikipedia as an offline collection of references and it could be improved by exploiting the answer entities. Query structure is very important when it is applied in document structures. As seen in the results for each query, our approach captured the topics of all the queries (the queries that assigned '0' were not ranked by the evaluators); our previous approaches failed to do this. Our final approach is an extension of the previous models. We used the same features as the second model and other relevant features from Wikipedia knowledge, and thereby improved our approach for some topics.

In summary, we focused on enhancing our model's performance to retrieve relevant results. Typically, the evaluations above do not include results for all the submitted models. For example, if a researcher submitted several models (different models or one model with different parameter settings), the assessors chose the researcher's best model. This indicates that the researcher was unable to decide which model is better for a particular set of queries. In addition, the models that ranked on top for a specific query set would not necessarily be on top for other

query sets. For example, in Tables 5.4 and 5.5, the top approach is the University of Glasgow's 'uogTr'; but, in fact, these researchers submitted six models, and all approaches except the best one had lower precision than the other models on the list. A decision is not usually made before testing a particular model, and this is a fair way to determine which model is better for a specific list of testing queries. Thus, the top models ranked for specific testing queries might or might not be among the top models if the testing queries are changed.

This approach was presented in the TREC 2012 web track (Al-akashi and Inkpen, 2012). We explained how Wikipedia content and document phrases can cooperatively support finding relevant results, with low disk overhead and increased retrieval performance. Our method is based on topic identification in documents' content through the investigation of keywords/keyphrases from meta-data with high impact values. Our model provides a variety of analytical capabilities, including phrase extraction, keyword correlation, web page topic finding, document grouping, phrase-based inverse document frequency, and document versus domain topic weighting. Compared to our work in TREC 2010 and 2011 (Al-akashi and Inkpen, 2010, 2011), this approach is a more sophisticated and robust method of processing all types of queries.

Chapter 6

Final Enhancements

6.1 Introduction

The degree of interaction between users on the Web has increased exponentially over the past decade, and they encounter many challenges when dealing with such vast amounts of data. While much of the data is private interactions and limited (e.g., emails and blogs), there has recently been a surge in social communications data (e.g., Twitter and Facebook). So far, only a few applications have attempted to improve user experiences with social networking and related data.

Facebook and Twitter have become the leaders in social networking, with over 500 million users (Zuckerberg, 2010). In social applications, users post a wealth of information on these networks, which can be used to define their online personality. A social application can learn user preferences through static information such as book and movie interests, and dynamic information such as user locations. Another significant feature is a user's social circle (e.g., the posts of friends' friends). From a search engine perspective, learning social interactions can be very helpful when personalizing results for users. However, in addition to being social communication platforms, Facebook and Twitter are enabling third party applications to track these interactions through the use of API authentication.

Facebook has taken two major steps that are impacting the field of search. First, in September 2009 they made their data available to any third party service (Zuckerberg, 2008). Secondly, as of September 2010 they started adding web links to external web resources, based on the recommendations of users (Lunt, 2004). However, despite these developments, little has been done with this newly available data.

In late 2009, Cuil launched a product called Facebook Results (Talbot, 2009), which indexed authenticated users' and their friends' wall posts, comments and interests. Cuil claimed that this data is exponentially sparse, and using the related data greatly enriched the user experience. While social data is still relatively new on the web, there have been some attempts to use it to enhance web searching. Klout, a San Francisco based startup, measures the influence of users on their circle of friends and determines their score (UserRank) and the topics they are most influential about (UserTopicRank) (Rao, 2010). In a recent development, web search engines can now use such information to determine the authority of social data. In October 2010, Bing and

Facebook announced the 'Bing Social Layer', which provides the ability to search for people on Facebook, and see the related links a user's friends had liked in Bing's search results (Nadella, 2010). As well, Twitter provides search engines with a third party application (API) that can search for users that share the same interests.

Furthermore, search engines have another challenge, which is related to how they display their search results. Researchers have noted that, although textual summaries in results are compact, concise and download quickly, they require the user to read many documents; they cannot be scanned anywhere near as fast as visual content, which allows users to get the essence of a topic within 110 milliseconds or less (the time it takes to read one or two words). Consequently, researchers have often suggested that visual content-based search results improve textual summaries better than plain text. This is a rapidly developing area, and improvements in the interfaces will very likely lead to improved search experience and more capable information creators and users. Therefore, social searching and graphical search interfaces are important future areas of advancement.

What would happen if we combine the best ideas from the Social Web and the Semantic Web? The Social Web is an ecosystem of participation, where value is created through the aggregation of many individual user contributions. The Semantic Web is an ecosystem of data, where value is created by the integration of structured data from many sources. What applications can best synthesize the strengths of both these approaches, and creates a new value paradigm that is both rich with human participation and powered by well-structured information? This proposes a class of applications called *'collective knowledge systems'*, which unlock the 'collective intelligence' of the Social Web via the knowledge representation and reasoning techniques of the Semantic Web.

This chapter briefly summarizes the state-of-the-art of development of previous models of search engines through the use of social data. We also discuss our ideas for enhancing search results for information seekers by improving search and result interface design.

6.2 Indexing Real-Time Data

The advent of services such as Twitter and Facebook has made it extremely easy for researchers to create real-time data in the form of micro-posts. Twitter users write and describe whatever

they wish in 140 characters, and whether they publicize a link to an external article they like, or just share a passing thought, the information is very valuable from a search engine real-time perspective. However, as different projects have shown over the past few years, this information is even more important if it is mined and presented to the users in real-time (Milajevs et al., 2013). Building real-time indexing first requires access to raw micro-post data. Fortunately, as a real-time data service, Twitter provides such data in periods of seven days to researchers who request it. Since this data must be processed and presented in real-time, our system avoids dealing with it all at once. There are ~90M tweets generated daily, about 1,040 per second (Rao, 2010). At 140 bytes apiece, 12.6 GB of tweet-data is created every day. Having dealt with petabytes of data, dealing with a corpus of this size is trivial for any modern search engine (Abhishek et al, 2012). Thus, our system connects to the Twitter index to retrieve data for a particular topic regarding a user's query (we use API authentication to request such data). However, are all tweets equally important? From our perspective, the important tweets on a particular topic must be rated many times between users (they are cycled within a period). More specifically, the tweets that point to external links have higher impact than others, because users are more likely to request diverse topics through web pages, as they are more natural than the short tweet texts. By combining co-occurrence and information gain with a time decay factor, it is possible to analyze tweets and derive related topics in real-time.

As with traditional indexing, achieving this involves crawling, parsing, extracting any external links if present and ignoring them otherwise, and building the index. The system processes every user contribution; it examines the text, tags, and all of the other structured data in the collection.

This stage is similar to the anchor texts in web page indexing, but instead of anchor texts we have tweet texts, and instead of PageRank we have UserRank. Though traditional search engines, such as Google and Microsoft, are aware of the importance of tweet data, they have not worked with it yet. Real-time data and related topics help users discover information about current topics better than classical web pages.

Since the data must be retrieved and indexed in real-time, and must be ranked and shown to the users from the most to least recent, our system crawls, parses and builds the index on-the-fly as a hash file. This means that each key is assigned by a link to the external page, while the content of the table holds all the tweets that point to the external page. While scanning all the

tweets in a certain period, the content of the table accumulates the tweets that correspond to each key. Then the tweets are ranked by the frequency of target Web activity represented by a particular Web page, as exemplified in Table 6.1.

Key (Web Target)	**Content (topical tweets)**	**Count**
URL_1	$Tweet_1, Tweet_2, \ldots Tweet_n$	3
URL_2	$Tweet_1, Tweet_2$	2
URL_3	$Tweet_1$	1
URL_4	$Tweet_1$	1
URL_n	$Tweet_1$	1

Table 6.1 The tweets index table

6.3 Indexing the Anchor Text

Anchor texts can significantly improve Web page retrieval (Park et al., 1992), and optimized anchor links can boost Web ranking and targeted search engine traffic. Anchor texts are typically used to indicate the subject matter of the pages linked to. This pattern of Web usage is applied in search engine algorithms to enhance the relevance of the target, or the source page. The keywords in anchor texts can enhance the relevance of the target page pertaining to the keywords used. The relevance of the page containing the anchor text is also enhanced to some degree due to relevant keywords appearing on the page, but the real gain in our framework is related to the target URL. Traditional search engines, such as Google, only pick up texts from the anchor texts of indexed pages, which indicate that Google's algorithm is configured to index anchor texts as separate anchor data, thereby making it evident. Our research suggests that the weight given to anchor text has recently been addressed in the Google algorithm. The basic technique to materialize a portion of the web is to navigate from known URLs; path regular expressions are used to describe this navigation. A path regular expression can be in the form of `d1 => d2`, which means document d1 points to d2, and d2 is stored on a different server from d1; or `d1 -> d2`, meaning that d1 points to d2 and d2 is stored on the same server as d1. For example, suppose we want to find a list of triples of the form `(d1,d2,anchor-label)`, where `d1` is a document stored on a local site, `d2` is a document stored on a remote site, and `d1` points to `d2` by a link labeled as anchor-label.

Due to the lack of hardware and storage in our study, we used the anchor-text of the entire corpus (500 million documents) for indexing and ranking this collection, and get better results than indexing the documents using full-text, as researchers mentioned (Kamps et al., 2010). This provides significant effectiveness gains in home page finding and topic distillation tasks, since using anchor scores reduces the size of a corpus without decreasing home page search performance. Scanning our corpus led to indexing approximately 3.3 billion anchor links, with an anchor size file of 10 GB. Some pages have single incoming links and point only once to another page, whereas other pages have many incoming links. Our system first filtered out all the anchors that point out to other webpages once, resulting in 2.5 billion anchors, and then counted the frequency of each anchor by creating a hash table on-the-fly. The table was then mapped to the physical disk, as shown in Table 6.2.

second mortgage loan phentermine	http://www.bookpassage.com/?id=4048	10
phentermine online free shipping	www.bookpassage.com/?id=4071	2
phentermine generics	www.bookpassage.com/?id=4072	11
phentermine mg at cms	www.bookpassage.com/?id=4076	10
buy prescription phentermine	www.bookpassage.com/?id=4092	11
pill are phentermine	www.bookpassage.com/?id=4099	10
phentermine need prescription online	www.bookpassage.com/?id=4103	8
does metabolism phentermine speed up	www.bookpassage.com/?id=4108	2

Table 6.2 Example of anchor count for number of links that point to the same page

Secondly, we addressed the issue that most URLs were anchored by different texts with different lengths, because users have different writing styles. Table 6.3 shows an example of five anchors linked to same target page: ‘www.spacedaily.com’.

Spacedaily	www.spacedaily.com	13
space daily	www.spacedaily.com	25
Space News From SpaceDaily.Com	www.spacedaily.com	5
prestige space mission	www.spacedaily.com	19
Daily Space News	www.spacedaily.com	2

Table 6.3 The URLs, anchors and their occurrences

The system combined all the anchors that point out to the same pages in one vector. The duplicated words in the combined text were removed from each vector, and the values of the anchors were summed up to represent the final value for the combined anchors, as shown below:

spacedaily space daily news from com prestige mission	**spacedaily.com**	**64**

We used our module below, in which the anchor texts were used in a similar way as the sentences in the previous section. The incoming link of anchor text L_i in document *D* was treated like a sentence in D. However, the weight given to the similarity between an anchor text and a query can be different than that between a sentence and a query, and the importance of an anchor text for the relevance of *D* can be different than that of a sentence in *D*. The contribution of anchor text L_i, whose links point to document *D*, to the relevance of *D* was computed as:

$$\beta \sum_{i=1}^{n} W(A_i, D)$$

The constant β was used as the weighting factor for the anchor-text-query similarity. The following pseudo code shows how we computed anchor-text.

```
SELECT d.URL,e.URL,a.label
FROM Document d SUCH THAT
          'www.mysite.com' ->* d,
     Document e SUCH THAT d => e,
     Anchor   a SUCH THAT a.base = d.URL
WHERE a.href = e.URL AND a.label = 'label';
```

Finally, we used block-content indexing to improve indexing performance and avoid delays, storing each group of vectors in one table, and assigning each table to a particular block number. We used the Apache Lucene high-performance full-featured text search engine library to index each table. In terms of a query search, Lucene launches the tables that are relevant to the query. Our searcher can then scan through the selected table to find the particular vector that completely matches the query. Our experiments showed that the anchor length was effective for topic distillation and home page finding tasks, and it improved the retrieval effectiveness. In terms of indexing speed, our system indexed all the available anchors (2,532,857,505) in a relatively

quick seven hours, and the size of index is 20.8 MB, as compared to the original text size of 10 GB. As shown in Figure 6.1, the speed of indexing was quickest at the beginning, because the index was initially empty and the indexer built the index quickly. The operation was slower later, as the index was aggregated into one file.

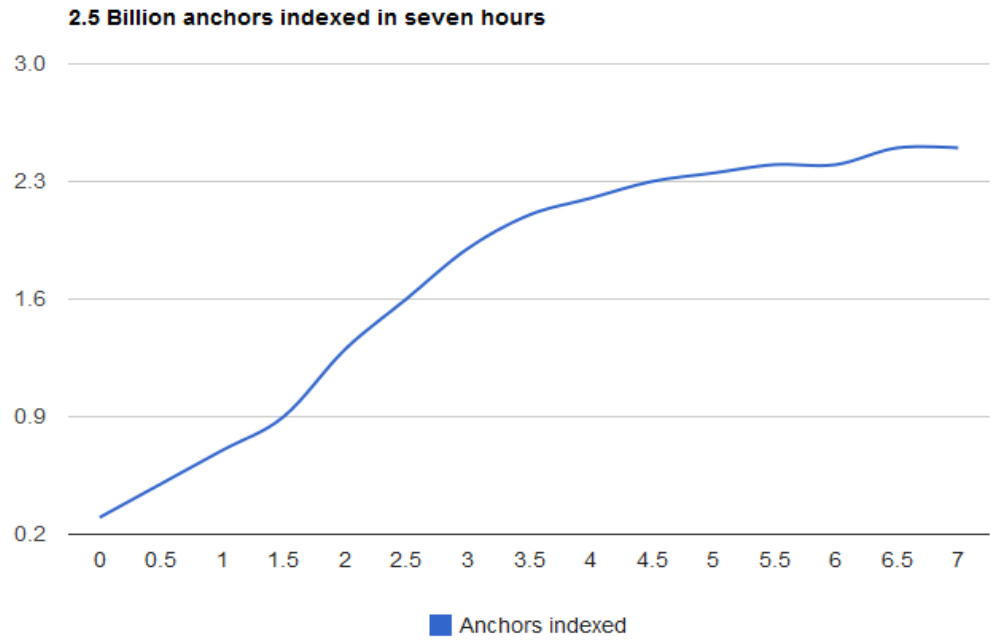


Figure 6.1 Anchor indexing period

6.4 Implementation Notes

Our framework uses phrasal indexes in a centralized server as the primary key, and we moved on from this to supporting our search engine results with real-time data, to make the approach more scalable. Building a search engine around these concerns in one system is difficult, so our system is programmatically transferred to become a client-side model. The important issue in our approach is that the framework is written in Java scripting code and the (dot) net programming language. The advantage of well-maintained, self-programmed code is more than just stability; we can also review the features and give feedback to users. Our system is comprised of two sections: the programmed server and the index files. The programmed server converts the computer into a virtual server that receives the query strings from users, processes them in the central computer, and returns the results to the users, as shown in Figure 6.2. The user first sends a query through our main page at http://eecs.uottawa.ca/~falak081, and the system compiles and

converts the query into jQuery mode, which can use the 'Schema Crawler'[51] to retrieve data from the index files and the social servers (Twitter and Facebook) as an object notation json and xml modes. The user is then prompted to receive the query results from the sub-domain http://fsrch.eecs.uottawa.ca/. All transaction requests occur in the programmed server, and the sub-domain holds the index files.

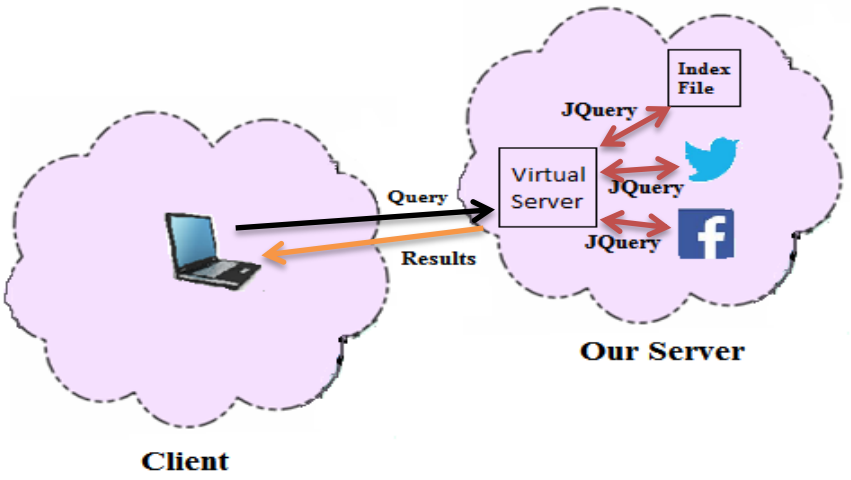


Figure 6.2 The high-level infrastructure of our search engine

6.5 User Interface Designing and Evaluation

Result interface is a process of techniques that are related to the field of Human-Computer Interaction (HCI). This field analyses how people think about and respond to user interfaces, and how best to design them to suit users' needs and inclinations. Based on years of experience, a set of practices and guidelines have been developed to facilitate the design of successful interfaces. Web-centered design concentrates on designing and building Web interfaces according to people's needs.

Visual-block based Web search results are typically derived by creating snapshots (i.e., miniaturized images) of the graphical appearance of original or semi-original documents. The layout and graphics in these visualizations can potentially provide clues about the type of document and its contents. Furthermore, if a user has seen the document or a similar one in the past, they might be able to find it quickly. Results based on visual thumbnails usually allow users

[51] Schema Crawler is our programming software that uses AJAX and jQuery to retrieve data from remote servers in JSON or xml format.

scan through the types of results faster than the snippets. Not all query results have the same presentation in the interface; that is, the design process begins by determining what the intended users' goals are, then devising an interface that can help them achieve those goals by completing a series of tasks.

However, evaluating a user interface is often a different process than evaluating a ranking algorithm or a crawling technique. A ranking algorithm can be evaluated by precision, recall and the speed of indexing and answering, and a crawler by quantitative metrics such as coverage and freshness. The quality of a user interface is determined by how people respond to it, and if they find what they are looking for quickly. The preference can be determined by many factors, including speed, aesthetics, diversification and perceived ranking accuracy. In search interfaces particularly, the results of a new visualization interface method must be perceived as being qualitatively and quantitatively better than similar results in a traditional list (Hearst M., 2009). The difficulty to evaluate might explain why the user interfaces of search results have not changed since Web search was first developed. How to best evaluate a result interface depends on the current stage of the development cycle. Discount usability methods are typically used when starting a new design or idea. One discount method involves showing a few potential users several designs, and asking them to indicate which seem most promising. Usually the design-test-redesign process cycles several times before an acceptable starting point for an interactive prototype is found. A formal experiment must be designed to take potentially confounding factors into account by balancing the order in which competing designs are shown, and those involved must avoid expressing bias toward one design over another. These studies are used to answer specific questions about how well particular design elements perform. While they can uncover important subjective results, such as whether a new design is strongly preferred over a baseline, the nature of result interfaces makes it difficult to find accurate quantitative differences with a small number of judges. Many factors contribute to this, including the strong effects of tasks or queries on system behavior. Another problem is that a search can go in many different directions, which makes it difficult to compare quantitative outcomes directly. To address some of these issues, two approaches for evaluating result interfaces have gained popularity in recent years. Evaluations are based on the objective measures recorded in the logs or on questionnaires. In some cases, the benefits of a new approach are only evident to searchers after they interact with it over time; thus, a long-term study could reveal the benefits. Sometimes, an interface that

is initially engaging due to impressive graphics could eventually become tiresome, and cause users to move back to a familiar baseline interface.

6.6 Our Search Results Presentation and User Interface

Users of Web search engines are often forced to sift through long ordered lists of snippet documents returned by the searches. The search engine community has explored document grouping as an alternative method to organize retrieval results, but grouping has yet to be deployed on most major search engines. Some search engines (e.g., Yahoo) organize their output into custom folders, based on categorized document levels. However, this does not indicate how the folders are created, or how well they correspond to users' interests (Zamir, 1999). In this section, we finalize our approach by introducing a new interface to our system that dynamically groups the retrieved results into blocks ordered by user's priorities.

Information seeking is inherently imprecise, because when users approach a search system they often have only minimal understanding of how they can achieve their goals. Thus, a user interface should help them recognize what kind of results they can retrieve to address their information needs. In the past, very little was known about what makes for an effective search result interface, but in the recent years more has been learned about which ideas work best from a usability perspective. Document presenting has long been investigated as a post retrieval document visualization technique (Zamir, 1999).

Our search results interface groups the final list of documents according to their hosts in the index classes, and the document topics. A search session begins with a user entering a query in the query box (see Figure 6.3). A noteworthy characteristic of our system is that the user does not need to input the complete required query; they can choose which query terms to use from a list of query suggestions, which is derived from the AltaVista Web search log file. After all index classes have returned their results synchronously, the main results page displays the different types of results (Figure 6.4). The documents are presented in multiple selections, and each selection is structured as a single family of documents. The groups of results are ordered with respect to the user's preferences, and the ranks are lower in the blocks toward the bottom of the page. Each block conveys the content of the hosted documents by displaying the screenshot, the titles, and the snippets of the documents, if they available. Some results, such as those from Facebook, YouTube and home pages, are opened automatically by our system, which guides

users to be in the same context without losing the results session; thus the user can navigate in these websites using the same frames. If the block is closest to the user's interests, the user can do several things. For example, if a certain title seems promising, the user can go directly to the document by clicking on the title, or exploring in the same window by using the left and right arrows. The documents in all the blocks are presented in a ranked list. Figure 6.4 shows the results interface for a query 'diana inkpen'. Each option in the interface is explained individually.

1- **Main Window:** Generally, our system distributes the results based on the user preference; that is, the results are distributed from the top left as: home pages, Facebook, twitter, Wikipedia, Twitter user references, Wikipedia user references, results from insight domain, and others based on the type of task.
2- **Home:** In a ranked list, search engines consider the first page to be highly ranked, and highly relevant to the user. As a result, beside showing the screenshot, title, and snipped text of the home page, our system tries to open the home page directly. This issue maintains the context of the results and keeps the user in the same session of result. The home page opens in the same window as the regular browser, which means users can navigate in a hosted site, and keep other result lists without changes.
3- **Contact information:** Users sometimes look for short strings of information (e.g., contact details). This window shows the indexed information regarding administrative and technical contacts for only the first page on the ranked list (i.e., the home page).
4- **Recent activities:** This window shows the real-time events and recent activity for the official website that ranked highest on the ranked list. This information is indexed from the account of the website in Facebook (if it is registered).
5- **Facebook:** This window shows the official Facebook page. The Facebook account of the member is displayed as it is on the Facebook website. Similarly, users can interact with that Facebook pages completely, without needing to visit the page on the Facebook website.
6- **Videos:** Fortunately, YouTube pages are an essential part of subset B of our collection. Whenever these pages are retrieved from the index, they group in a single window. Each video clip is wrapped by the title and other information, such as the duration of the video. At query time, the YouTube documents are accessed cooperatively using two indexes: the domain name index and the document title index. The domain name index is used to

determine the ‘youtube’ string in the retrieved links, and the document title index is used to retrieve the corresponding documents from the class ‘youtube’. As stated, our search engine attempts to run some results in the same window. The window shows video clips. When the user clicks on the video it opens in the same window, not in another browser.

7- **Academic:** Books and other reference materials are essential aspects of all Wikipedia articles. We indexed all the external links by their anchor texts, and consequently the Wikipedia pages in window ‘1’ are scanned for their external links. We used a simple parser to extract the keyword ‘book’ from each anchor text (if it was available). The resulting pages are grouped in this window, and each vector in the window is represented by its logo (photo) and the title of the book.

Overall, our search interface is structured as an entity-centric approach. This means that the results are structured into groups of entities for particular topics, which encourages the user to interact with the current results. This functionality makes our searcher more dynamic, with results that interact in a social context.

Figure 6.3 Our search engine interface

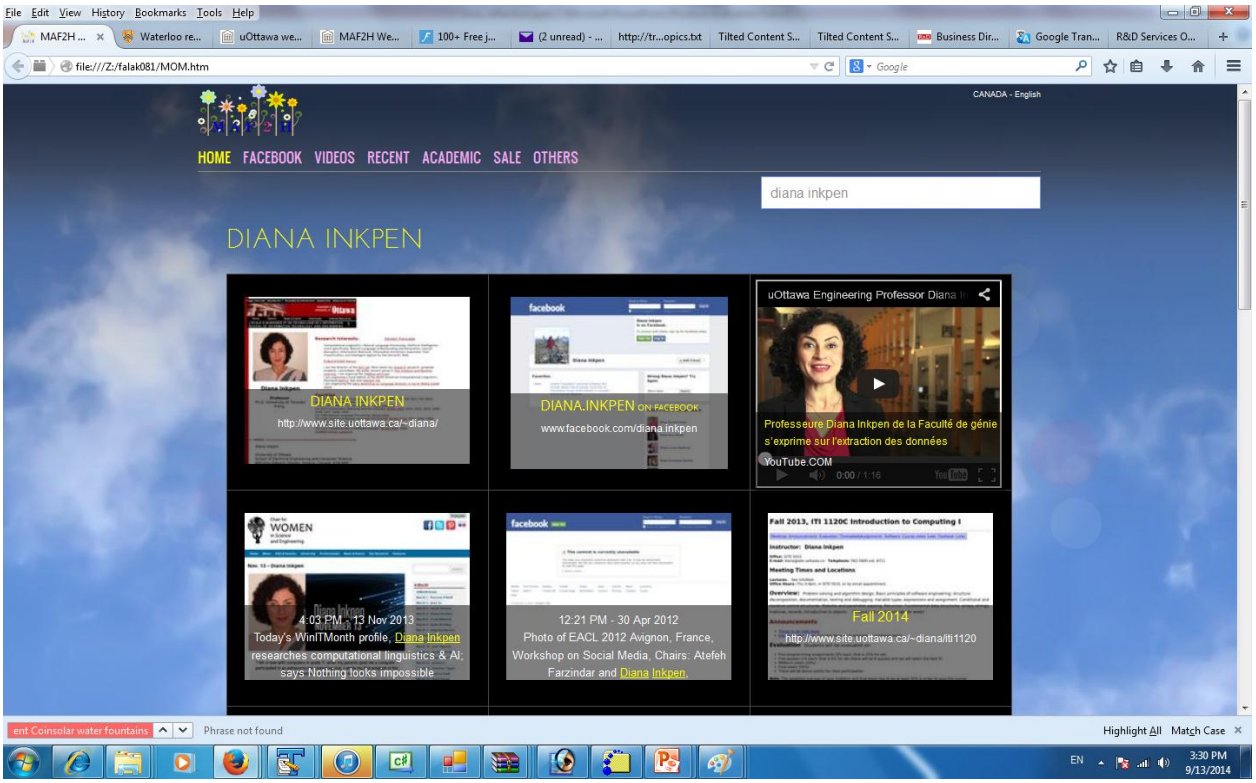


Figure 6.4 Our results interface

6.7 Discussion

Current search engines do not fully leverage real-time data, nor do they specialize in indexing only the domains that are crawled, or exploit social data and real-time blogs. This chapter proposed that the Social and Classical Webs be integrated, and that collective knowledge systems are the 'killer' applications of this union. The keys to using collective knowledge systems effectively and getting true collective results are tightly integrated the user-contributed content and the machine-gathered data and to harvest the relevant results from this combination of structured and unstructured information.

In this chapter, we discussed our advanced search engine that uses real-time data over social sites. Our model produces interactive query results from richly unstructured and interlinked data collected from disparate sources on the Social Web (Facebook and Twitter). Our resources made use of both social and classical data, represented by anchor texts from the web pages in our collection. The 0.5 billion documents (set A) we used involved approximately 3.3 billion anchors, which makes our model very diverse and capable of addressing scalability issues effectively.

Our new approach demonstrates what can be done in a collective knowledge system. It uses social processes to attract content providers, 'snap-to-grid' to elicit structured data from users, pivot browsing and faceted searching to aggregate the structured and unstructured data, and clustering technology to induce implicit underlying dimensions. In addition, to power our search engine and harvest the value of the collected knowledge, it combines user preferences data. Despite the grid density of our interface structure, our system does not require named entity extraction, taxonomic categorization, semantic search or other techniques associated with the Social Web.

We also presented our novel faceted navigational interface that uses the hyperlinks in Web pages in an interactive interface design, to clearly convey navigation choices and maintain the context of a search session. This type of display attempts to maximize the amount of text shown, while retaining a look that is as uncluttered as possible.

Improving information seekers' human-computer interaction experience is the main task of most search engines. This is a rapidly developing area, and improvements in the interface will likely lead to enhanced search results and better-enabled information creation. Search engines with features that improve social searching and result interfaces are important today. The most significant aspect of our design is unobstructed access. Users primarily use search engines only for online searches, and other social features in the interface should be secondary. This saves time for users, and makes the interface more responsive. However, directly accessing emotional content is also very important to users. So, building a purposeful search engine means finding a balance between the dynamic content in the Web and static elements in the interface layout, and understanding how these ultimately affect the user experience.

Finally, we consolidate complementary results in entities for more sufficient data. Hence, our interface is different from current search engine interfaces. Results in the sizable-windows are arranged according to the priorities, relevance and importance.

However, our framework addresses some of the difficulties and drawbacks in modern search engines such as Google. Table 6.4 shows a comparison between the highly-ranked Google search engine and our search engine, by highlighting the key points of user needs.

Our Search Engine	**Google Search Engine**
1- Results are aggregated from offline and online indexes (Twitter and Facebook).	1- Results are retrieved from offline data only.
2- One server for indexing the collection and searching the query (horizontal platform).	2- Several servers and a distributed grid of computers (approximately 10,000) for indexing the collection and searching the query (vertical platform).
3- Results are more controlled and optimized as they are combined in one computer.	3- Results are less controlled because they are combined from several servers.
4- Results are displayed on blocks as groups of topics, making them more subjective and interactive.	4- Results are displayed in a simple list.
5- The index is composed of 50 million documents plus only uses the anchor texts of 500 million documents (approximately 3.3 billion anchors) and real-time data.	5- The index is composed from more than 10 billion offline documents (crawled documents).
6- Results display on one scrollable page as groups of similar topics (e.g., videos, info, tweets, Facebook, books, photos).	6- Results display on a simple list and split onto several pages.
7- It is suitable for all types of queries.	7- It is suitable for all types of queries.
8- The response time is milliseconds.	8- The query response time is milliseconds.

Table 6.4 General comparison between the Google search engine and our search engine

Chapter 7

Conclusions and Future Directions

7.1 Conclusions

The main objective of the thesis was to create an indexing method that facilitates finding valuable and relevant information in Web collections. Our system is meant to index the entire content of a web collection, but to consider only the titles, URLs, and other meta-data for capturing the topics of contents. We used different indexing algorithms that exploit the query and document structures. We believe these novel methods will improve the design and functionality of future Web search engines.

In this work, we build models (presented in Chapters 3, 4 and 5) that can identify relevant documents from Web collections. To do this, we used various representation techniques based on document content, query structures (the variation of query terms occurring in the document content based on query structures), and the applicable knowledge available in Wikipedia. We showed that the Wikipedia structure and content provide valuable knowledge that helps find relevant data in Web collections. We find relevant pages based on term impact, a novel notion which is an improvement on term frequency, inverse document frequency, and similar weighting schemes used by traditional systems.

First, the model that obtained relevant results for both types of Web tasks (Chapter 3) used the impact of meta terms that were extracted from document-tags (URLs, titles, headings, images). The main contents of documents were excluded from the indexing process. We found not all documents focus their topics on meta-contents and meta-contents were not enough for capturing the document keywords. Therefore, this approach was fail for retrieving relevant results for some queries especially with those types focused on diversity tasks.

Second, the model that obtained high relevant results for both types of Web tasks as well (Chapter 4) used the impact of terms that were extracted only from URLs, titles and headings. We decreased the number of document's tags and we increased the area of indexing. To compromise between the speed of indexing and the size of index, we exploited the document keywords that exist only in meta-data; in which only, the meta-terms that had higher impact on the document-content were considered as prominent terms in their documents, otherwise the documents were excluded from the index. We found that the model was not entirely adequate, so

we added more knowledge and information to support the indexed data. We used the knowledge available in Wikipedia and information in the Alexa network to enhance the list of retrieved documents, by first extracting the relevant data from Wikipedia and formalizing it into valuable knowledge stored in a structured index. Thus, the Web documents that were indexed and stored in the equivalent index node as the Wikipedia pages were ranked differently than documents stored in the index node alone. This meant that the documents stored with the relative Wikipedia pages are enhanced by the knowledge available in the parent nodes. Some documents stored in the index alone could potentially also be enhanced via their domain names; that is, a domain that is ranked high by ALEXA.NET will have a higher probability of being retrieved, due to its domain popularity.

We discovered some drawbacks during the experimental testing phase of this method. For example, not all documents have prominent terms in their URLs and titles, particularly those with many topics in their content. The drawbacks in our second model are related to how the Wikipedia documents are indexed. Though 50% of these documents are redundant, they have different article titles or are not classified as Wikipedia topics (i.e., containing only definitions and descriptions). Another consideration is that we used our own methods to filter the spam documents from the index file. In the realistic approach, the assessment team employed the UWaterloo spamming file to filter the spam documents from the relevantly judged documents. This step did not work with our model, because the other models were not obliged to filter out their results using the UWaterloo spam file. In our case, this caused confusion between our results and the relevance judgments.

Our third model (Chapter 5) addressed these drawbacks and those from our previous models, and achieved very good results. We also experimented with more diverse query representations, using five types of index structures to address almost all other types of query structures. In Chapter 5, we developed our previous index and built an index that is more advanced and efficient than the previous indexes from Chapters 3 and 4. To increase system efficiency, we reduced the path from the root to each node in the index, such that the path used literal names instead of encoding names (two nodes for each term or phrase rather than eight). This allowed the data to be indexed and retrieved very quickly and the index minimized efficiently. Our results showed that this model is not only an improvement over our previous models but, compared to the top approaches in the TREC 2012 Web track, it is superior to all other methods, as well. In

conclusion, we showed the impact of using Wikipedia for enhancing web indexing and searching. Through the comparisons in our index classes, our experiments showed that Wikipedia has more impact than other index classes, even though other classes shared their information with Wikipedia.

7.2 Future Directions

We demonstrated the usefulness of Wikipedia for adhoc and diversity search and we showed the importance of term impact / weighting. Nonetheless, our system did not perform well for some type of queries, and this type of queries deserve further investigation. For example, when a user types a few terms in a WIR system as a query, the system understand relatively well how those query terms and documents should be weighted. Rare query terms provided by a user are very important for predicting document relevance if there is a weak matching between the query and the document. But when some query terms match a topic in Wikipedia collection, these query terms should have a different weight / importance. We recommend several lines of future work, first related to term impact/weighting and scalability:

1- We believe the retrieved documents for seldom terms in difficult queries could have been relevant, but they were not, because documents in the relevant judgment file were selected from another part of the dataset (set 'A'). For a few query terms, our system failed to capture the right topic from subset 'B'. We think that indexing the full set 'A' would close the gaps of these kinds of queries in order to satisfy all users' viewpoints. Though set 'A' is too large for a centralized index, computing and using term impact is possible if the right tradeoff is found between the size of the index and the users' needs.

2- We believe that the term weighting in the diversity task in particular deserves a lot more work; for instance we used the subset 'B' of the collection, which contains the main root for each website that was crawled; this means that subset 'B' focused on the main pages for each website/domain, and those documents mostly hold the main keywords in the meta-data. This is why we got the significant results for using the meta-keywords for capturing the document topics; but for other documents that were crawled and saved it in the set 'A', the meta-data may or may not useful for finding the document topics.

3- Our aim was to design a centralized index regardless the size of data, but we were limited to the subset 'B' of the collection due to hardware limitation. If more resources are available

(hardware), using our approach for indexing the whole set of data (subset ‘A’) is work to try in future, since we believe it may obtain good results. Extending our work with this large set of data will reveal more issues for parameter setting for our model, which would require further investigation.

Another issue that our system could not address is that the Wikipedia contains pages for famous people and current celebrities, but not for current professionals. Our system crawled and indexed social data from Facebook and Twitter in real-time fashion, also focused on celebrities. Other social platforms, such as LinkedIn, should be studied further and added in future versions of our system.

For the diversity task, our system focused on retrieving documents from the specific domains/websites which could be relevant for increasing the diversity, e.g., about.com, answers.com, etc., and the data is indexed topically for these domains/websites. As future recommendation and besides the data that was crawled by our system in real-time, we believe that more crawling and retrieving should be done in real-time by using Semantic Web technology (RDF and other markups). The index could be\come more compressed or even move towards having no local index.

Another direction of future work is to connecting our framework to distributional semantics models in order to achieve better empirical coverage of the natural language semantics phenomena.

Finally, we recommended to using our framework with another datasets gathered from other resources rather than using only data from TREC.

Appendix I: Training Queries

Training Queries TREC 2009

1:obama family tree
2:french lick resort and casino
3:getting organized
4:toilet
5:mitchell college
6:kcs
7:air travel information
8:appraisals
9:used car parts
10:cheap internet
11:gmat prep classes
12:djs
13:map
14:dinosaurs
15:espn sports
16:arizona game and fish
17:poker tournaments
18:wedding budget calculator
19:the current
20:defender
21:volvo
22:rick warren
23:yahoo
24:diversity
25:euclid
26:lower heart rate
27:starbucks
28:inuyasha
29:ps 2 games
30:diabetes education
31:atari
32:website design hosting
33:elliptical trainer
34:cell phones
35:hoboken
36:gps
37:pampered chef
38:dogs for adoption
39:disneyland hotel
40:michworks
41:orange county convention center
42:the music man
43:the secret garden
44:map of the united states
45:solar panels
46:alexian brothers hospital
47:indexed annuity
48:wilson antenna
49:flame designs
50:dog heat

Appendix II: Testing Queries

Testing Queries TREC 2010	Testing Queries TREC 2011	Testing Queries TREC 2012
51:horse hooves	101 :ritz carlton lake las vegas	151:403b
52:avp	102: fickle creek farm	152:angular cheilitis
53:discovery channel store	103: madam cj walker	153:pocono
54:president of the united states	104: indiana child support	154:figs
55:iron	105: sonoma county medical services	155:last supper painting
56:uss yorktown charleston sc	106: universal animal cuts reviews	156:university of phoenix
57:ct jobs	107: cass county missouri	157:the beatles rock band
58:penguins	108: ralph owen brewster	158:septic system design
59:how to build a fence	109: mayo clinic jacksonville fl	159:porterville
60:bellevue	110: map of brazil	160:grilling
61:worm	111: lymphoma in dogs	161:furniture for small spaces
62:texas border patrol	112: kenmore gas water heater	162:dnr
63:flushing	113: hp mini 2140	163:arkansas
64:moths	114: adobe indian houses	164:hobby stores
65:korean language	115: pacific northwest laboratory	165:blue throated hummingbird
66:income tax return online	116: california franchise tax board	166:computer programming
67:vldl levels	117: dangers of asbestos	167:barbados
68:pvc	118: poem in your pocket day	168:lipoma
69:sewing instructions	119: interview thank you	169:battles in the civil war
70:to be or not to be that is the question	120: tv on computer	170:scooters
71:living in india	121: sit and reach test	171:ron howard
72:the sun	122: culpeper national cemetery	172:becoming a paralegal
73:neil young	123: von willebrand disease	173:hip fractures
74:kiwi	124: bowflex power pro	174:rock art
75:tornadoes	125: butter and margarine	175:signs of a heartattack
76:raised gardens	126: us capitol map	176:weather strip
77:bobcat	127: dutchess county tourism	177:best long term care insurance
78:dieting	128: atypical squamous cells	178:pork tenderloin
79:voyager	129: iowa food stamp program	179:black history
80:keyboard reviews	130: fact on uranus	180:newyork hotels
81:afghanistan	131: equal opportunity employer	181:old coins
82:joints	132: mother's day songs	182:quit smoking
83:memory	133: all men are created equal	183:kansas city mo
84:continental plates	134: electronic skeet shoot	184:civil right movement
85:milwaukee journal sentinel	135: source of the nile	185:credit report
86:bart sf	136: american military university	186:unc
87:who invented music	137: rock and gem shows	187:vanuatu
88:forearm pain	138: jax chemical company	188:internet phone service
89:ocd	139: rocky mountain news	189:gs pay rate
90:mgb	140: east ridge high school	190:brooks brothers clearance
91:er tv show	141: va dmv registration	191:churchill downs
92:the wall	142: illinois state tax	192:condos in florida
93:raffles	143: arkadelphia health club	193:dog clean up bags
94:titan	144: trombone for sale	194:designer dog breeds
95:earn money at home	145: vines for shade	195:pressure washers
96:rice	146: sherwood regional library	196:sore throat
97:south africa	147: tangible personal property tax	197:idaho state flower
98:sat	148: martha stewart and imclone	198:indiana state fairgrounds
99:satellite	149: uplift at yellowstone national park	199:fybromyalgia
100:rincon puerto rico	150: tn highway patrol	200:ontario california airport

GLOSSARY

ADHOC TASK

Web search task designed to find documents that are highly relevant answers to queries that are known in advance.

APACHE LUCENCE

It is a Java-core open source library which can be used to produce a centralized index.

DIVERSITY TASK

Web search task similar to the adhoc task, but that is trying to retrieve relevant documents that cover all the aspects of a search query.

DOM

Document Object Model is an internal representation of HTML within the web browser.

HTML ELEMENT

An **HTML element** is an individual component of an HTML document or "web page", once this has been parsed into the Document Object Model.

HTML TAG

It is a basic unit of an HTML document. The name of the *element* is given in the name of the tag, and specifies the meaning associated with a block of text. Some *elements* are empty since they don't affect a block of text.

QUERY

Web search query is a string that a user enters into Web search engine to satisfy his/her information needs.

SEMANTIC WEB

It is a standard (common framework) that allows data to be shared and reused across applications.

SOCIAL-SEMANTIC WEB

It is a collective knowledge system that combines resources from different methodologies and technologies, and then provides information based on human contributions, e.g., DBpedia.

WIR

Web Information Retrieval is a system for semantic matching between a query and document in a Web collection, in order to obtain information that is relevant to user needs, e.g., entity finding and question answering.

WSE

Web Search Engine in a system (software and hardware) that is designed to index and search information on the large Web collection in an efficient and fast manner. The Web collection is usually gathered by a crawler.

BIBLIOGRAPHY

Abhishek, D. and Ankit, J. (2012), 'Indexing the World Wide Web: The Journey So Far', Google Publications Chapter.

Ahn, D., Jijkoun, V., Mishne, G., Muller, K., Rijke, M., and Schloback, S. (2004), 'Using Wikipedia at the TREC QA Track'. In Proceedings of the 13th Text Retrieval Conference (TREC).

Al-akashi, F. and Inkpen, D. (2010), 'University of Ottawa at TREC 2010 Web Track: Ranking Web Documents Using Meta-Data'. In Proceedings of 19th Text Retrieval Conference (TREC).

Al-akashi, F. and Inkpen, D. (2011), 'Ranking Web Pages Using Collective Knowledge'. In Proceedings of the 20th Text Retrieval Conference (TREC).

Al-akashi, F. and Inkpen, D. (2012), 'Intelligent Web page retrieval using Wikipedia knowledge'. In Proceedings of the 2nd International Conference on Web Intelligence, Mining, and Semantics (WIMS), ACM, isbn 978-1-4503-0915-8, pages 51.

Al-akashi, F. and Inkpen, D. (2012), 'Query-Structure Based Web Page Indexing'. In Proceedings of 21st Text Retrieval Conference (TREC).

Al-akashi, F. and Inkpen, D. (2014), 'Term Impact-Based Web Page Ranking'. In Proceedings of the 4th International Conference on Web Intelligence, Mining, and Semantics (WIMS), ACM, ISBN 978-1-4503-2538-7.

Al-Kamha, R. (2004), 'Grouping Search-Engine Returned Citations for Person Name Queries'. Master thesis, Brigham Young University.

Amati, G. (2003), 'Probabilistic models for information retrieval based on divergence from randomness'. Doctoral thesis, University of Glasgow.

Amati, G. and Carpineto, C. (2010), 'FUB at TREC-10 Web Track: A probabilistic framework for topic relevance term weighting'. In Proceedings of the 10th Text Retrieval Conference (TREC), pages 182-191.

Amitay, E. (2001), 'What Lays in the Layout: Using anchor-paragraph arrangements to extract descriptions of Web documents'. Doctoral thesis, Macquarie University.

Anh, V. and Moffat, A. (2010), 'The Role of Anchor Text in ClueWeb09 Retrieval'. In Proceedings of the 18th Text Retrieval Conference (TREC).

Asirvatham, A. and Ravi, K. (2007), 'Web Page Classification based on Document Structure'. In Proceedings of the IIIT conference in Information Technology.

Aslam, J. and Montague, M. (2001), 'Models for meta search'. In Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 276-284.

Baeza, R. and Ribeiro, B. (1999), 'Modern Information Retrieval'. ACM press.

Bahle, D., Williams, H., and Zobel, J. (2002), 'Efficient Phrase Querying with an Auxiliary Index'. In Proceedings of the 25th Annual International Conference on Research and Development in Information Retrieval (ACM SIGIR'02).

Bahle, D.,Williams, H., and Zobel, J. (2001), 'Compaction techniques for nextword indexes'. In Proceedings of 8th International Symposium on String Processing and Information Retrieval (SPIRE), pages 33-49.

Ballesteros, L. (2011), 'Resolving Ambiguity for Cross-Language Information Retrieval: A Dictionary Approach'. Doctoral thesis, University of Massachusetts Amherst.

Banerjee, S., Ramanathan, K., and Gupta, A. (2007), 'Clustering Short Texts using Wikipedia'. In Proceedings of the SIGIR'07, ACM 978-1-59593-597-7, pages 23-27.

Baykan, E., Henzinger, M., Marian, L., and Weber, I. (2009), 'Purely-based Topic Classification'. In proceedings of the 18th International World Wide Web Conference, ACM, 978-1-60558-487-4.

Beitzel, S., Jensen, E., Chowdhury, A., and Frieder, Q. (2007), 'Automatic classification of web queries using very large unlabeled query logs'. In Proceedings of the ACM Transactions on Information Systems (TOIS), Vol. 25, Issue 2, No. 2.

Becker, B.. Gschwind, S., Ohler, T., Seeger, B., Widmayer, P. (1996), 'An asymptotically optimal multiversion B-tree'. In Proceedings of the VLDB Journal — The International Journal on Very Large Data Bases, Volume 5 Issue 4, Pages 264-275.

Billerbeck, B. and Zobel, J. (2004), 'Techniques for Efficient Query Expansion'. In Proceedings of the 11th International Conference, SPIRE, pages 30-42.

Billerbeck, B., Craswell, N., Fetterly, D., and Najork, M. (2011), 'Microsoft Research at TREC 2011 Web Track'. In Proceedings of the 20th Text Retrieval Conference (TREC).

Bonnefoy, L., Bellot, P., and Benoit, M. (2010), 'LIA-iSmart at TREC 2010: An Unsupervised Web-based Approach for Filtering Answers'. In Proceedings of the 19th Text Retrieval Conference (TREC).

Brin, S. and Page, L. (1998), 'The Anatomy of a Large-Scale Hypertextual Web Search Engine'. In Proceedings of the 7th International World-Wide Web Conference (WWW).

Broschart, A. and Schenkel, R. (2010), 'MMCI at the TREC 2010 Web Track'. In Proceedings of the 19th Text Retrieval Conference (TREC).

Bryan, K. and Leise, T. (2006), 'The Linear Algebra behind Google'. Journal: SIAM Review, ACM, Vol. 48, Issue 3, pages 569-581.

Cafarella, M., Christopher, R., Suciu, D., Etzioni, O., and Banko, M. (2007), 'Structured Querying of Web Text A Technical Challenge'. In Proceedings of the Conference on Innovative Data Systems Research (CIDR).

Cafarella, M. (2009), 'Extracting and Managing Structured Web Data'. Doctoral Dissertation, University of Washington.

Cai, D., He, X., and Han, J. (2006), 'Tensor Space Model for Document Analysis'. The 29th Annual ACM International Conference on Research and Development in Information Retrieval (SIGIR, 1595933697/06/0008).

Callan, J., Lu, Z., and Croft, B. (1995), 'Searching distributed collections with inference networks'. In Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 21-28.

Chang, M. and Poon, C. (2008), 'Efficient Phrase Querying with Common Phrase Index'. Information Processing and Management: an International Journal, Vol. 44, Issue 2, pages 756-769.

Chapelle, O., Metzler, D., Zhang, Y., and Grinspan, P. (2010), 'Expected Reciprocal Rank or Graded Relevance'. Yahoo! Labs and Google Inc. In Proceedings of the 18th ACM Conference on Information and Knowledge Management (CIKM'09), pages 621-630.

Chen, X., Peng, Z., Wang, J., Yu, X., Lin, Y., Xu, H., and Cheng, X. (2010), 'ICTNET at Web Track 2010 Ad-hoc Task'. In Proceedings of the 19th Text Retrieval Conference (TREC).

Chen, X., Peng, Z., Wang, J., Yu, X., Liu, Y., Xu, H., and Cheng, X. (2011), 'ICTNET at Web Track 2011 Ad-hoc Task'. In Proceedings of the 20th Text Retrieval Conference (TREC).

Clarke, C., Craswell, N., and Soboroff, I. (2009), 'Overview of the TREC 2009 Web Track'. In Proceedings of the 18th Text Retrieval Conference (TREC).

Clarke, C., Craswell, N., Soboroff, I., and Craswell, N. (2010), 'Overview of the TREC 2010 Web Track'. In Proceedings of the 19th Text Retrieval Conference (TREC).

Clarke, C., Craswell, N., Soboroff, I., and Voorhees, E. (2011), 'Overview of the TREC 2011 Web Track'. In Proceedings of the 20th Text Retrieval Conference (TREC).

Clarke, C., Craswell, N., and Voorhees, E. (2012), 'Overview of the TREC 2012 Web Track'. In Proceedings of the 21st Text Retrieval Conference (TREC).

Cormack, G., Smucker, M., and Clarke, C. (2010), 'Efficient and effective spam filtering and re-ranking for large web datasets'. In Proceedings of the 19th Text Retrieval Conference (TREC).

Craswell, N. and Hawking, D. (2002), 'Overview of the TREC-2002 Web Track'. In Proceedings of the 11th Text Retrieval Conference (TREC), pages 86-95.

Craswell, N. and Hawking, D. (2003), 'Query-Independent Evidence in Home Page Finding'. ACM Journal Transactions on Information System (TOIS), Vol. 21, Issue 3, pages 286-313.

Craswell, N. and Hawking, D., (2010), 'Query-Independent Evidence in Home Page Finding', Trystan Upstill, Australian National University and CSIRO Mathematical and Information Sciences.

Craswell, N., Fetterly, F., Najork, M. (2010), 'Microsoft Research at TREC 2010 Web Track'. In Proceedings of the 19th Text Retrieval Conference (TREC).

Craswell, N., Hawking, D., and Robertson, S. (2010), 'Effective site finding using link anchor information'. In Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval, pages 250-257.

Creswell, D. (2002), 'Evaluation of Software-based Text Search and Retrieval Products that use Natural Language Expressions'. Technology Review Journal.

Cutting, D., Karger, D., Pedersen, J., and Tukey, J. (1992) 'Scatter/Gather: A cluster-based approach to browsing large document collections'. In Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval, pages 318-329.

Davison, B. (2002), 'The Design and Evaluation of Web Prefetching and Caching Techniques'. Doctoral thesis, State University of New Jersey.

Drymonas, E. (2006), 'Exploring multi-word similarity measures for Information Retrieval applications: the T-SRM method'. Master thesis, Technical University of Crete.

Elsayed, T., Asadi, N., Metzler, D., Wang, L., and Lin, J. (2010), 'UMD and USC/ISI: TREC 2010 Web Track Experiments with Ivory Tamer'. In Proceedings of the 19th Text Retrieval Conference (TREC).

ESKO, P. (2009), 'REMARKS ON SYNTAGMA AND WORD-FORMATION'. In Proceedings of the Folia Linguistica. Volume 16, Issue 1-4, Pages 241–262, ISSN 1614-7308.

Fox, E. and Shaw, J. (1994), 'Combination of Multiple Searches'. In Proceedings of the 2nd Text Retrieval Conference (TREC).

Fujita, S. (2001), 'Reflections on aboutness TREC-9 evaluation experiments at Justsystem'. In Proceedings of the 8th Text Retrieval Conference (TREC).

Fujita, S. (2002), 'More reflections on aboutness TREC-2001 evaluation experiments at Justsystem'. In Proceedings of the 10th Text Retrieval Conference (TREC), pages 331-338.

Gao, B., Anastasiu, D., and Jiang, X. (2010), 'Utilizing User-Input Contextual Terms for Query Disambiguation'. In Proceedings of the 23rd International Conference on Computational Linguistics: Posters, pages 329-337.

Garcia, E. (2011), 'A Tutorial on Okapi BM25'. RSJ-PM Tutorial.

Glover, E. (2001), 'Using Extra-Topical User Preferences to Improve Web-Based Metasearch'. Doctoral thesis, University of Michigan.

Grobelink, M. and Mladenic, D. (2005), 'Simple Classification into Larger Topic Ontology of Web Documents'. In Proceedings of the 20th National Conference on Artificial Intelligence (AAAI-05), 17th Conference on Innovative Applications of Artificial Intelligence (IAAI-05), Journal of Computing and Information Technology - CIT 13, Issue 4, pages 279-285.

Grossman, D. and Frieder, O. (2004), 'Information Retrieval Algorithms and Heuristics'. The 2nd Edition, Springer.

Gurrin, C. and Smeaton, A. (2004), 'Replicating Web Structure in Small-Scale Test Collections'. An Information Retrieval Journal, ACM, Vol. 7, Issue 3-4, pages 239-263.

Guthikonda, S. (2007), 'Kohonen Self-Organizing Maps', Journal of Statistical Software, Vol. 21, Issue 5.

Harman, D. (1995), 'Overview of the second Text REtrieval Conference (TREC-2): Information Processing and Management'. In Proceedings of the 2nd Text Retrieval Conference, pages 271-289.

He, J., Hollink, V. and Boscarino, C. (2011), 'CWI at TREC 2011: session, web, and medical'. In Proceedings of the 20th Text Retrieval Conference (TREC).

Hemayati, R.,Meng, W., and Yu, C. (2007), 'Semantic-Based Grouping of Search Engine Results Using WordNet'. In Proceedings of the joint 9th Asia-Pacific web and 8th international conference on web-age information management conference on advances in data and web management, Springer, pages 678-686.

Hearst, M. (2009), 'BOOK SEARCH USER INTERFACES'. CAMBRIDGE UNIVERSITY PRESS.

Hiemstra, D. and Hauff, C. (2010), 'MapReduce for Experimental Search'. In Proceedings of the 19th Text Retrieval Conference (TREC).

Huang, X., Hosseini, D., Rohian, H., and An, X. (2007), 'York University at TREC 2007: Genomics Track'. In Proceedings of the 16th Text Retrieval Conference (TREC).

Hudong, Li. (2003), 'An Inverted Index Generator for CINDI'. Master's thesis, Concordia University.

Imran, H. and Sharan, A. (2009), 'Thesaurus and query Expansion'. International Journal of Computer Science and Information Technology (IJCSIT), Vol. 1, No 2, pages 89-97.

Itakura, K., Clarke, Ch., Geva, Sh., Trotman, A., and Huang, W. (2011), 'Topical and structural linkage in Wikipedia'. In Proceedings of the 33rd European Conference on Information Retrieval, pages 460-465, 2011.

Jarvelin, K. and Kekalainen, J. (2002), 'Cumulative gain-based evaluation of IR techniques'. In Proceedings of the ACM Transactions on Information Systems (TOIS), Vol. 2, Issue 4, pages 422-446.

Javier Pic'on, J. (2009), 'Web People Search'. Doctoral thesis, Universidad Nacional De Educaci'on A Distancia, 2009.

Junichiro, M., Yutaka, M., Mitsuru, I., and Boi, F. (2004), 'Keyword Extraction from the Web for FOAF Metadata'. In Proceedings of the 1st International Workshop on Friend of a Friend, Social Networking and the Semantic Web.

Kamps, J., Kaptein, R., and Koolen, M. (2010), 'Using Anchor Text, Spam Filtering and Wikipedia for Web Search and Entity Ranking'. In Proceedings of the 19th Text Retrieval Conference (TREC).

Kaptein, R., Koolen, M., Kamps, J. (2010), 'Result Diversity and Entity Ranking Experiments: Anchors, Links, Text and Wikipedia'. In Proceedings of the 19th Text Retrieval Conference (TREC).

Karbasi, S. and Boughanem, M. (2006), 'Document Length Normalization Using Effective Level of Term Frequency in Large Collection'. In Proceedings of 28th European Conference on IR Research (ECIR).

Kaski, S., Honkela, T., Lagus, K., and Kohonen, T. (2004), 'WEBSOM-Self-organizing maps of document collections'. In Proceeding of 26th European Conference on IR Research (ECIR).

Kleinberg, J. (1999),'Authoritative Sources in a Hyperlinked Environment'. Journal of the ACM (JACM), Vol. 46, Issue 5, pages 604-632.

Kohonen, T. (1991), 'Self-Organizing Maps'. In proceedings of the IEEE 78, pages 1464-1480.

Kohonen, T. (2001), 'Self-Organizing Maps', Springer Series in Information Sciences, Vol. 30, Third Extended Edition.

Kohonen, T., Kaski, S., Lagus, K., Salog'rui, J., Honkela, J., Paatero, V., and Saarela, A. (2000), 'Self-organization of a massive document collection', IEEE Transactions on Neural Network, Vol. 11, pages 574-585.

Kolda, T., Bader, B., and Kenny, J. (2005), 'Higher-Order Web Link Analysis Using Multi-linear Algebra'. The Fifth IEEE International Conference on Data Mining (ICDM05).

Kriesel, D. (2011), 'A Brief Introduction to Neural Networks', dkriesel.com Publisher.

Kumar, S. A. (1997), 'Term Weighting Revisited', Doctoral thesis, Cornell University.

Langville, A. and Meyer, C. (2004), 'The Use of the Linear Algebra by Web Search Engines'. IMAGE Newsletter, Bulletin of the International Linear Algebra Society, 33.

Lee, J. (1997), 'Analyses of Multiple Evidence Combination'. In Proceedings of the 20th International Conference on Research and Development in Information Retrieval.

Lempel, R. and Moran, S. (2005), 'Rank-Stability and Rank-Similarity of Link-Based Web Ranking Algorithms in Authority-Connected Graphs'. An Information Retrieval Journal, ACM, Vol. 8, Issue 2, pages 245-264.

Leonard, D., Zhang, L., Lillis, D., Toolan, F., Collier, R., and Dunnion, J. (2010), 'UCD SIFT in the TREC 2010 Web Track'. In Proceedings of the 19th Text Retrieval Conference (TREC).

Li, H. (2003), 'An Inverted Index Generator for CINDI', Master thesis, Concordia University.

Liddy, E. (2007), 'How a Search Engine Works'. In Proceeding of the 12th Annual Search Engine Conference Best Paper Award.

Lioma, C., Blanco, R., Palau, R., and Moens, M. (2009), 'A Belief Model of Query Difficulty That Uses Subjective Logic'. Advances in Information Retrieval Theory Lecture Notes in

Computer Science Volume 5766, pages 92-103. In Proceeding of the Second International Conference on the Theory of Information Retrieval, ICTIR.

Lovins, J. (1969), 'Development of a Stemming Algorithm'. In Proceedings of the Mechanical Translation and Computational Linguistics', Vol. 11, nos. 1 and 2.

Lundquist, C., Grossman, D., Frieder, O., and Holmes, D. (1997), 'A Parallel Implementation of Relevance Feedback using the Relational Model'. In Proceedings of the World Multiconference on Systemics, Cybernetics, and Informatics.

Lundquist, C., Holmes, D., Grossman, D., Frieder, O., and McCabe, M. (1997), 'Expanding relevance feedback in the relational model'. In Proceedings of the TREC conference, pages 489-502.

MacKinnon, I. and Vechtomova, O. (2008), 'Improving Complex Interactive Question Answering Enhanced with Wikipedia Anchor Text'. In Proceedings of Advances in Information Retrieval, 30th European Conference on IR Research (ECIR).

Majavu, W. (2009), 'Classification of Web Resident Sensor Resources Using Latent Semantic Indexing and Ontologies'. Master thesis, University of the Witwatersrand.

Majavu, W., Zyl, T., and Marwala, T. (2008), 'Classification of web resident sensor resources using Latent Semantic Indexing and Ontologies'. In Proceedings of the IEEE International Conference on Source (SMC), pages 518-523.

Mandala, R., Takenobu, T., and Hozumi, T. (2000). 'Improving information retrieval system performance by combining different text-mining techniques'. In Proceedings of the Intelligent Data Analysis Journal, pages 489 – 511.

Manning, C., Raghavan, P., and Sch'tze, H. (2008), 'Introduction to Information Retrieval'. Cambridge University Press, ISBN-10: 0521865719, ISBN-13: 978-0521865715.

McBryan, O. (1994), 'GENVL and WWWW: tools for taming the Web'. In Proceedings of the First International Conference on World Wide Web (WWW).

McCabe, C. and Chowdhury, A. (1999), 'A Unified Environment for Fusion of Information Retrieval Approaches'. In Proceedings of the 18th international conference on Information and Knowledge Management (CIK), ACM, pages 330-334.

McCreadie, R., Macdonald, C., Santos, R., and Ounis, I. (2011), 'University of Glasgow at TREC 2011: Experiments with Terrier in Crowdsourcing, Microblog, and Web Tracks'. In Proceedings of the 20th Text Retrieval Conference (TREC).

Metzler, D. (2007), 'Automatic Feature Selection in the Markov Random Field Model for Information Retrieval'. In Proceedings of CIKM, pages 253-262.

Moffat, A., Webber, W., and Zobel, J. (2007), 'Strategic System Comparisons via Targeted Relevance Judgments'. In Proceedings of the ACM SIGIR, pages 375-382.

Moffat, A. and Zobel, J. (2008), 'Rank-biased precision for measurement of retrieval effectiveness'. In Proceedings of the ACM Transactions on Information Systems, pages 1-27.

Montague, M. and Aslam, J. (2002), 'Condorcet Fusion for Improved Retrieval', In Proceedings of the 11th international conference on Information and knowledge management (CIKM), pages 538-548.

Milajevs, M. and Bouma, G. (2013), 'Real Time Discussion Retrieval from Twitter'. In Proceedings of the International World Wide Web Conference Committee (IW3C2). Rio de Janeiro, Brazil. ACM 978-1-4503-2038-2/13/05.

Najork, M. (2009), 'The scalable hyperlink store', In Proceedings of the 20th ACM Conference on Hypertext and Hypermedia, pages 89-98.

Naveen and Dharmender, K. (2012), 'Web Search Engines: Mining Right Information'. International Journal of Computer Applications (IJCA).

Nguyen, D. and Callan, J. (2010), 'Combination of Evidence for Effective Web Search'. In Proceedings of the 19th Text Retrieval Conference (TREC).

Nie, L.,Davison, B., and Qi, X. (2008), 'Topical Link Analysis for Web Search'. In proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval, pages 91-98.

Osinski, S. (2004), 'Dimensionality Reduction for Search Results Clustering'. Master thesis, University of Sheffield.

Osinski, S. and Weiss, D. (2005), 'Lingo: A Concept-Driven Algorithm for Clustering Search Results'. IEEE Intelligent Systems, Vol. 20, pages 48-54.

Park, E., Moon, S., Jan, M., and Ra, D. (1992), 'Web Document Retrieval Using Sentence-query Similarity'. In Proceedings of the 11th Text Retieval Conference.

Pantoja, C., Marin, M., Costa, V., and Bonacic, C. (2011), 'An Evaluation of Fault-Tolerant Query Processing for Web Search Engines'. In Proceedings of the 17th international conference on Parallel Processing, Vol. 1, pages 393-404.

Plachouras, V. (2006), 'Selective Web Information Retrieval'. Doctoral thesis, University of Glasgow.

Plachouras, V., Ounis, I., and Amati, G. (2005), 'The Static Absorbing Model for Hyperlink Analysis on the Web'. Journal of Web Engineering, 4(2), pages 165-186.

Lempel, R. and Moran, S. (2001), 'The stochastic approach for link-structure analysis (SALSA) and the TKC effect'. In Proceedings of the ACM Transactions on Information Systems - TOIS.

Rachel Lo, T., He, B., and Ounis, I. (2005), 'Automatically Building a Stopword List for an Information Retrieval System'. The 5th Dutch Belgium Information Retrieval Workshop (DIR), Utrecht, the Netherlands.

Radlinski, F., Kurup, M., and Joachims, T. (2008), 'How Does Clickthrough Data Reflect Retrieval Quality'. In Proceedings of the 17th ACM Conference on Information and Knowledge Management, pages 43-52.

Robertson, S., Zaragoza, H., Taylor, M. (2004), 'Simple BM25 extension to multiple weighted fields'. In Proceedings of the ACM Conference on Information and Knowledge Management', pages 42-49.

Rocchio, J. (1971), 'Relevance feedback in information retrieval'. In the Smart Retrieval System, Experiments in Automatic Document Processing, pages 313-323.

Salton, G. Wong, A., and Yang, C. (1975), 'A Vector Space Model for Automatic Indexing'. Communications of the ACM, Vol. 18, issue11, pages 613-620.

Salton, G., and Buckley, C. (1988), 'Term-weighing approaches in automatic text retrieval'. Information Processing and Management: an International Journal archive, Vol. 24, Issue 5, pages 513-523.

Santos, R., McCreadie, R., Macdonald, C., and Ounis, I. (2010), 'University of Glasgow at TREC 2010: Experiments with Terrier in Blog and Web tracks'. In Proceedings of the 19th Text Retrieval Conference (TREC).

Selberg, W. (1999), 'Towards Comprehensive Web Search', Doctoral thesis, University of Washington.

Serdyukov, P. and Vries, A. (2009), 'Delft University at the TREC 2009 Entity Track: Ranking Wikipedia Entities'. In Proceedings of the 18th Text Retrieval Conference (TREC).

Shen, D. (2007), 'Learning-Based Web Query Understanding'. Doctoral thesis, The Hong Kong University of Science and Technology.

Shen, D., Pan, R., Sun, J. Wu, K., Yin, J., and Yang, Q. (2006), 'Query enrichment for web-query classification'. ACM Transactions on Information Systems (TOIS), vol. 24, Issue 3, pages 320-352.

Shen, D., Sun, J., Yang, Q., and Chen, Z. (2006), 'Building bridges for web query classification'. In Proceedings of the 29th annual international ACM SIGIR06 conference on Research and development in information retrieval, pages 131-138.

Sigur'sson, M. and Halling, S. (2007), 'Zeeker: A topic-based search engine'. Master thesis, the Technical University of Denmark (DTU).

Singhal, A. and Kaszkiel, M. (2000), 'AT&T at TREC-9'. In Proceedings of the 9th Text Retrieval Conference (TREC).

Singhal, A. (2001). "Modern Information Retrieval: A Brief Overview". Bulletin of the IEEE Computer Society Technical Committee on Data Engineering 24 (4): 35–43.

Smucker, M. (2011), 'Crowdsourcing with a Crowd of One and Other TREC 2011 Crowdsourcing and Web Track Experiments'. In proceedings of the 20th Text Retrieval Conference (TREC).

Song, R., Xin, G., Shi, S., Wen, J., and Ma, W. (2006), 'Exploring URL Hit Priors for Web Search'. In Proceedings of 28th European Conference on IR Research- Advances in Information Retrieval (ECIR).

Steinbach, M., Karypis, G., and Kumar, V. (2000), 'Comparison of Document Clustering'. KDD Workshop on Text Mining, pages 109-111.

Strohman, T., Metzler, D., Turtle, H., and Croft, W. (2005), 'Indri: A language-model based search engine for complex queries'. In Proceedings of the International Conference on Intelligence Analysis.

Symonds, M., Zuccon, G.,Koopman, B., and Bruza, P. (2012), 'QUT Para at TREC 2012 Web Track: Word Associations for Retrieving Web Documents'. In Proceedings of the 21st Text Retrieval Conference (TREC).

Tang, T. (2006), 'Quality-oriented information retrieval in a health domain'. Doctoral thesis, Australian National University.

Upstill, T. (2005), 'Document Ranking Using Web Evidence'. Doctoral thesis, Australian National University.

Upstill, T., Craswell, N., and Hawking, D. (2003), 'Query-independent evidence in home page finding'. In Proceedings of the ACM Transactions on Information Systems (TOIS), Vol. 21, Issue 3, pages 286-313.

Upstill, T., Craswell, N., and Hawking, D. (2003), 'Query- Independent Evidence in Home Page Finding'. ACM Transactions on Information Systems, Vol. 21, No. 3, pages 286-313.

Vesanto, J. and Alhoniemi, E. (2000), 'Clustering of the Self-Organizing Map'. In Proceedings of the IEEE Transactions on Neural Networks, Vol. 11, No. 3.

Vogt, C. and Cottrell, W. (1999), 'Fusion via a Linear Combination of Scores'. Information Retrieval journal, Vol. 1, Issue 3, pages 151-173.

Voorhees, E., Gupta, N., and Johnson-Laird, B. (1994), 'The Collection Fusion Problem'. In Proceedings of the 3rd Text Retrieval Conference (TREC).

W. Bruce, W., Metzler, D., and Strohman, T. (2009), 'Search Engines: Information Retrieval in Practice'. Wddison-Wesley, first edition, ISBN-10: 0136072240, ISBN-13: 978-0136072249.

Wang, H., Lee, K., and Yang, Q. (1999), 'Real-Time Document Collection Search Engine With phrase Indexing'. United States Patent, US 5920854.

Wang, H., Lee, K., and Yang, Q. (2004), 'Search Engine with Natural Language-Based Robust Parsing for User Query and Relevance Feedback Learning'. United States Patent, US 6,766,320 B1.

Weber, I (2007), 'Efficient Index Structures for and Applications of the Complete Search Engine'. Doctoral Thesis, Universit'at des Saarlandes.

Weiss, D. (2001), 'A Clustering Interface for Web Search Results in Polish and English'. Master thesis, Poznan University of Technology.

Weiss, D. (2006), 'Descriptive Clustering as a Method for Exploring Text Collections'. Doctoral thesis, Poznan University of Technology.

Wen, J., Nie, J., Zhang, H. (2002), 'Query clustering using user logs'. ACM Transactions on Information Systems (TOIS), Vol. 20, Issue 1, pages 59-81.

White, T. (2009), 'Hadoop- The Definitive Guide: MapReduce for the Cloud'. O'Reilly, isbn 978-0-596-52197-4, pages 1-501.

Xing, Y and James, A (2010), 'A Content based Approach for Discovering Missing Anchor Text for Web Search', In Proceedings of SIGIR'10, pages 19–23.

Yang, K. (2002), 'Combining Text, Link, and Classification-based Retrieval Methods to Enhance Information Discovery on the Web'. Doctoral thesis, University of North Carolina.

Ye, Z., Huang, X., He, B., and Lin, H. (2009), 'York University at TREC 2009: Relevance Feedback Track'. In Proceedings of the 18th Text Retrieval Conference (TREC).

Yu, Y., Isard, H., Fetterly, D., Budiu, M., Erlingsson, U., Gunda, P., and Currey, J. (2008), 'DryadLINQ: a system for general-purpose distributed data-parallel computing using a high-level language'. In Proceedings of the 8th USENIX Symposium on Operating Systems Design and Implementation, pages 1-14.

Zamir, O. (1999), 'Clustering Web Documents: A Phrase-Based Method for Grouping Search Engine Results'. Doctoral thesis, University of Washington.

Zhao, H., Meng, W., and Yu, C. (2007), 'Mining Templates from Search Result Records of Search Engines'. In Proceedings of the 13th ACM International Conference on Knowledge Discovering and Data Mining (SIGKDD), pages 884-893.

Zheng, W. and Fang, H. (2011), 'A Study of Pattern-based Subtopic Discovery and Integration in the Web Track'. In Proceedings of the 20th Text Retrieval Conference (TREC).

Zheng, W., Wang, X., and Fang, H. (2010), 'University of Delaware at Diversity Task of Web Track 2010'. In Proceedings of the 19th Text Retrieval Conference (TREC).

Zheng, W., Wang, X.,Fang, H., and Cheng. H. (2011), 'An exploration of pattern-based subtopic modeling for search result diversification'. In Proceedings of the annual joint conference on Digital libraries (JCDL), pages 378-388.