# Query-Structure Based Web Page Indexing

Falah H. Al-akashi falak081@uottawa.ca
Diana Inkpen diana@eecs.uottawa.ca
School of Electrical Engineering and Computer Science
University of Ottawa
Ottawa, K1N6N5, Canada

## ABSTRACT

Indexing is a crucial technique for dealing with the massive amount of data present on the web. In our third participation in the web track at TREC 2012, we explore the idea of building an efficient query-based indexing system over Web page collection.

Our prototype explores the trends in user queries and consequently indexes texts using particular attributes available in the documents. This paper provides an in-depth description of our approach for indexing web documents efficiently; that is, topics available in the web documents are discovered with the assistance of knowledge available in Wikipedia. The well-defined articles in Wikipedia are shown to be valuable as a training set when indexing Webpages. Our complex index structure also records information from titles and urls, and pays attention to web domains.

Our approach is designed to close the gaps in our approaches from the previous two years, for some queries. Our framework is able to efficiently index the 50 million pages available in the subset B of the ClueWeb09 collection. Our preliminary experiments on the TREC 2012 testing queries showed that our indexing scheme is robust and efficient for both indexing and retrieving relevant web pages, for both the ad-hoc and diversity task.

## 1 INTRODUCTION

The rapid growth and massive quantities of data on the Internet have increased the importance and complexity of information retrieval systems. The amount and the diversity of the web data introduce shortcomings in the way search engines rank their results.

In this work, we propose to push search engine behaviour in a new direction, away from link analysis and toward actual content and topic analysis of web pages. Typically, the current

approach of content-based indexing does not scale well for phrasal queries, because the positions of the words need to be recorded, requiring a lot of storage space. On another hand, indexing without considering word location for proximity search causes two documents to seem similar if they have some words in common even when they are on different in topics. Term frequency is important for determining the topic of the documents; but this is not case for all document configurations, because documents may contain different topics located on different parts.

In order to experiment with a new intelligent search model for the web, we evaluate our algorithm by using the queries/topics made available by NIST in 2012, as part of TREC web search track. The advantage of using these queries is that NIST made available expected solutions, called relevance judgments, consisting of lists of documents that are relevant answers to each query; therefore we can evaluate our proposed model. In TREC 2012, we focused on two tasks: the classic text retrieval task called adhoc retrieval, and the diversity task.

The rest of our paper describes each component in our system. Section 2 describes our indexing approach. Section 3 describes the query processing steps. Section 4 describes query expansion. Section 5 explains the experimental results for our run. Finally, section 6 closes this paper with our conclusion regarding this approach.

## 2 OUR APPROACH

Indexing is crucial for the task of finding relevant information on the Web. Various indexing methods are used in a wide range of applications, such as Home-page finding, Entity finding, and Web pages classification. The design of highly-scalable indexing algorithms is needed, especially with an estimate of one billion pages currently accessible on the web. Previous work classifies indexing of web documents in two types: word-based and phrase-based indexing [6].

In our entry in TREC 2012, we built an index that handles phrases and concepts with different length using two references: Wikipedia articles and the Million Query Track[1] data. Our index is structured as a core distributed search files. It uses sub-trees of a fixed length; each internal node has one leaf that contains documents relevant to that node. The depth from the root to each leaf corresponds to the word or phrase which is two nodes for each word or phrase (we use the first term from each phrase for labelling the first level in the index; whereas the

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

remaining terms are used for labelling the second level). The subsequent nodes from the root to the leaves map all index words or phrases. Each leaf in the index holds all indexed documents regarding a particular topic, and the path name from the root to each leaf corresponds to a possible query terms. Since our index from last year's system could not find relevant results for some queries, we addressed this drawback in this model.

We used a pool of five index classes. The index nodes in each class use a regular name instead of the coding names that we used in previous versions of our system. Each class contains a particular type of indexed data, as listed and shown in the figure below:

1- *Wiki*: index class that holds all Wikipedia documents.

2- *Domains*: index class that holds all domain names; as well as all their titles and urls.

3- *Title*: index class that holds all documents that we indexed using phrases from their titles.

4- *Words Combination Index:* index class such that the nodes are labeled with keywords selected from urls and titles; the content of nodes holds vectors of significant terms.

5- *Topical Index:* index class that holds all other documents except Wikipedia pages and homepages. The documents in this class are indexed based on our collective phrases selected from Wikipedia and One Million Query Track.

Figure 1 shows the architecture of our system. In the following sections, we will describe each index class in more details.

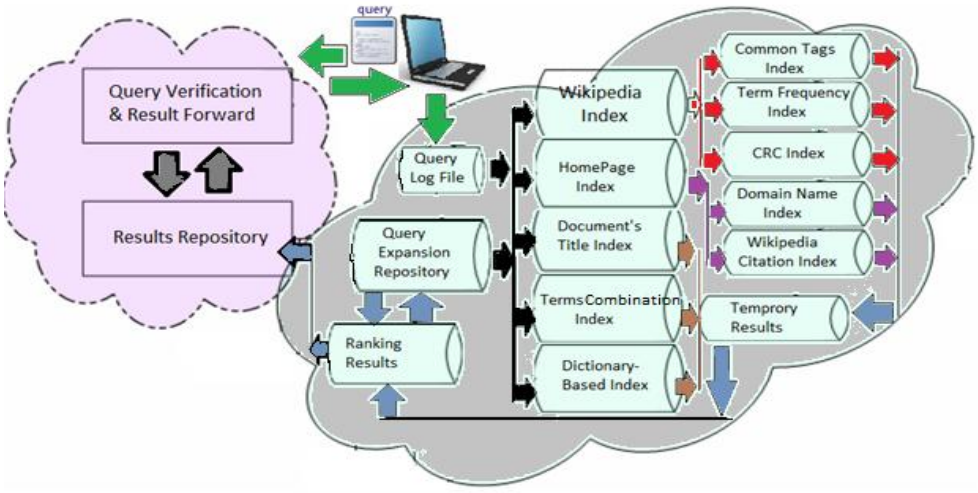


Figure 1: The Architecture of Our Index Structure

## 2.1Wikipedia Repository Index

Wikipedia contains approximately 5 million articles in English. The data in each article is structured into several fields, and sometimes it has a relationship with other articles using tags or links to expand a certain topic. Each document has a unique vocabulary name and identifier; each article has multi-faceted vocabulary terms to describe the same article's content. To efficiently index Wikipedia documents, we used three equivalent methods for grouping and indexing similar articles, in terms of reducing the index size and the time required for indexing and searching.

### 2.1.1 Using Common Tags (Vocabularies)

Each document in Wikipedia has a unique vocabulary name and identifier. Sometimes, Wikipedia articles use different vocabulary terms to describe the same article. This means that some articles are repeated many times with different vocabulary names and the TREC identifiers used in the *ClueWeb* data. All similar articles must be grouped and bounded together in a cluster of similar articles. To accomplish this, our system scanned through the content of each article and gathered all the significant tags and vocabulary terms (those that were in bold fonts). Each article name is used for creating and titling the index node; whereas the significant tags are stored together in the content node. Overall, during the scanning all Wikipedia documents, documents that share the same tags must share the same contents. Thus, the content of any node in the index aggregates all the similar articles.

### 2.1.2 Using Terms' Impact

Not all Wikipedia articles are real articles. In our investigation, there are 50% of articles categorized as short; they contain only a few words. Also, some topics are covered more comprehensively than others. We propose to use term impact again rather than a term frequency. We investigate new ways to compute term impact, as explained later on. After computing the weights/impact value for each term in the documents, the terms with high weights are selected to be representative to their documents. We used a strict cut-off (threshold) value of impact in the range of [2.5-5.5]. The candidate terms are built and used to label the index nodes (if they were not built previously), while the document identifiers are stored in the content of corresponding nodes. Therefore, each node in the index could store a cluster of documents that imply equal or close terms impact.

### 2.1.3 Using the CRC-Dictionary

To save time for indexing and to reduce the size of the index, we used a checksum algorithm (CRC16) to group the documents that have high similarity (they are redundant). We also avoided repeating the computation of term frequency in similar documents. To do so, the system scanned through each document's content in the Wikipedia corpus and computed the CRC16 for the header paragraph. We selected the header paragraphs, because some articles have little changes in other paragraphs. The generated CRC16 values were used for represent the keys of a dictionary *D (a hash table)*, whereas the values of the keys contain all the document identifiers (TREC identifiers). Overall, during the scanning all Wikipedia documents, documents that share the same contents must share the same CRC values. Thus, the content of any key in the dictionary aggregates all the similar articles. Finally, the dictionary *D* was transferred to the index pool.

## 2.2 Home Pages Indexing

We used two complementary methods for indexing the Home Pages. The first method stores the indexed data at a high level stage in the index, whilst the second method stores the indexed data at a low level stage in the index. Both results are combined together in the class of home pages located in the index pool.

### 2.2.1 Using Domain Names

Our system indexed all domain names and created a new class in the index pool for each home page. Each node in the index's class is named by the domain's name; additionally, all urls that belong to each site are stored as vectors in that node. Our system is built to index nodes from the domain names regardless of the type of extension. The contents of the nodes contain the documents' ids and their urls. However, we assigned a different ranked value for each extension.

### 2.2.2 Using Wikipedia External Links

Wikipedia is often a good reference for most homepages. The external links in the Wikipedia repository are used for the homepage finding task and potentially work better than searching in anchor texts of web pages [14]. Our system scanned the content of each article and indexed the home page from the external link section. In general, Wikipedia writers used different terms for

introducing a home page, i.e. "website", "homepage", "official", etc. To tackle this situation, we used regular expressions for extracting the home pages from Wikipedia documents.

## 2.3 Document Titles Index

We notices that phrases in the titles are often connected together by using conjunction words, i.e., "or" , "and" , "at" , "in" , "on" , "by" , "with" , "from" , or "for"; or punctuation characters, i.e., ":" , "|" , "(" , ")" , "-" , "," , or "&". Thus, segmenting the titles of documents into phrases is essential to know the most important key-phrases that the document content focuses on. We used these characters and words for partitioning the titles into list of terms and phrases.

On the other side, not all these terms or phrases have equal impact in their document's content. To compute the impact of the extracted terms and phrases, our system computed the cosine similarity between a vector that represented each item (term or phrase) and a vector that represented the document's content. Finally, the extracted items were used for creating and labeling the nodes in the index class; whilst the contents of nodes include: the TREC identifiers, and the impact values.

## 2.4 Terms Combination Index

Usually, queries refer to terms that are available in different positions in documents; for instance some terms are located in the urls; whereas the remaining terms are located in the content. A combination index class, however, focuses on this type of query.

To build the class, first, the frequency of each term in the documents collection was computed. Second, the three terms that were the most frequent in each document were selected. Finally, for each keyword in the url and title of document, the index node was created and named as the name of that keyword; whereas the content of node held the three terms that most frequent. An example for the link "*http://www.opm.gov/oca/pay/HTML/02maxgs2.asp* , the most frequent terms in this document's content are: "GS", "pay", and "rate"; the terms "opm", "oca", and "pay" extracted from the URL; and the terms "MAXIMUM", "GENERAL", "SCHEDULE", and "LIMITATIONS" extracted from the title were used for creating the index nodes. The content of each node holds a list of the most frequent terms ("GS", "pay", and "rate"), in addition to the document identifier.

## 2.5 Topical Dictionary-Based Index

Our final technique for building the phrasal index aims at detecting the main topics of documents. It is composed by three mutual hash tables, such that each hash table has a unique and specific topic. All hash tables are built on-the-fly before mapping them to the physical storage. We used two phases during the creating of this index:

- **Phase 1 - Document-topic investigation**

The first step in this phase is investigating the most important topic in the document. We used three hash tables that work cooperatively. The topics of documents were turned out from the title of Wikipedia articles and from the list of queries available in the file "Million Query Track"[2]. After scanning through the Web documents, each document was assigned to a particular topic. Thus, each node contains a set of documents that involved the same topic; as shown here as an example:["angular cheilitis", {document-1, url | .... |document-n,url}]

- **Phase 2 -Computing Topics' Weights**

Usually, if a site focuses on a topic "*civil right movement*", all documents that belong to that site probably contain the phrase "*civil right movement*". As a consequence, the content of such index node should contain most documents that belong to that site. In this section, we will focus on computing the two impacting factors, in the document and in the site (domain), for computing the documents topics.

1. **Document-Topic Weighting**

   This step checks the phrase distribution in the documents contents using an online "*HTTP request*"[3]. If the phrase is available in the correct distribution, the weight of document is computed using the cosine similarity between two vectors, the first vector represents the phrase in each node and the second vector represents the document's content. The result is stored in the memory as: [Document #, url, phrase-id, score].

2. **Site-Topic Weighting**

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

[3] http://msdn.microsoft.com/en-us/library/system.web.httprequest.aspx

We applied a top-down traversal algorithm[4] for computing the weights of the sub-trees in each site. To make the content of the index nodes optimal for the number of documents, our system used an automatic cut-off value. The cut-off value is changeable and relies on the number of documents in each sub-tree; this means the value is low if the sub-tree holds a few numbers of indexed documents (children) and vice versa; however, the sub-trees that have little contributions are cut off from their parents. If the index node contains a large number of documents, the cut-off starts at 1 and grows up whenever the number of remaining documents in the final list is higher than 200. We chose a threshold of 200 to keep a balance between the precision and recall values. However, the remaining tree is compressed and bounded to a new tree.

After reducing the size of the tree, the total weight of each document is recomputed by using the following formula*:*

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

## 3 QUERY PROCESSING

Query processing is an essential step for any search engine. The query processor is the portion of server that accepts, parses, and executes the query syntax. A query processor has the following duties:

- Query goals (detecting the type of query).
- Distribute the queries over indexes.
- Query execution.
- Query optimization (query expansion and normalization).

In our experiments, we categorized queries into four types (each type processed by specific index classes):

- **Title**: this means that relevant pages contain all the query terms in core positions, as full key-phrases, e.g.., “Ron Howard" or "Sore Throat".
- **Domain**: this means that relevant documents are located in a particular site or domain, e.g., “University of Phoenix” or "Churchill Downs".

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

- **Occurrence:** this means that relevant documents are weighted using the occurrence of query terms in documents content, e.g., “Fibromyalgia" or "Lipoma".
- **Combining:** this type of query is processed using primitive keywords from urls and/or titles that imply important weights in the documents content, e.g., "gs pay rate" or "brooks brother’s clearance".

The first step of the query processor is looking for the query in the cached results available in the index repository. If such result is not available, the query will be forwarded to a pertinent index class. The forwarding relies on the class priority. Our system uses the following priorities for detecting the pertinent index class for each type of query:

- If the query length is one word, searching will occur in the home-page index and the dictionary-based index, because one word queries often refer to the home pages, e.g., “arkansas”, or refer to the terms that are most frequent in document’s contents, regardless the documents are home pages or not, e.g., “grilling”.
- If the query length is two words or more, searching will occur in four index classes:
  - The home-page index class (domain name), e.g., the queries “churchill downs”, "quit smoking", and “newyork hotels". The pre-processing removes the spaces before searching, i.e., www.churchilldowns.com, “www.newyorkhotel.com”, or inserts dashes, i.e., “www.quit-smoking.com” or "www.newyork-hotel.com", respectively. A simple form of stemming is used, too. The home-page index class (Wikipedia external links) holds the home pages for other queries, too, e.g., “california franchise tax board”.
  - The term-combination-index class, e.g., a query like "becoming a paralegal".
  - The title-based index class, e.g., "old coins" in "www.ancientcoins.ca", or a query like “gs pay rate" in “www.gspay.com”; in which the term "rate" was extracted from the content; or it refers to the site www.opm.gov/oca/pay/; in which the terms "gs" and "rate" were processed from the content. Another example, for the query "brooks brothers clearance", refers to the site "brooksbrothers.com"; the term "clearance" was processed in the content.
  - The dictionary-based index class, e.g., query like "black history", "septic system design", "dogs clean up bags" or "furniture for small spaces".

# 4 QUERY EXPANSION

The previous section introduced a wide range of phrasal indexing mechanisms and established that phrasal indexing has the potential to improve retrieval effectiveness. Now we focus on query expansion, User's feedback is utilized implicitly by computing the behaviour of some users who adapted the preferences in the dynamic properties of Wikipedia collection. We used two algorithms for query expansion: the shared-links and the manner of titling similar articles.

## 4.1 Using Shared-Links

Wikipedia articles could expand the current articles to other articles by using shared links. Usually, the target articles also point backward to the source articles. We assume that if an article (A) has a link that points to an article (B), and the article (B) has a link that points backward to the article (A), the two articles A and B are related. Therefore, our system gathered all links for each article and built a hash-table in which the articles names represent the keys of table and the gathered links are stored in the content of subsequent keys.

## 4.2 Using Titling Variation Aspect

As we mentioned earlier, Wikipedia articles often use term variants. Some similar articles have different titles, and these variants should be used to expand queries that match any title of article. For example, an article "lipoma" is titled by Wikipedia writers as: "fatty tumor", "fatty lipoma", "lypoma", "lipom at ousneoplasm", "lipomas", and "lipomatosis". These variations are gathered by our system during the indexing of the Wikipedia articles; that is, each title in the Wikipedia is a key in the dictionary, whilst the other titles were stored in the content of that key.

# 5 EXPERIMENTAL RESULTS

We submitted the results of our model for two tasks, the adhoc and the diversity tasks of the web track. For some queries, our precision was zero because many documents retrieved for those queries were ranked as spam in the TREC relevance judgments. The spam detection for the TREC collection was done automatically by using the University of Waterloo IR system [12]. We believe that some of the documents that are ranked as spam in the TREC collection are not really spam.

We present our results in comparison to the best results over all the 48 runs submitted by all the participants for the test queries (50). Despite the fact that we participated in TREC using only the subset B of the very large web collection, we compare with all the submissions. TREC uses general criterion for making comparison based on ERR values for all participants, regardless if the group used the subset A or B. The tables 1 and 2 below show our position in the top 8 runs in TREC 2012, according to the track`s overview paper [13]. If comparing only with the systems that used the subset B, we obtained the best results. If comparing with all systems, we were in the third position for the ad-hoc task and in second position for the diversity task. Figure 1 shows more details for the results of our system.

| 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 1: Top adhoc task results ordered by ERR@20 for the best runs over 48 runs [13]

| Group | Run | Cat | Type | ERR-IA@20 | $\alpha$-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 2: Top diversity task results ordered by ERR-IA@20 for the best runs over 48 runs [13]

| Summary Statistics | |
|---|---|
| Run ID: | DFalah121 |
| Task : | Diversity and adhoc |
| Run type : | automatic |
| Document collection category: | B |
| External resources used: | no additional resources |
| Number of topics: | 50 |

| Adhoc measures | | Diversity measures | |
|---|---|---|---|
| Retrieved | 13109 | $\alpha$-nDCG@10 | 0.4910 |
| Relevant | 3523 | $\alpha$-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 |

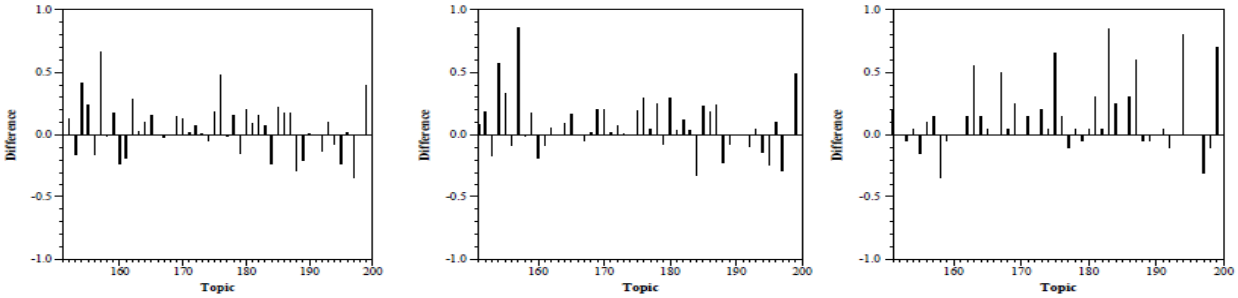


Figure 2: Web Track 2012 diversity and adhoc summary results for our system

## 6 CONCLUSIONS

This paper described our system for the TREC 2012 web track. We explained how the query volume and phrase index can efficiently support the finding of relevant results with low disk overhead. Our method used our own custom indexing and ranking model based on topics identification in the documents' content through the investigation of query structures. This model provides a variety of analytic capabilities, including: phrase extraction, concept correlation, Web page topic finding, topics classification, documents grouping, and document/site weighting. This method is more sophisticated than our previous methods and more robust for processing all types of queries. We addressed some drawbacks of our methods from 2010 and 2011; for example, we kept the stop words in the key-phrase index. This allowed us to successfully process queries that contain stopwords, such as "becoming a paralegal" or "furniture for small spaces", by breaking him into phrases and giving them equal weights since they have the same importance in the query.

## REFERENCES


[1] Hodong Li. "An Inverted Index Generator for CINDI", Master's Thesis, Computer Science, Concordia University, Canada, 2003.

[2] Magnus Sigurðsson Søren, Christian Halling, "Zeeker: A topic-based search engine", Master of Science in Engineering, the Technical University of Denmark (DTU), 2007.

[3] D. Bahle, H. Williams, and J. Zobel. "Compaction techniques for nextword indexes". In Proc. 8th International Symposium on String Processing and Information Retrieval (SPIRE 2001), pages 33-45, San Rafael, Chile, 2001.

[4] Dirk Bahle, Hugh E. Williams Justin Zobel. "Efficient Phrase Querying with an Auxiliary Index", School of Computer Science and Information Technology, RMIT University, Melbourne, Australia, 2001. The 28th European Conference on IR, ECIR 2006.

[5] Matthew Chang and Chung Keung Poon. "Efficient Phrase Querying with Common Phrase Index", Dep. of Computer Science, City U. of Hong Kong, China. Springer - Verlag 2006.

[6] Hodong Li. "An Inverted Index Generator for CINDI", Master's Thesis, Computer Science, Concordia University, Canada, 2003.

[7] Xu Chen, ZeyingPeng, Jianguo Wang, XiaomingYu,Yue Liu, HongboXu, Xueqi Cheng, "ICTNET at Web Track 2011 Ad-hoc Task", Institute of Computing Technology, Chinese Academy of Sciences, Beijing, Graduate School of Chinese Academy of Sciences, Beijing, TREC Proceedings 2011.

[8] Trystan Upstill, Nick Craswell, David Hawking, "Query-independent evidence in home page finding", Australian National University, Canberra, Australia, CSIRO Mathematical and Information Sciences, Canberra, Australia, 2003.

[9] Eda Baykan, Monika Henzinger, Ludmila Marian, Ingmar Weber, "Purely-based Topic Classification", 18th International World Wide Web Conference, 2009.

[10] Ruihua Song, GuomaoXin, Shuming Shi, Ji-Rong Wen, Wei-Ying Ma, "Exploring URL Hit Priors for Web Search", Microsoft Research Asia, Beijing China. The 28th European Conference on IR, ECIR 2006.

[11] R. Kaptein, M. Koolen, J. Kamps, Result Diversity and Entity Ranking Experiments: Anchors, Links, Text and Wikipedia, University of Amsterdam. TREC Proceedings 2010.

[12] G. Cormack, M. Smucker, and C. Clarke. "Efficient and effective spam filtering and re-ranking for large web datasets", University of Waterloo. arXiv:1004.5168, 2010.

[13] Charles L. A. Clarke, Nick Craswell, and Ellen M. Voorhees. "Overview of the TREC 2012 Web Track", TREC working notes, 2012.