Ranking Web Pages Using Collective Knowledge

Falah H. Al-akashi Diana Inkpen
falak081@uottawa.ca 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. Indexing can be performed based on words or on phrases. Our approach aims to efficiently index web documents by employing a hybrid technique in which web documents are indexed in such a way that knowledge available in the Wikipedia and in meta-content is efficiently used. Our preliminary experiments on the TREC dataset have shown that our indexing scheme is a robust and efficient method for both indexing and for retrieving relevant web pages. We ranked term queries in different ways, depending if they were found in Wikipedia pages or not. This paper presents our preliminary algorithm and experiments for the ad-hoc and diversity tasks of the TREC 2011 Web track. We ran our system on the subset B (50 million web documents) from the ClueWeb09 dataset.

Categories and Subject Description

Web Information Retrieval: Content Analysis, Indexing, and Ranking

General Terms

TREC Participation – Web Track

Keywords

Wikipedia, Grid Indexing Structure, Web Indexing & Ranking, Wikipedia article-based indexing, Collective Knowledge

Introduction

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 [1].

In word-based indexing, single words are used to build an index table and to find a set of relevant pages according to some computations. In order to find key-phrases that are important

for each topic, a set of articles can be assembled from a particular dictionary. This was used recently for document clustering, entity finding, or document classification in small collections of documents, but it has not been used on the large scale webpage indexing.

Moreover, the current approach of word-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 words in common, even if they are on different in topics. Term frequency is important in determining the topic of documents; but this is not case for all documents configurations, because documents may contain different topics located on different parts. Only a few systems consider this aspect, for example by using a sliding window for each topic [2]. Using phrases in the index in addition to words is another way to deal with this problem.

1 Building the Index

In this section, we describe the idea behind our indexing method for handling phrases and concepts (Wikipedia categories in our case). Our technique is robust and allows the retrieval of documents relevant to a specific topic. Our inverted index is structured as a core distributed hash-search tree. Our method overcomes the problems that appear when using hash tables for indexing a very large collection of web documents. Currently, a centralized index leads to slow query response time, unless it is used in a distributed environment e.g., Map-Reduce [3]. Our index, although centralised, allows for fast retrieval.

The idea for an index that stores data in an efficient way is inspired by the block-oriented storage contexts used by file systems. We avoid the rebalancing issues of some indexing trees, e.g., binary trees. Our index uses sub-trees in a fixed interval. Each internal node has sixteen children and the leaf nodes contain the information about the documents relevant to the index term. The depth from the root to each leaf corresponds to a concept in our index, and it is eight nodes for each concept (index term). This means that the subsequent nodes from the root to the leaves can map all vocabulary terms (concepts / categories) from Wikipedia. Each vocabulary term has a fixed interval (8 bits). Each node was assigned one digit in the interval of 16 bits, 1-F; the value of a digit was assigned to each node, depending on the location of the node in the tree and the encoding stage of each vocabulary term. As a result, the volume of our index for all vocabulary terms from Wikipedia is 0xffffffff or 4,294,967,296 nodes. The encoding algorithm that generated the serial number or hash code for each concept or vocabulary term for drawing the path from the root to the leaf used a Cryptographic Hash Algorithm. Each leaf in the index holds a file of all indexed documents regarding a particular topic, and the label (path) should match a query term for that class (leaf). We used tree types of leaves for each concept or vocabulary term; each leaf holds similar types of documents structured as a table of vectors with a certain number of dimensions. The three categories are: a table for all Wikipedia documents, a table for all home pages, and a table for other web documents. An example of index entries is shown in Figure 1.

Figure 1. Example of index entries

1.1 Assembling the Titles of the Nodes

Our index was pre-designed before indexing the collection (Wikipedia documents, home pages, and other documents). The titles of our index nodes use knowledge from Wikipedia articles. Interestingly, Wikipedia is well structured for describing the topics; for instance the abbreviations and bolded terms are almost always important for the content. Technically, not all pages in the Wikipedia are important for indexing, for instance "Disambiguation" and "File" pages; as a consequence, they were removed from our consideration. We also removed all non-textual pages or non-article pages. We extracted the following terms from the remaining Wikipedia articles:

- All abbreviation terms.
- All bold terms or phrases available on the head of the content.
- All phrases on the headers.
- All titles of pages.
- All terms or phrases located between brackets or split by any punctuation characters.
- All phrases and terms split by conjunction words such as "or", "and".

Stop-words were removed from the content in a pre-processing step. Then, we obtained three types or terms: single words or abbreviations (such as "AVP", "Diana"), two-word phrases (such as "Martha Stewart"), and longer key-phrases (such "President United States"). These terms were used initially for building our index nodes and giving them titles accordingly.

1.2 Vector Generation

In this section, we describe our representation for a given document. The most-used representation in information retrieval is based on the vector space model, in which a document is treated as n-dimensional vector, and each dimension represents to a specific type of information. We add more functions in each vector; dimension "i" is assigned to function "i". As we mentioned, we used three types of tables, and each type is organized by a similar number of dimensions:

$T^w = \{D^1, D^2, \dots., D^n\}$; where D^n represents n documents indexed in the Wikipedia-Table, *w*
$T^H = \{D^1, D^2, \dots., D^n\}$; where D^n represents n documents indexed in the Home Page-Table, *H*
$T^O = \{D^1, D^2, \dots., D^n\}$; where D^n represents n documents indexed in the Other-Table; which means all the collection except the Wikipedia articles and home-pages documents.
Each leaf in the index is a file that holds three such tables; its name is selected from the names of Wikipedia articles. A few operations were performed on the names of the articles before using them for giving titles to our index nodes or leaves; for instance all stop-words, symbols, and, numbers were removed.

2 Indexing the Repository

As we mentioned, we categorized the documents in our repository index into three types: Wikipedia articles, home pages, and other documents. Each type holds a particular type of document. In this section we will describe the indexing algorithm that we used for each category.

2.1 Wikipedia Repository Indexing

Wikipedia is a collective knowledge source of 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 article has a unique vocabulary name (identifier or ID); sometimes Wikipedia uses different faceted vocabulary terms to describe the same article. We extracted the important information (key-phrases) from each article and used it as setup information to rank each leaf node in our customized index. Our system scanned through the whole Wikipedia repository (the version that is included in the ClueWeb collection) and computed the following normalized vectors for each document:

- Words-occurrences (frequencies); we assigned a threshold value to exclude all low frequency terms and to keep the ones with frequency higher than the threshold.
- Outgoing-links occurrence frequencies; outgoing-links are all URLs that point internally to other Wikipedia articles. For us, if an article points to other articles frequently, at more than one position in the content, then all these articles are related or similar in topic.

- All external URLs.

HTML markups, ads, navigation links, and stop-words were removed. As a result, the outputs were transformed and represented as vectors of the following structure:

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

where D^{URL} is a document url, D^{ID} is a TREC identifier (TREC-ID), D^{lf} is represents the outgoing-link frequencies (for the links that are repeated frequently 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 Wikipedia repository are transformed into several tables of vectors available in the leaves of the index.

2.2 Home Pages Indexing

The use of page content for home-page finding is problematic for several reasons. Often the first page in a site is a home page which mostly contains navigational links for sitemap. Some potentially useful evidence for home page finding is query-dependent. This includes the presence of query words in the document's text, which is referring to anchor texts, or in the document's URL. It is known that full-text relevance ranking is not particularly effective for home page finding [4]. Other potentially useful evidence is query-independent. This was demonstrated in the TREC-2001 home page finding task [4]. The best run was submitted by Westerveld et al. from UTwente/TNO [5] and used the pages URLs as evidence. Generally, query-dependent and query-independent are not the case for all situations of home-page finding, because URLs sometimes use shortcuts or abbreviation terms to represent some underlying meanings beyond the domain name, e.g., "nist.com". We used different representations when we processed URLs for home-pages. We used two basic methods for the home-page finding task: the first method is standard and it uses the structure of URL names, and the second method is suitable for most abbreviated and embedded terms.

2.2.1 Processing URL Structures

According to the general structure of URLs, we classified them into five categories (after stripping off all the symbols, numbers, and all the trailing terms such as index, default, and welcome), depending on the location of the term in the URL:

- If a term is located in the main domain section, e.g., "www.diana.com/".
- If a term is mixed or embedded with other terms, such as Air France in "airfrance.ca".
- If a term is represented as a shortcut or an abbreviation, e.g., "www.uottawa.ca".
- If a term is located in the sub-domain section, e.g., "trec.nist.com/".
- Any an URL was ended at document's name and preceded by a symbol "~", e.g., "www.uottawa.ca/~cadams".

Thus, we have only five possibly evidences in the URL forms. We used different scores for each status; we assigned the ranking value "1", "2", or "3" for a term that is located on the main-domain, sub-domain, and document-name, respectively.

Home-page finding methods require finding equivalent pages by converting hyperlinks might 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", "http://www.bmo.com/ (language code)/ index.html", should be all represented as "www.bmo.com/".

2.2.2 Embedded Keyword Extraction

URL keywords or terminology extraction is a challenging task. Researchers employed different algorithms, such as a statistical "n-grams" [5] or natural language processing methods for tokenizing and analyzing URL data to extract keywords that can be utilized to index content. Besides web page popularity, we exploit using query log files assembled by Alexa.com[1] for finding out the original keywords behind the embedded keywords in domain names. Alexa.com computes traffic for all popular search engines. We used the *tf* method for computing the occurrence of frequent queries in log file. The log file contains the important queries for each site accessed by users; for instance "airfrance", "uottawa", and "nist" are defined in log file as queries: "Air France", "University of Ottawa", and "National Institute of Standards and Technology", respectively.

Our method was tested for home page finding for the TREC 2011 web track topics. For example the query "jax chemical company" involved retrieving all home pages available in the corpus; therefore our system was obtained a precision p@5=1.0 and p@10=1.0 for this query.

2.3 Other Collection Indexing

Usually, web pages are indexed using their content, but not all pages are useful for indexing their content, for instance multimedia pages may contain videos, sound, or images; home-pages sometimes contain little text; other web pages may contain topical content such as programming code or navigational links which are useless for indexing. Generally, we used two types of index structures: word based index and key-phrase based index.

2.3.1 Word Based Index

Often the meaning of a document is conveyed by words located in meta-content (such as URLs, titles, and headers). Normally, there is at least one term shared between meta-content and the document content. If we can represent the shared word by a short vector and the document content by another vector, it is possible compute the similarity and the impact of that term in the document. Basically, meta-content is available in three fields ("title", "headers", and "URL") and it is necessary to manipulate all these fields together, because (i) we assume that not all documents contain important terms in alone of these fields (ii) usually keywords in the title, headers, and URL are complementary to each other. If the header h1 is not available or it is similar to the title, we chose "h2" or "h3". A very short meta-text may not contain enough information and a long text may contain unnecessary or redundant information. Also, it is necessary to index the main content in order to provide a comprehensive indexing. We use meta-keywords to trigger the document's topic, and then to go inside the index for other query terms.

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

Two documents from different sites might have meta-content with different impact, even they have similar meta-content. Documents whose content has higher similarity to its meta-content should be judged more relevant. Our model was modeled to measure the closeness of any document content to its meta-content, by computing the cosine similarity between the document content and the meta-content, with the cosine similarity between two vectors [8]:

$$SC(D, M) = \frac{\sum_{j=1}^{t} W_j M_j}{\sqrt{\sum_{j=1}^{t} (W_j)^2 \sum_{j=1}^{t} (M_j)^2}} \qquad (1)$$

where W_j is the weight of each term j in the document, M_j is a weight of each term j in the meta-content (the value of each meta term is equal to the weight of that term in the document, if it is available; otherwise the value is 0), t is the number of document terms.

In fact, the cosine similarity was used individually for each term in the meta content; therefore each term in the meta content has one similarity value with the document. To compute the similarity measure for each term in meta-content, we processed the document content as follows:

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

Once the *tf* value of each term was computed, we used equation (2) for computing the impact of each term from the meta-content. Each meta-term is assigned its cosine similarity measure with the document content. To determine which meta-term is significant, we use a specific threshold value to choose the best terms and ignore others. On other hand, as we mentioned before, not all terms in the document are available in the meta-content; likewise it is rare to find all the query terms located in meta-content. Therefore, it is impractical to rank web documents only by their meta-contents; we need to add the term frequencies in the content of the document. To reduce the storage needs, we ignored all the terms that occurred only once. Each term in meta-content has a vector with a certain dimension; including the cosine measure, and the significant terms frequencies, in addition to the "docID" and the "termID". Document relevance is computed by adding the cosine similarity for the first query term which is available in the meta-content; otherwise, only terms frequencies are computed. Sometimes, queries contain digits when looking for more precise results, e.g., "hp mini 2140"; therefore we address this issue by adding one extra dimension to the vector to yield document's title. Hence, the meta-content of "n" terms is broken down to "n" vectors; and then each vector is transmitted to a corresponding node in the index, as shown below:

{docID}{TermID}{SC (Term)}{<t1, f><t2, f><t3, f>… <tn,f>}{Title}

2.3.2 Key-Phrase Based Index

Our method does not employ computations for the terms that occurred only once; but they can still be used for phrasal queries. Some documents are based on a fixed interval of sequence

terms; these terms could 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; for instance the query "map of brazil" is sometimes located once at one position in the document; hence terms occurrences are not important for the document's relevance. However, our method uses the key-phrase index with the respect of computing terms frequencies for all the terms in the content. For example, the query "Martha Stewart and imclone" requires to compute the proximity search for all terms; computing the term frequency for the term "imclone" and the key-phrase frequency for the phrase "Martha Stewart" is important for computing document's relevance. To compute the key-phrase frequency, first we strip off all stop-words, symbols, characters, and single letters from the document content. Next, we compute the frequency of single terms, double contiguous terms, then length three, four, etc., as far as terms occurred together frequently. Then, for each key-phase we compose a vector of fixed dimension, as shown below:

{docID}{Key-phraseID, f}{<t1,f><t2,f><t3,f>….<tn,f>}{Title}

where key-phraseID is a hash key that is generated using the same algorithm that generated the hash keys for each node in our index.

3 Enhancing the Ranking Results

Traditional methods for ranking documents are not optimal in terms of search engine optimization (SEO). Smoothing ranked list and changing documents positions in our search results was used based on a number of factors designed to provide end-users with helpful and accurate search results. In our method, we used two strategies based on human references to improve our rankings.

3.1 Using alexa.com

With the massive amount of data available on the web, not all data are reliable and valuable. There are a lot of sites with untruth full content that might be ranked high by our model. Navigational queries, for example, are always looking for reliable and valuable information; this information is usually available in sites that can be trusted. Summarizing, site reputation, ranked locally and globally, is important in our relevancy algorithm. We used this factor for enhancing our ranking algorithm by filtering out all the poor sites. We exploited the information from www.alexa.com by assembling all reputation values for the main domains in our corpus.

3.2 Using Wikipedia

As we mentioned earlier in this paper, documents that are classified as home pages or documents that are important articles in the Wikipedia might change their rank in our final ranking list. Human references are robust arguments to bias the ranking towards some documents. Since we previously indexed all the important external references to Wikipedia articles, the ranking algorithm will make a match between the ranked list and the archived indexed documents that existed in each node for the search query. As a result, the matching documents will get higher positions in the final list.

4 Query Expansion

Search engines use query expansion to increase the quality of the search results. It is assumed that users do not always formulate search queries using the best terms. The goal of query expansion in to increase recall, without decreasing too much the precision, by including in the result pages which are more relevant (higher quality), or at least equally relevant. By ranking the occurrences of both the user entered words and synonyms and alternate morphological forms, documents with a higher density (high frequency and close proximity) tend to migrate higher up in the search results, leading to a higher quality of the 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. However, to improve retrieval performance in our system, we used query expansion for those queries that were classified as Wikipedia articles; anchor terms or phrases that frequently occurred in each article were used to expand the query topic (anchor terms have been indexed previously); for instance, the topic "all men are created equal" that ranked our system as high precision P@5=0.8 and P@10=0.7, because the query was expanded with the phrases "Gettysburg Address" and "Declaration of Independence", and in this way the system succeeded to retrieve document that did not contain the query term but they were highly relevant.

5 Query Processing

Query processing is an important processing step and it includes: detecting the type of the query, query normalization and query expansion. Basically, we have five types of queries according to the: title, domain, frequency.

- **Title**: this means that relevant pages contain all query terms in core positions, as full keyphrases, e.g. "arkadelphia health club" or "map of brazil".
- **Domain**: this means that relevant documents are located in a particular site or domain, e.g., "jax chemical company".
- **Occurrence:** this means that relevant documents were judged using the occurrences of query terms in the document, e.g., "fact of uranus" or "Martha stewart and imclone".

A few examples of queries and how they are processed in our system are shown in Figure 1. Each query is processed using the index with three types of leaves. We use overlapped term positions besides the priority factor for each term in the query. Our system uses the following criteria for processing a query:

- If the query length is one term, searching is done in two indexes: the home-page index and the word-based index, because a one-term query could look for a home page, e.g., "uottawa", or, the term could be frequent in a document's content, regardless if it is a home page or not, e.g., "afganistan".
- If the query length is two or three terms, searching occurs in three indexes: the home-index, the word-based index, and the key-phrase based index, e.g., "Map of Brazil" or "Ralph Owen Brewster".

- If the query length is four or more, searching occurs in two indexes: the word-based and the key-phrase based index, e.g., “Ritz Carlton Lake Las Vegas”. In the case of keyphrase-based index, the query would be searched as: “Ritz Carlton Lake Las Vegas”, “Ritz Carlton lake Las”, and “Ritz Carlton Lake”; whereas in case of the word-based index, first, a node “Ritz” is located and the system goes through each document vector to find the other query terms. Next, a node “Carlton” is located and then the system walks through each document vector to finding other query terms; and so on, regarding other terms query. The results from all the search situations are aggregated in one list, without duplication.
- If the query involves a prepositional or conjunctional term, then the first term and the single term that occurred before or after the conjunction is weight more than other terms in the query, for example the terms “uplift” and “Yellowstone” on the query “uplift at Yellowstone national park”; or a term “french” and “casino” in a query “french lick resort and casino”.

6 Document Ranking

In this section, we explain our custom model for ranking webpages which uses an extended for of the cosine measure similarity. As we said, not all query terms have equal impact or weight. For each query, there is one term has more impact than others; for instance the query, **“*Martha stewart and imclone*”** is focused on a term “imclone” more than on the other terms. Describing our method for finding the important or prominent term in each query requires more details which are out the scope of this paper. However, the following equation is used to rank each document regarding the above query, for example.

$$\text{Rank } (D_i,Q) = SC(Di, imclone) + \left(\frac{\sum_{j=1}^{t} W_{j\,i}}{100}\right)$$

where $SC(Di, imclone)$ is the cosine similarity (equation 1 above) for query term “imlcone” in document D_i, W_{ji} is the weight of query term j in document i; and, t is the number of query terms. We added the sum of all the weights of query terms not only the term “imclone” is important in the query, but other terms are also important. That is, we used the cosine similarity for the term that has more impact than others in the query, plus the sum of the weights if all the query terms (the value is divided by 100 is to find the percentage value). Finally, our system biases the final ranking list by the site’s reputation and the Wikipedia preferences.

7 Spam Documents Filtering

The ClueWeb09 collection contains a lot of spam documents. We filtered out spam documents that would hurt the quality of our retrieval. Cormack et al. [7] studied the spam filtering in the “ClueWeb” collection and showed that the spam filtering could significantly improve the performance of a system. Therefore, computing term frequency and cosine term similarity in our system could detect spam documents that use many junk words that affect the impact of each term (the cosine term similarity in our method). We assigned a threshold value of total term

frequencies for each document retrieved, as well as a threshold value for cosine term similarity. If the cosine term frequency is lower than the threshold, the document is considered spam. On the other side, junk documents hurt the ranking list, too, if they are appear. The junk documents may contain only a little information and scattered words. Therefore, we used another specific threshold value for removing these types of documents.

8 Experimental Results

We submitted one run for the adhoc and the diversity tasks of the web track. Our run was based on the collection of document Category B of the ClueWeb09 corpus (50 million documents). For some queries, the precision was zero because relevance judgements contained only documents selected from the other part of the collection (Category A). For other queries, our method was not able to model the topics of the queries. Table 1 and 2 list the results of different metrics for both the diversity and the adhoc metrics as average for the 50 test queries.

Run	αnDCG@5	αnDCG@10	αnDCG@20	P-IA@5	P-IA@10	P-IA@20
DFalah11	0.4418	0.4727	0.5020	0.2963	0.2546	0.2198
Run	NRBP	MAP-IA@20	ERR-IA@20	strec@20	--	--
DFalah11	0.3699	0.0726	0.4058	0.7370	--	--

Table 1: Diversity task results

Run	NDCG@20	ERR@20	P@5	P@10	P@20	MAP
DFalah11	0.2044	0.1219	0.3320	0.2960	0.2750	0.0794

Table 2: Adhoc task results

9 Comparison to other systems

Our submitted run was considered only for the ad-hoc task. We can compare our result with other systems that worked with the subset of the data collection named category B. It is not fair to compare to the systems that used the whole data collection (category A), because some documents that were in the excepted solution could not be retrieved by our system since they were not in the reduced dataset. Table 8 presents the comparative results, according to the track's overview paper [9].

Group	Run	Cat	ERR@20	nDCG@20	P@20	MAP
N-A	srchvrs11b	B	0.131	0.233	0.298	0.110
Univ. of Ottawa	DFalah11	B	0.122	0.204	0.275	0.079
Univ. of Amsterdam (Kamps)	UAmsM705tiLS	B	0.119	0.202	0.273	0.085

Table 3: Comparison of our system with other systems from Web Track 2011.

10 Conclusion

Overall, our method used our own custom indexing and ranking model based on Wikipedia knowledge. This model provides a variety of analytic capabilities, including: concept extraction, concept correlation, text summarization, spam filtering, and term to document similarity.

We addressed some drawbacks of our 2010 system. Now we kept stop words in the key-phrase index. This allowed us to successfully process queries such as "to be or not to be, that is the question". The conjunctions and prepositions also allowed us to separate important terms in some queries, e.g., for the queries "Martha Stewart and imclone" and "earn money at home", the important terms are "imclone" and "home", respectively. Our results in 2011 are improved when compared to our results from last year; despite the fact that many topics of this year are underspecified (i.e., multi-faceted) and traditional Web search engines have difficulty with queries of these types, as mentioned on the track's webpage[2].

REFERENCES

[1] Xiangji Huang, Damon Sotoudeh-Hosseini, Hashmat Rohian and Xiangdong An, "York University at TREC 2007: Genomics Track", School of Information Technology, York University, Toronto, Ontario, Canada, Department of Computer Science & Engineering, York University, Toronto, Ontario, Canada, 2007.

[2] Xu Chen, Zeying Peng, Jianguo Wang, Xiaoming Yu,Yue Liu, Hongbo Xu, Xueqi Cheng, "ICTNET at Web Track 2010 Ad-hoc Task", Institute of Computing Technology, Chinese Academy of Sciences, Beijing, Graduate School of Chinese Academy of Sciences, Beijing, 2010.

[3] Djoerd Hiemstra and Claudia Hau, "MapReduce for Experimental Search", University of Twente, 2010.

[4] N. Craswell, D. Hawking, and S. Robertson. 2001. Effective site finding using link anchor information. In Proceedings of ACM SIGIR'01 (New Orleans, LA). 250–257, 2010.

[5] Nick Craswell and David Hawking, "Query-Independent Evidence in Home Page Finding", Trystan Upstill, Australian National University and CSIRO Mathematical and Information Sciences, 2010.

[6] Eda Baykan, Monika Henzinger, Ludmila Marian, lngmar Weber, "Purely URL-based Topic Classification", Ecole Polytechnique, Google, Lausanne, Switzerland. 2009.

[2] http://plg.uwaterloo.ca/~trecweb/2011.html

[7] G. V. Cormack, M. D. Smucker, and C. L. A. Clarke. “Efficient and effective spam filtering and re-ranking for large web datasets”, April 2010.

[8] David A. Grossman, Ophir Frieder, “Information Retrieval Algorithms and Heuristics”, Second Edition, Illinois Institute of Technology, Chicago, IL, USA, 2004 Springer.

[9] C.L.A. Clarke, N. Craswell, I. Soboroff, and E.M. Voorhees, NIST, Overview of the TREC 2011 Web Track. In Proceedings of TREC 2011. The Twentieth Text Retrieval Conference.