# Design and Implementation of a Search Engine using Vector Space and PageRank

[CSCE 5200 Final Project Report] [*]

Guangchun Cheng
Department of Computer Science and Engineering
3940 N. Elm, Room F231
Denton, TX 76207-7102, USA
guangchuncheng@my.unt.edu

## ABSTRACT

We designed and implemented a web search engine based on the combination of vector space model and PageRank. The system indexed given number of web pages within a domain, the inverted index of these web pages was constructed, and their PageRank values scores computed. A new approach was proposed to combine the similarity value between each document and query and the PageRank scores. For any given query, the candidate web pages were retrieved using vector-space model, and sorted based on a non-linear combination of cosine similarity from vector-space model and PageRank scores.

The system was designed and developed with highly reusable modules, consisting of crawler, retrieval, PageRank and utility among others. An web interface was created to accept user queries and to show the search results within the domain of University of North Texas (unt.edu). Experiments were done to verify the effectiveness of the system.

## Categories and Subject Descriptors

H.3 [**Information Storage and Retrieval**]: Information Search and Retrieval; E.5 [**Data**]: Files—*Sorting/searching*

## General Terms

Theory

## Keywords

Web search, search engine, vector space, PageRank

## 1. INTRODUCTION

Search engines such as Google and Yahoo are nowadays becoming necessity tools for information retrieval, learning and many other aspects of life. In this project, a web search engine was designed to crawl and index part of unt.edu web pages, based on which relevant web pages can be retrieved when given a query.

While different search engines may have different search models and principles, vector space model [4] and PageRank [2] are commonly used by most popular ones, and they can be combined together to improve the search performance. Vector space model (or term vector model) is an algebraic model for representing text documents as vectors of indexed term weights, and the relevance of two documents, such as query and candidate documents, is computed as the similarity between two vectors. Different from vector space model as a content-based approach, PageRank was developed by Google to express the links or structure of the web, which is independent of the queries. The simple idea behind is that the most popularly linked web pages have higher creditability. One drawback of simple PageRank is due to its query-independent property. The search engine design for this project combines vector space model and PageRank in order for better performance.

### 1.1 Related Work

Automatic information retrieval and web search engine are relatively new areas, and attract much attention especially due to the information explosion in the past several decades. A well-known information retrieval model was vector space model, firstly presented by Gerarl Salton *et al.* [4] in 1975. This model is still a popular model used as a basis of many modern search engines. It combines the local term frequency (*tf*) with global document frequency (*df*) to measure each term's capacity to distinguish a document from others.

While vector space model is successful, it has some drawbacks. One of them is that it did not take the quality of information sources into consideration. Brin and Page [2] discovered the importance of hyperlinks in web pages, and proposed a new algorithm call PageRank, which is now used, together with other technologies, by popular web search engines such as Google.

To combine the results from these two models, some suggested use linear combination [1]. In the project, a different approach was used.

[*]The system is available at `http://students.csci.unt.edu/~gc0115/`.

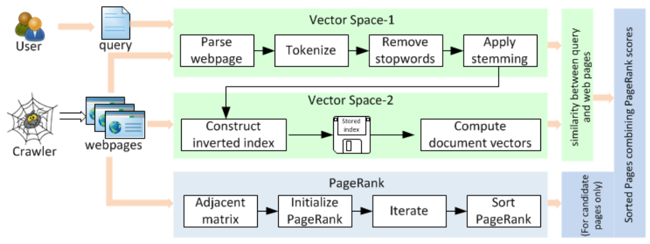


**Figure 1: Overview of the system**

### 1.2 System Overview

The system mainly consists of two parts which are the search engine and the web-based user interface. The search engine sorts the indexed web pages according to the combined scores from vector-space model and PageRank for given queries, and the sorted results (URLs and titles of web pages) were presented in a web-based interface based on common gateway interface (CGI). The framework of the system is shown in Fig. 1 .

Based on this framework, the rest of this paper is organized in the following way. Section 2 introduces the main components in vector-space model and how the challenges were resolved. Section 3 discussed the intelligent part of this work: an implementation of PageRank and how it is combined with vector-space to adjust the search results. In section 4, experiments results are shown to demonstrate the effectiveness of the system. Results with and without PageRank were shown, from which we can see the incorporation of PageRank improves the performance of the system. Section 5 concludes this project and shows the consideration for future work. In the following section, we use "document" and "web page" interchangably without misunderstanding.

## 2. VECTOR SPACE BASED RETRIEVAL

The use of vector space model for information retrieval can be divided into three steps. The first step is to index each web page, including parsing web pages (such as remove HTML tags), tokenizing the terms, and potentially stopwords removal and stemming. The second part is to construct the vector representation for each document. Many strategies may be employed, and the most popular one is TF-IDF as shown in Salton *et al.* paper [4]. To facilitate the computation of TF-IDF, an inverted index of the web pages are usually constructed first, and this inverted index structure is serialized for efficiency purpose. The last step involves computing the similarity between user-provided query (deemed as short document) and each of the candidate web pages. We will describe the design of each part in the following sections.

### 2.1 Web Pages Indexing

Web page indexing is to establish an easy structure to represent such information as web pages. It starts with the pre-processing to the web pages, which includes web page parsing, tokenization, stopword removal and stemming. Following that is to construct an inverted index to compactly represent the web page collection. Using such inverted index, a term vector representation of each page can be easily formalised.

*Pre-processing*

The **acquisition** of web pages is the first step. We designed a web crawler to collect given number of web pages within specified domain (unt.edu for example). The crawler collects web page using a breadth-first-search (BFS) strategy. It starts with a starting URL as the root node, and follows the hyperlink to create the search tree. The prototype of the crawler Crawler::gCrawler() is

**Input**: startURL:starting URL
restrictiveness:restricted domain
numPages: total number of page to retrieve
fURLs: file to store URLs
fPages: folder to store webpages
fIdUrl: file to store Id-URL
**Output**: %URLs: Hash of URLs {URL=>parent}

The next step is to **extract** suitable information in each downloaded web page. Although the HTML formats (tags and their properties) are sometimes valuable, we only keep the content in each web page, i.e. we remove all the HTML tags using regular express supplied by Perl. The links in the anchor tag ("<a>") will be used for PageRank analysis in section 3.

Each web page was then **tokenized** with separators. We firstly replace potential separators with white space, and then split the text using white space. For simplicity, the potential separators used in this work are punctuation symbols which are not leaded or followed by digital numbers. For example, a document "The population of China is 1.3 billion, 20% of the world's" was tokenized as 12 tokens {The, population, of, China, is, 1.3, billion, 20%, of, the, world, s}. It would probably produce better result by removing single characters such as "s" in this example, but they were not removed for the evaluation in this report.

By default, the system removes the **stopwords** and applies **stemming** to each token. The stop words used were from a list of common words available at `http://www.cse.unt.edu/~rada/CSCE5200/Resources/common_words`. These words commonly exist among all document so that they do not have good discrimination for web page retrieval. However, it can greatly reduce the computational cost by removing them due to their large number of occurrences. For the example above, the document will become "population China 1.3 billion 20% world s". Porter Stemmer [**?**] was further applied to each remaining tokens. Stemming recognizes words with the same root as the same one, and can deal with morphological changes to some extent. "Population" will be stemmed to "populat", and "China" to "china".

*Inverted Index*

After the previous pre-processing steps, each web page becomes a sequence of meaningful tokens. The vocabulary for a collection of web pages is the set of unique **terms** from all the tokens. In order for efficient computation of TF-IDF in next section, an inverted index is constructed, as illustrated in Fig. 2. This figure shows the occurrences (web page $D_j$'s) of each indexed vocabulary term. Because this structure is constructed from the viewpoint of index terms and their associated web pages, so it is a **inverted index**. More detailed explanation is in the coming section. The function to construct this inverted index is Re-

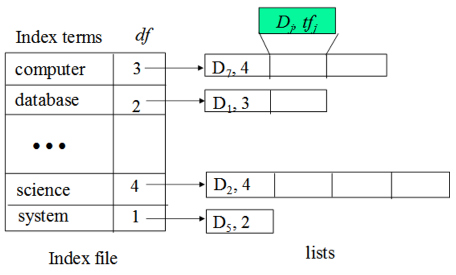


**Figure 2: Schematic diagram of inverted index (http://www.cse.edu/∼rada/CSCE5200/).**

trieval::constructInvertedIndex()

| | | |
|---|---|---|
| **Input**: | dirname:the directory of the corpus | |
| | refMaxTF: | max TF of each document |
| | stemming : | if apply a stemmer(optional) |
| | stopelimnt: | if eliminate stopwords(optional) |
| **Output**: | %indexInverted: | the inverted index |

## 2.2 Vector Representation and TF-IDF weighting

Vector space model treats each web page as a vector of occurrences of terms. Each web page $d_j$can be represented as a vector

$$\mathbf{d_j} = (w_{1j}, w_{2j}, ..., w_{Nj})$$

where $N$ is the size of the vocabulary, and $w_{ij}$ is a measure (term weights) of term $t_i$ in web page $\mathbf{d_j}$ such as **term frequency**. In this work, TF-IDF was used to compute $w_{ij}$ and represent each web page.

In the classic vector space model proposed by Salton *et al.* [4], the term weights in the documents are products of local term frequency (*tf*) and global document frequency (*df*), i.e TF-IDF. The basic idea is that a term existing in most documents has less discrimination capacity then terms only appearing in few documents. For instance, search document using ”the” may not yield desired results. Therefore, the term weight is defined as

$$w_{ij} = tf_{ij} \cdot log\frac{|D|}{df_i}$$

where $tf_{ij}$ is the frequency of term $t_i$ in document $d_j$, $|D|$ is the total number of documents, and $df_i$ is the number of documents containing term $t_i$. $|D|$ can be easily obtained, and $df_i$ and $tf_i$ have already stored in the inverted index as shown in Fig. 2. The procedure to compute the similarity is Retrieval::computeSimilarity()

At this point, each web page has been converted to a numeric vector representation. Note that there are various different weighing sachems, and classic TF-IDF is only one of them. Interested reader can refer to Salton and Buckley's paper [3].

| | | |
|---|---|---|
| **Input**: | refIndex:inverted index | |
| | refLenghtTotal: | document length |
| | refMaxTFs: | max TF of each document |
| | strQuery: | query sequence |
| | termComp: | term frequency weighting |
| | collectComp: | collection frequency weighting |
| | normalize: | normalization or not |
| | termCompQuery: | query term frequency |
| | collectCompQuery: | query collection frequency |
| | stemming: | if apply a stemmer(optional) |
| | stopelimnt: | if eliminate stopwords(optional) |
| **Output**: | %indexInverted: | the inverted index |

## 2.3 Similarity Computation

For any given query $q$, we treat it as a short document (web page),

$$\mathbf{q} = (w_{1q}, w_{2q}, ..., w_{Nq})$$

and

$$w_{iq} = \left(0.5 + \frac{0.5tf_{iq}}{max\{tf_{ij}\}}\right) \cdot log\frac{|D|}{df_i}$$

In this way, both web pages and query are represented as fixed-length vector, and any similarity metric can be used to sort the web pages. In this project, we use cosine similarity, so the similarity of a web page $\mathbf{d_j}$ and a query $\mathbf{q}$ is defined as

$$S_v(d_j, q) = \frac{\mathbf{d_j} \cdot \mathbf{q}}{||\mathbf{d_j}||\,||\mathbf{q}||}$$

In implementation, only those web pages containing at least one term from the query are considered. Sorting $S_v(d_j, q)$, a list of web pages based on relevance are obtained. Actually this list can be presented to the end users. In this project, however, we combine it with PageRank to utilize the link information in the web page documents.

## 3. INTELLIGENT: COMBINING PAGERANK

Vector space model is a general approach for document retrieval, not designed for hyper-linked set of document such as World Wide Web. One intuition behind link-based information retrieval is *different sources of information have different creditability.* People are inclined to trust authoritative information sources, and the in-link provides sorts of this information.

Among link-based analysis, one of the most popular is PageRank [2], while is widely used by Google and other search engines. We implemented a simple PageRank using random walk mechanism, and then proposed a method to combine the PageRank scores to $S_v(d_j, q)$.

## 3.1 PageRank and Its Implementation

The Web is a directed graph (**webgraph**) with web pages connected by the hyperlinks. We present the webgraph as $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ where $\mathbf{V}$ is a set of web pages and $\mathbf{E}$ the links. PageRank is a probability distribution used to represent the likelihood that a person randomly clicking on links will arrive at any particular page [5]. There are two different perspectives on PageRank computation, and we adopted the numeric computation one. After given initial PageRank score for each page $S_{pr}^{(0)}(V_i)$, the scores are updated iteratively. Define

$$In(V_i) = \text{predecessors of } V_i,$$
$$Out(V_i) = \text{successors of } V_i,$$

then

$$S_{pr}^{(i+1)}(V_i) = (1-d) + d \sum_{j \in In(V_i)} \frac{1}{|Out(V_j)|} S_{pr}^{(i)}(V_j)$$

The webgraph is stored as a 2-dimensional hash table ,and in order to obtain the predecessors and successors more quickly, we separated the webgraph into two hash tables: **dagIN** and **dagOUT**. If page $V_i$ links page $V_j$,

$$\mathbf{dagIN}\{V_i\}\{V_j\} = 1$$

$$\mathbf{dagOUT}\{V_j\}\{V_i\} = 1$$

In this way, in-links and out-links can be easily obtained using hash slicing:

$$In(V_i) = \mathbf{dagIN}\{V_i\},$$
$$Out(V_i) = \mathbf{dagOUT}\{V_i\},$$

The implementation is straightforward following the PageRank algorithm. Finally we obtain the PageRank score of each page, $S_{pr}(d_j)$. Note that it is query-independent.

### 3.2 Combination of PageRank with Vector Space

To combine the results from vector space model and PageRank, an intuitive solution is a linear combination. However, this would favor web pages with high PageRank score even though their content is much less related to user's query than other pages. The idea behind our design is:

*when the pages go less relevant, the impact of PageRank should become less too.*

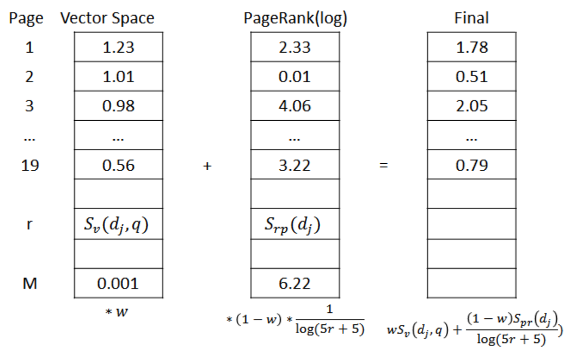
The combination is as follows:

1. Retrieve sorted webpages according to vector space model. The score is denoted as $S_v(d_j, q)$.
2. Obtain corresponding log-PageRank scores, denoted as $S_{pr}(d_j)$.
3. Combine these two scores using the following formula:

$$S(d_j, q) = wS_v(d_j, q) + \frac{(1-w)S_{pr}(d_j)}{log(r(d_j, q)) + \alpha} \quad (1)$$

   where $r(d_j, q)$ is the rank of document $d_j$ from the vector space model, and $\alpha$ is an adjustment parameter ($log5$ in current implementation). See Fig. 3 for an illustration example.

$S(d_j, q)$ was used to sort the candidate web pages from vector space model, and then presented to the online web interface.



| Page | Vector Space | | PageRank(log) | | Final |
|---|---|---|---|---|---|
| 1 | 1.23 | | 2.33 | | 1.78 |
| 2 | 1.01 | | 0.01 | | 0.51 |
| 3 | 0.98 | | 4.06 | | 2.05 |
| ... | ... | | ... | | ... |
| 19 | 0.56 | + | 3.22 | = | 0.79 |
| | | | | | |
| r | $S_v(d_j,q)$ | | $S_{rp}(d_j)$ | | |
| | | | | | |
| M | 0.001 | | 6.22 | | |
| | $*\,w$ | | $*\,(1-w)*\frac{1}{\log(5r+5)}$ | | $wS_v(d_j,q)+\frac{(1-w)S_{pr}(d_j)}{\log(5r+5)})$ |

**Figure 3: Combination of vector space and PageRank.**

## 4. EXPERIMENTAL RESULTS

In this section, we will describe the online interface, and present experimental results on this platform. For test purpose, we collected web pages using the designed crawler, and parsed each web page to construct the inverted index. This procedure should not been executed each time when given a query because it takes time (tens of minutes or hours, depends on the number of web pages). Therefore, for efficiency purpose, we store the hash table of inverted index to disk using a Perl module called *Storable*. The PageRank was also pre-computed, and saved to disk using the same method.

Some strategies can be used to update the system with new or more web pages. In current implementation, the system examines if these stored files exist or not when given a query. If not, system will try to construct them from local collection of web pages. If the web pages are not stored locally, it will start the crawler to download web pages.

When given a query, candidate web pages are retrieved using vector space model, and then their PageRank scores are combined to determine the final rank of all the candidate pages. All candidate pages are shown to user, together with the final score and the page titles. An interface is shown in Fig. 4 and Fig. 5.

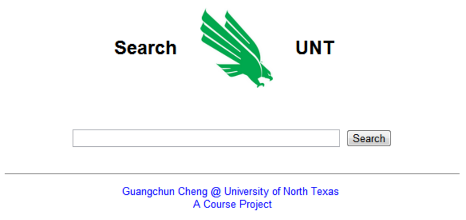
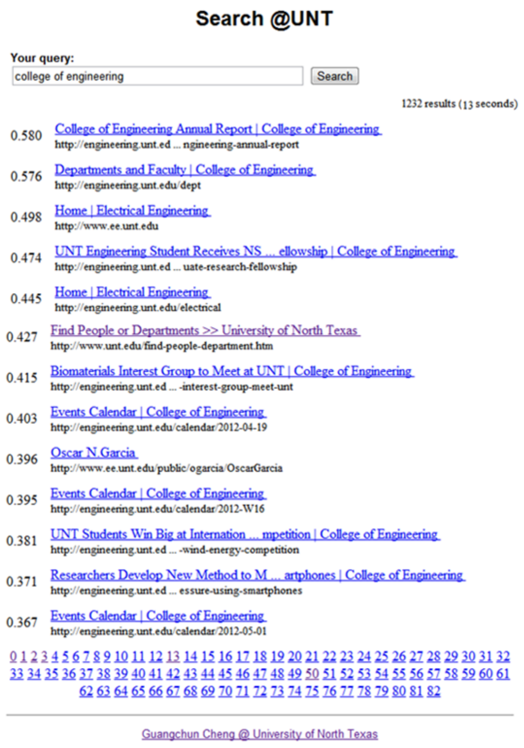


**Figure 4: online interface: homepage.**

### 4.1 Setting

For the following evaluation, we collected 3,000 web pages, starting from wwww.unt.edu and grabbing page in BFS. All successfully downloaded pages were tokenized in the way that stopwords were removed and stemming was applied. All the candidate web pages were present through the online interface as shown in Fig. 5.



Search @UNT

Your query:
college of engineering
Search

1232 results (13 seconds)

0.580 College of Engineering Annual Report | College of Engineering
http://engineering.unt.ed ... ngineering-annual-report

0.576 Departments and Faculty | College of Engineering
http://engineering.unt.edu/dept

0.498 Home | Electrical Engineering
http://www.ee.unt.edu

0.474 UNT Engineering Student Receives NS ... ellowship | College of Engineering
http://engineering.unt.ed ... uate-research-fellowship

0.445 Home | Electrical Engineering
http://engineering.unt.edu/electrical

0.427 Find People or Departments >> University of North Texas
http://www.unt.edu/find-people-department.htm

0.415 Biomaterials Interest Group to Meet at UNT | College of Engineering
http://engineering.unt.ed ... -interest-group-meet-unt

0.403 Events Calendar | College of Engineering
http://engineering.unt.edu/calendar/2012-04-19

0.396 Oscar N.Garcia
http://www.ee.unt.edu/public/ogarcia/OscarGarcia

0.395 Events Calendar | College of Engineering
http://engineering.unt.edu/calendar/2012-W16

0.381 UNT Students Win Big at Internation ... mpetition | College of Engineering
http://engineering.unt.ed ... -wind-energy-competition

0.371 Researchers Develop New Method to M ... artphones | College of Engineering
http://engineering.unt.ed ... essure-using-smartphones

0.367 Events Calendar | College of Engineering
http://engineering.unt.edu/calendar/2012-05-01

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61
62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82

Guangchun Cheng @ University of North Texas

**Figure 5: Online interface: search results.**

## 4.2 Result Evaluation

10 queries were tested, and the top 10 retrieved web pages of each query were evaluated manually to get the precision and recall. This experiment was done for both with and without PageRank. The ten queries are are shown in Table 1.

**Table 1: Evaluation queries**

| Query | Webpage URL |
|---|---|
| 1 | college of engineering |
| 2 | computer science cse |
| 3 | department of physics |
| 4 | finalcial aid |
| 5 | library |
| 6 | rada mihalcea |
| 7 | college of music |
| 8 | student activities |
| 9 | police |
| 10 | parking |

We evaluate the results by considering if the page has the same topic as the query, or say my *intent*. Although it is not very accurate, it is a simple method. By change the $w$ in (1) to 1 and 0.5, we obtained the results without and with PageRank respectively. The precision using the top 10 results are shown in Table 2. From the table we can see there is no dramatic difference in the precision for most of the queries. However, if we look at the retrieved web pages for a query, the results are different.

## 5. CONCLUSIONS AND FUTURE WORK

**Table 2: Manual performance evaluation**

| Query | P(w/o PageRank) | P(w/ PageRank) |
|---|---|---|
| 1 | 1.0 | 1.0 |
| 2 | 0.3 | 0.3 |
| 3 | 1.0 | 1.0 |
| 4 | 1.0 | 1.0 |
| 5 | 1.0 | 1.0 |
| 6 | 1.0 | 1.0 |
| 7 | 0.6 | 0.9 |
| 8 | 0.6 | 0.6 |
| 9 | 1.0 | 0.8 |
| 10 | 1.0 | 1.0 |
| Avg. | 0.83 | 0.86 |

A search engine combining vector space model and PageRank was designed and implemented. With the size of 3,000 web pages, the system can retrieve relevant web pages in about 10s per query (time depends on the load of the server.), and achieve a desirable performance in term of precision.

The future work can be done in several directions. (1) Collect more web pages to improve the retrieval results, and at the same time use distributed computing techniques such as MapReduce to improve the response time. (2) Bring more settings to the end users. While now most configuration can be done within CGI (Perl) scripts, it will be useful if the user can configure some parameters, such as when to update the collection and how many pages to collect. (3) Web page parsing should be refined, so that most useless pages can be removed and more meaningful information can be extracted form the web pages. (4) Try different approaches for the combination of vector space model and PageRank.

## 6. REFERENCES

[1] ASU. Retrieved from information retrieve project (http://rakaposhi.eas.asu.edu/cse494/f11-project-part2.htm), May 8 2012.

[2] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Comput. Netw. ISDN Syst.*, 30(1-7):107–117, Apr. 1998.

[3] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. In *Information Processing and Management*, pages 513–523, 1988.

[4] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620, Nov. 1975.

[5] Wikipedia. Retrieved from pagerank (http://en.wikipedia.org/wiki/PageRank), May 8 2012.