Design and implementation of ICS Web Search Engine

Jianlin Cheng
Information and Computer Science Department
University of California, Irvine

1. Introduction

Site-specific search engine has become increasingly important for the web sites of education institutes, government and private companies because it provides more detailed information that general search engine usually can't offer. In order to facilitate user's surfing experience on their web sites, many institutes either license the searching tools from general search engine companies such as Google or create their own primitive search engines. Neither of these two options is ideal in some case because licensing search engine usually costs a lot of money and the searching quality of self-created primitive search engine is not satisfactory due to the lack of the expertise of applying modern technologies of building search engine, although these kind of technologies are available to public due to many researcher's hard work in the field of information retrieval in several decades. So the research in building a low cost or free site-specific search engine that can produce detailed and satisfactory results in response to user's query has a wide and practical application as more and more information are put on the internet.

As an experiment of building a robust, reliable and capable site-specific search engine, I built a simple search engine for web site of ICS department of UCI using the technologies explored in the information and retrieval class (ICS 207). My experience in creating the search engine for a big web site like ICS shows that it is possible to build a reliable and capable search engine with reasonable engineering cost because of the availability of advanced search engine technologies and a lot of free software such as Porter Stemming code and FOA code for AIT data set.

2. Algorithm and Architecture of Search Engine

Building web search engine is a complex process. It is usually consisted of the following main processes: web crawling, web indexing, web searching and web ranking.

Web crawling

Web crawling is a process to collect all the web pages that are interested to search engine. It's a usually a challenging task for general search engine [1]. But web crawling is quite easy for site-specific search engine because the developers of site-specific search engine usually have access to the web pages of the web site. In this project, I downloaded about 12,000 most frequently accessed web pages from ICS web site as the main data set used by ICS web search engine. Each web page is assigned a unique id (called as document id in

-- Converted from Word to PDF for free by Fast PDF -- www.fastpdf.com --

this report). The mapping between unique ID and the URL of the web page is created and stored in a text file in crawling phase.

Web indexing

Web indexing is one of most complicated and critical process in building search engine. It includes the following steps:

Web parsing In this step each web page is parsed into pure text without html tag and the tile of web page is extracted out. The pure text of each web page is used as the document to match against the user's query in the search phase. The title of each web page is used as the description of the web link in the hitlist returned to user.

Extracting keywords In this step the lexical tokens are generated from the text files of the web pages using the lexical analyzer generator in FOA software. Then the Porter stemmer (also included as a part of FOA) is used to remove the surfacing markings of tokens. The use of stemmer here is quite important to improve the performance of each engine. First the use of stemmer obviously can reduce the size of keyword vocabulary and consequently results in a compression of the index files, also most importantly the two keywords that were once treated independently are considered interchangeable, which will increase the recall of the search engine [2]. Also a negative dictionary accompanied with FOA software is used to remove the noise words that don't reflect the contents of the document.

Inverted Indexing Inverted Index is a mapping from key word to the documents in which it appears. The correspondence between a particular keyword in the document is called posting. In this step a posting data structure that recorded all the appearances of the keywords in the web pages are created. For each key word, the posting includes the following information: total frequency, document frequency, the frequency in some documents and document id. The document id is sorted in descending order of the frequency of keyword in the document. This will make the document with more occurrences of keyword to be retrieved first. In order to normalize the similarity according to the length of each document, the document length (the number of keyword the document have) are also counted and stored in the document length file at this step.

Web Searching

In this process, the search engine first indexes the user's query to extract the keywords from the query. The same stemming technology and negative dictionary are used to preprocess the query. Then the keywords in query are matched against the inverted index to find all the document id's that contains the keywords. Then id is used to look up the URL and title of the corresponding web pages and return the results to the ranking procedure.

Web Ranking

Web ranking is very important to the searching quality of the search engine. Ideally the web page more relevant to user's query should have higher rank in the hitlist. In this search engine, Roberson and Sparck Jones TF-IDF method is

-- Converted from Word to PDF for free by Fast PDF -- www.fastpdf.com --

used to weight the keywords and cosine similarity measure is used to rank the web pages.

The TF-IDF weighting method is described as the following formula:

$$w_{kd} = f_{kd} * \left(\log \frac{(NDoc - D_k) + 0.5}{D_k + 0.5} \right)$$

Where f_{kd} is the frequency of keyword in document, NDoc is the total number of documents, D_k is document frequency of keyword.

Suppose q is the vector of query, d is the vector of document, the cosine measure is described as the following formula:

$$sim(q,d) = \cos(q,d) = \frac{\sum_{k \in (q \cap d)} w_{kd} . w_{kq}}{\sqrt{\sum_{k \in d} w_{kd}^2} \sqrt{\sum_{k \in q} w_{kq}^2}}$$

After the web pages are ranked according to their similarity with query, the web links and titles of the retrieved web page are returned to the user via the web user interface. The UI will show the results as html to user.

3. Design and Implementation

System Overview

The ICS web search engine is implemented in Perl or Java and can run on both Linux and Windows. Currently the experiment system is deployed on the Linux machine. The web address is **http://contact.ics.uci.edu**.

The whole search engine has the following main parts: web crawler, web parser, indexer, searcher, ranker and user interface. Figure 1 illustrates the relation between these modules.

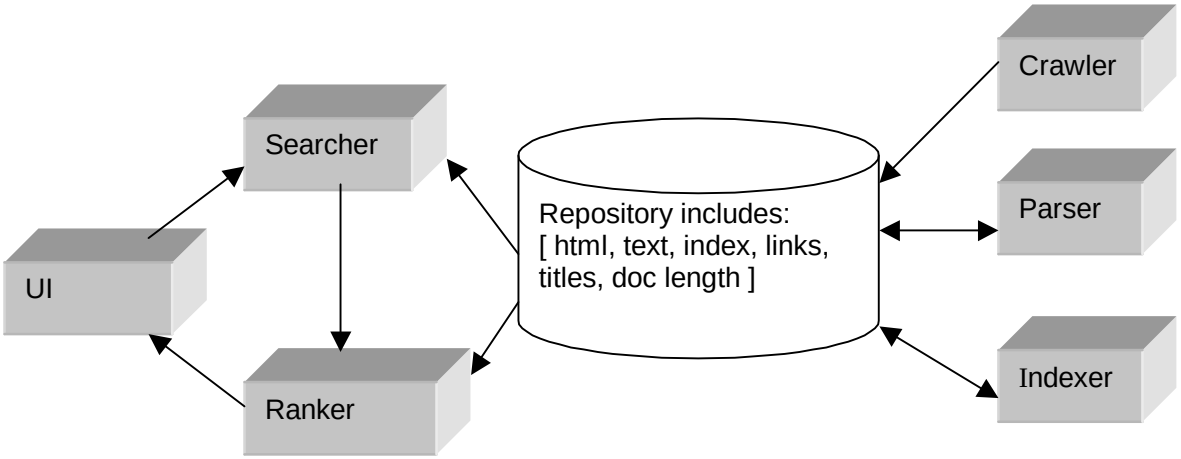

Figure 1 System Architecture of Search Engine

The crawler downloaded the web pages and saved the raw web pages into the main repository. Each web page is saved as an individual file. Each web

-- Converted from Word to PDF for free by Fast PDF -- www.fastpdf.com --

page is assigned a unique id and the mappings between the web link and id is stored in a big file (links.txt). The parser parses each html file into pure text file and the tile of the web page is also extracted. The mapping between web id and title are stored in one file (titles.txt). Given the parsed text files, the FOA indexer extracts the tokens from the text files, stem the tokens using Porter stemmer and generate keywords, and then scan the text files to create posting data structure for keywords. The inverted indexing is stored in InvertedIndex.txt.

The web UI is used for user to submit the queries to search engine and receive the search results from ranker. Given the user's query, the search engine converts the query to keywords, match the query against the inverted index and retrieve the documents containing the keyword, and then pass the web page id, web links, and titles to ranker. The ranker will rank the web pages according the similarity between web page and user's query and return the web links and titles to UI. Web UI presents the results to user.

Implementation Details

Crawler and Parser The web crawler and parser are developed in Perl because Perl is strong in text processing. Five Perl scripts were written to do all kinds of crawling and parsing job. *Crawler.pl* is used to download the web pages from the database of web pages. *GenerateLink.pl* assigns id to each web page and creates a mapping between document id and web link. *Parse.pl* is used to parse the html file into text file. *GenerateTitle.pl* is used to extract title from html and create a mapping between document id and title.

Indexer The indexer of FOA is used to create inverted index for the web pages. In order to reuse the indexer of FOA, the text files are first concatenated and converted into a big XML file. A Perl script *PreprocessDoc.pl* is created to do the conversion.

Searcher and Ranker The searcher and ranker in this search engine are developed from the FOA code. The following changes and improvement are made to the original FOA Java code.

- Change the searching mode from batch mode to interactive mode. Originally the search engine only can read the queries from file and output the results to file. I made changes to make it to receive the queries from user's input and return the result to user immediately.
- Add functionality to retrieve the web links.
- Add functionality to retrieve the web titles.
- Add functionality to support multiple users' access and continuously searching.
- Fix a few bugs of original code.

Web User Interface Web user interface is the bridge of connecting user and search engine. It collects queries from user's web browser, submit query to search engine, receive the results from search engine, then generate html from result and send the html back to user. The web user interface is developed using Java Servlet and deployed in TomCat, which is the standard Java framework to run Java Servlet.

-- Converted from Word to PDF for free by Fast PDF -- www.fastpdf.com --

Major Data Formats

Web links -- Links.txt, mapping between document id and URL

Table 1, Format and example of web links data

Document Id	Link
1	http://www.ics.uci.edu/~pazzani/

Web titles – Titles.txt, mapping between document id and web title.

Table 2, Format and Example of web titles data

Document Id	Title
1	Michael J. Pazzani

Posting Structure – InvertedIndex.txt

Table 3, Format and example of inverted index

Keyword	DocFreq	TotalFreq	Freq in Doc	Separator (-1)	Doc Id List	Separator (-2)	…
abrupt	3	3	1	-1	28, 36, 75	-2	…

Document Length – docLength.txt, the length of each document

Table 4, Format and example of document length

Document Id	Length
1	145

Running System

The running search engine is deployed on the Linux server. The search engine can be used as following steps:

- Use browser to connect to the web search engine at **http://contact.ics.uci.edu**
- Following web page will popup.

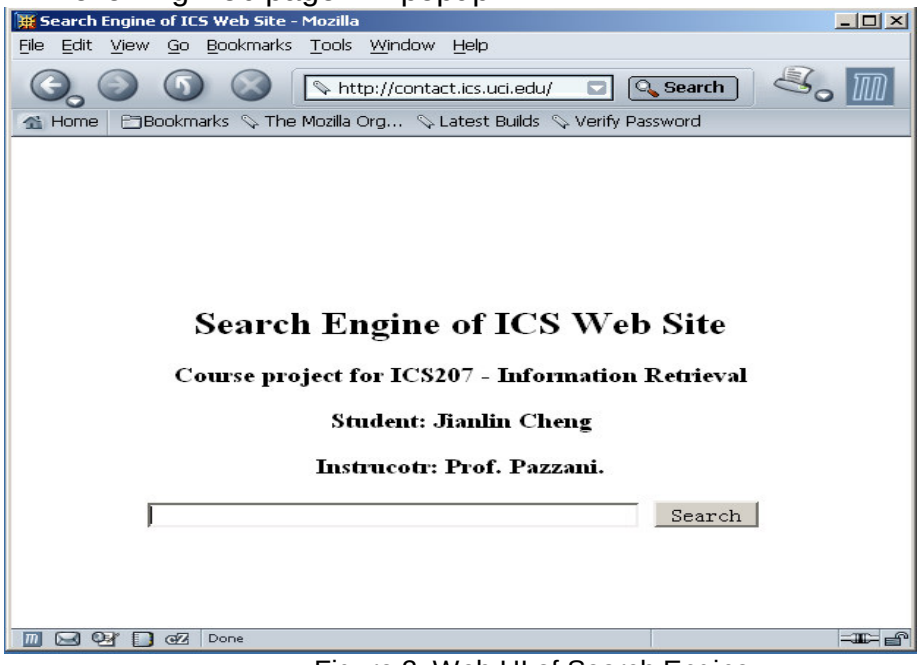

Figure 2, Web UI of Search Engine

-- Converted from Word to PDF for free by Fast PDF -- www.fastpdf.com --

- Input the query and press Search button to submit the query. For example, query “Student Affairs Office”.
- The returned results will show in the following web page.

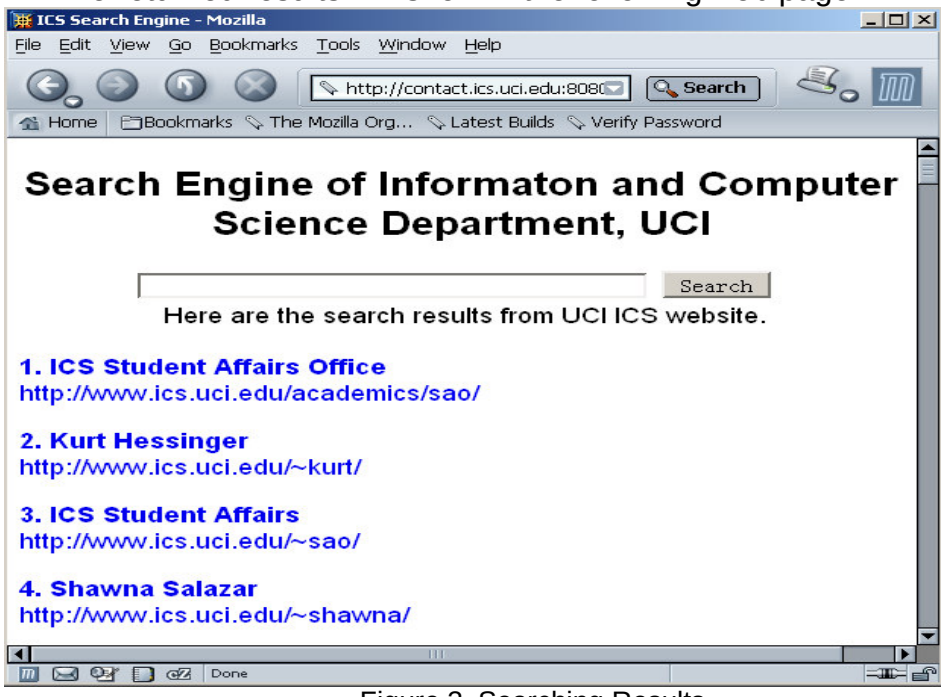

Figure 3, Searching Results

4. Evaluation

Evaluating of the search result is to measure how well the returned results meet the user’s particular information need. Here I will user two metrics to evaluate the performance of the search engine from two aspects.

Evaluation Metrics

Precision The precision is to measure how much of what the users see is relevant [2]. This is very important for search engine given millions of document on the internet. This measure is defines as:

$$\Pr ecision = \frac{|\mathrm{Re}tr \cap \mathrm{Re}l|}{|\mathrm{Re}l|}$$

Point Alienation Point alienation is a measure to evaluate the rankings of the documents. The basically idea is to compare the difference in rank between two differentially preferred documents to the absolute difference of these ranks [2]:

$$J \equiv \sum_{d \succ d'} \frac{Rank(d) - Rank(d')}{|Rank(d) - Rank(d')|}$$

If d is really preferred over d’ and rank(d) < rank(d’) as we expected, the numerator will be negative, on the contrary, the numerator will be positive.

-- Converted from Word to PDF for free by Fast PDF -- www.fastpdf.com --

Comparing this arithmetic difference to its absolute value and summing over all then pairs of retrieved documents that are differentially preferred, we get a score to evaluate the rankings of the web pages. The smaller the score is, the better the ranking is. The Point alienation score is between -1 and 1. The smaller the score is, the better then ranking is.

Experiment and Result

In this report I tested four different kinds of queries and evaluated the precision and point alienation of the first 10 retrieved results. I got the following results. Due to the limit of space, the retrieved web links are omitted here, but they are very easy to be retrieved by testing on the real system using these queries.

Query	Results	Precision	Point Alienation
1. Student Affairs Office	1,3 very relevant 2,8,9 relevant 4,5,6,7, 10 irrelevant	50%	-0.578
2. Support vector machine	3,4,5,9 very relevant 1,6 relevant 2,7,8,10 irrelevant	60%	-0.362
3. Machine learning data set	3,5,6,8 very relevant 4,9 relevant 1,2,7,10 irrelevant	60%	-0.035
4. Michael Pazzani	All first 10 web pages are relevant to some degree. But the home page is only ranked as 20th.	100%	0 (there is no obvious differential preference between first 10 results)

Table 5, Searching Result, Precision and Point Alienation

The results show that the general precision of the first ten retrieved web pages is above 50%, which is reasonably good as I expected. The rankings of retrieved web pages are not so good in some situation (such as query 4). For query 4, even though the similarity between query and the top web pages in the hitlist is high, the web pages are not very relevant. Instead the most relevant web page (home page of Prof. Pazzani) is only ranked 20th. This indicates that the similarity ranking alone is not enough to find the most relevant web pages in some case. The algorithms that utilize the reference relation between web pages such as PageRank [1] can rank the web page according to its relative importance, thus be able to improve the ranking further.

5. Conclusion

In this project I applied a set of IR technologies to build a middle size site-specific search engine based on FOA successfully. The algorithms such as word stemming, document indexing, keyword weighting and similarity ranking prove to be very useful to build a search engine with good performance. As the availability of information retrieval and search engine technology, it feasible to develop the

-- Converted from Word to PDF for free by Fast PDF -- www.fastpdf.com --

standard site-specific search engine that can be used by all kinds of institutions with reasonable engineering cost.

The similarity ranking method alone is not good enough to find out more important web page. So the link structure between the web pages must be used to improve ranking. Thus in future, it will be very interesting to implement PageRank algorithm to rank the web pages and compare the result with similarity ranking method used in this project.

References

1. Sergey Brin, Lawrence Page, the Anatomy of a Large-Scale Hypertextual Web Search Engine, Computer Networks and ISDN Systems, 1998.
2. Richard K. Belew, Find out about, A Cognitive Perspective on Search Engine Technology and the WWW, Cambridge University Press, 2000

Attachment

search.zip: All the source code for this project.
readme.txt: describes the structure and the function of code.
The running system, this report, the description of the project and source code is also available at **http://contact.ics.uci.edu**.

-- Converted from Word to PDF for free by Fast PDF -- www.fastpdf.com --