Custom Web Search Engine Project

Progress Report, 4.27.15

1. Project Contributors

- Emre Can Kucukoglu, eckucukoglu@gmail.com
- Fatih Hafizoglu, fatihafizoglu@gmail.com
- Yigit Ilguner, yigit.ilguner@gmail.com

2. Description of the Problem

In web search domain, website's anchor texts can be used by crawler to find other related websites more easily. In a limited corpus and time, user (query creator) selected website helps the crawler find more precise results. Project has 2 phases. In phase 1, according to user settings, crawler will be processed. Crawled data will be stored after indexing for future queries. Next phase, user queries will be answered. Since websites are crawled with phase 1 settings, corpus's recall and precision values are expected to be higher.
Moreover, in addition to focused, customized crawling phase, enhancements that optimize the scoring and ranking will be implemented in second phase.

3. Literature survey

There are plenty of open source projects to crawl the web. We analyzed three of them: Scrapy[1], Apache Nutch[2] and Heritrix[3].
Scrapy is an open source and collaborative framework for extracting the data from websites. It is written in python and since it doesn't support running in a distributed environment, it is mainly prefered when focused crawl operations will be done. Moreover it supports JSON, XML and CSV export formats.
Heritrix, which is written in Java, is the most stable ones between all of them. Even though, heritrix is However, it devoids of Apache Nutch's integration benefits for Apache Lucene[4] and Apache Solr[5]. Apache Solr is written in Java and runs as a standalone full-text search server within a servlet container. Since project has a web front end which is connecting to a database that searching will be done, Lucene/Solr api will be used to index the data directly from the database.

In second phase of project, while computing scores of a documents, customizable query processing optimization algorithms will be implemented. According to WAND optimization[6], we need to compute an upper-bound for each term, which is the highest possible contribution to the score from any of the documents in this term's postings list. Ideally, for a given term, this is the highest term-frequency value in the term's posting list times the term's IDF. With WAND optimization, processing time is expected to decrease dramatically.

Moreover, the result set should be as diverse as possible. Maximal Marginal Relevance [7] (MMR) iteratively constructs the result set by selecting a new element in corpus that maximizes the mmr function. In mmr function, diversity has a tradeoff value with relevance score function. Hence, MMR technique will be used in Custom Search Engine Project to produce diverse result set.

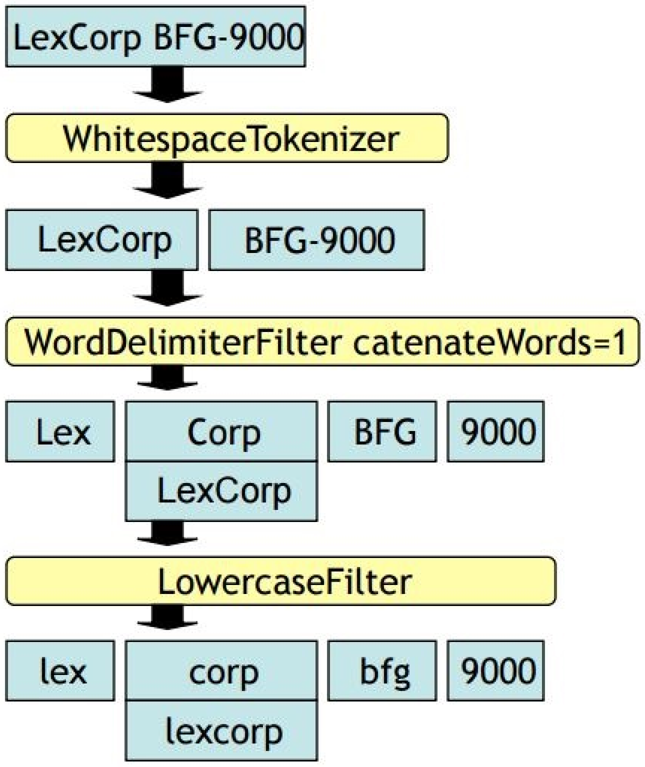
4. System Architecture Design

Data

Document corpus is constructed dynamically by beginning at a seeded web page and traversing through user specific number of levels. Instead of looking at a fixed collection, resizable corpus is decided for obtain more precise findings. Apache Nutch is used for searching and processing the documents.

Index

After crawling phase has completed, crawled websites will be given as a collection of Documents to Apache Lucene indexer. In Lucene, a Document is a collection of Fields. And a Field is content along with metadata describing the content. These fields will be used for fielded searching (e.g. title, author) in phase 2. Fielded search will be presented to user as an query option. All cases will be lowered. To split the words, whitespace tokenizer will be used. General
schema of document indexing analysis is shown below:



CompressionTools class of Lucene is used to compress and decompress binary data for stored fields. It supports gap compression and delta compression algorithms. Moreover, phrase and wildcard queries are supported by Lucene framework. Inverse indexing is used, so that for a term, the documents that contain it will be listed. This is the inverse of the natural relationship, in which documents list terms.

Query Processing
Vector space model is selected as query processing model. For ranking phase, WAND[6] optimization over tf-idf weights of documents is applied. In WAND approach, documents that satisfy a minimum score are stored in a heap. Pointers are assigned for each posting list of the query terms. At a time *t*, document identifiers of currently analyzed postings from every pointer are ranked in an ascending order. We look for the pivot document whose score after adding the preceding ones' exceeds the heap threshold. If a document includes enough query terms to predominate the limit of the heap, then the document is eligible to be picked. Afterwards, the element that has the minimum tf-idf score in the heap is removed and ordering in the list is updated by scores. Unless, the document does not reach enough score, then current postings that show the smaller identifiers are iterated until their document IDs become greater than or equal to pivot document's. After processing every posting, documents in the heap give the most relevant result for the query. Moreover, diversification of the query is ensured by MMR[7].

```
Algorithm 4 MMR
Input: candidate set $S$ and result set size $k$
Output: result set $R \subseteq S$, $|R| = k$
1: $R \leftarrow \emptyset$
2: $s_s \leftarrow \text{argmax}_{s_i \in S}(mmr(s_i))$
3: $S \leftarrow S \setminus s_s$
4: $R \leftarrow s_s$
5: while $|R| < k$ do
6:     $s_s \leftarrow \text{argmax}_{s_i \in S}(mmr(s_i))$
7:     $S \leftarrow S \setminus s_s$
8:     $R \leftarrow R \cup s_s$
```

Hence Maximal Marginal Relevance algorithm considers the tradeoff between relevance and diversity, it guarantees that result set is as diverse as possible. From the above algorithm, in line 6, from each candidate set of results is chosen according to its mmr function result.

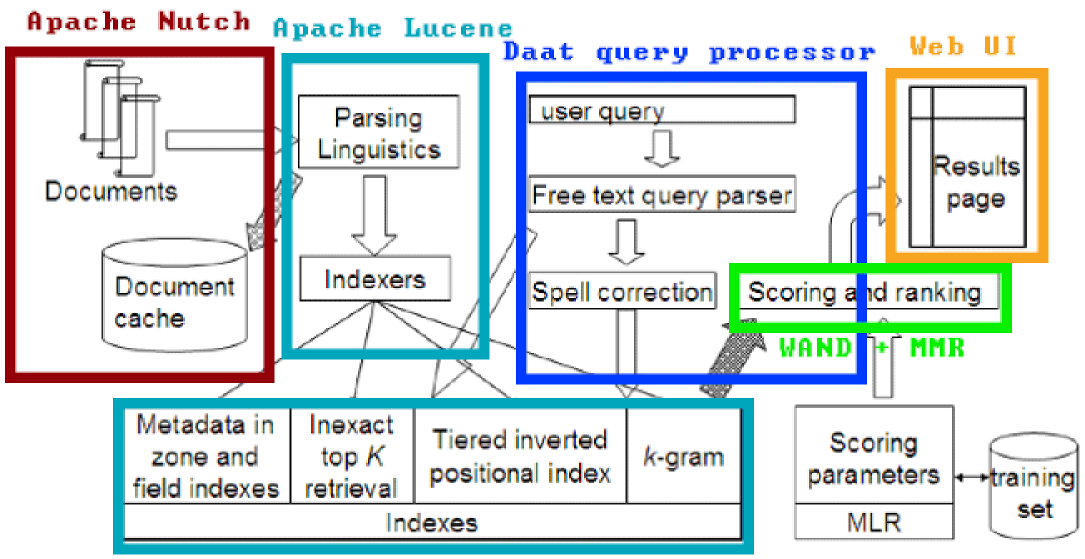
$$mmr(s_i) = (1 - \lambda)\delta_{sim}(s_i, q) + \frac{\lambda}{|R|} \sum_{s_j \in R} \delta_{div}(s_i, s_j)$$

As seen above, mmr function considers both query-result set similarity (i.e. relevance) and diversity between result sets.

User Interface

Web-based user interface gives opportunity to decide the initial web site and to adjust the parameters such as the maximum number of web pages investigated through, farthest page distance from the starting point for crawling and indexing. After the indexing stage is finished, search query is ready to enter. Besides being sorted, diverse and relevant results are aimed to display to user by exploiting the methods described in previous section. We also benefit from query parsing and and search features of Apache Solr. System will support wildcard queries, however snippets are not the aim of this project. Result set will be sorted according to their rankings.

Complete system can be seen below figure:



5. Current Progress

Until now, in project’s second phase, query processor part is implemented. After crawled pages are indexed with the help of Apache Lucene, user queries will be processed. In here, we use vector space model and we have implemented tf-idf score computation for documents. Document at a time processing is used instead of term at a time. Moreover, WAND optimization is completed to achieve less score computation according to frequencies of term within documents.

6. Remaining Work

The first phase which consists of analyzing and processing the documents, tokenizing the words obtained from crawling work and constructing the index is going to be implemented. Furthermore, user interface and search result representation will be integrated.

7. Expected Results

As a result of exploiting the initial web page given by the user, ranking and diversification methods applied to the initial results, it is expected to get more precise and specialized documents. Additionally, with the help of WAND optimization, almost 20% less score computations are expected to be done by the system.

8. References

[1]: http://scrapy.org/
[2]: http://nutch.apache.org/
[3]: https://webarchive.jira.com/wiki/display/Heritrix/Heritrix
[4]: https://lucene.apache.org/core/
[5]: http://lucene.apache.org/solr/
[6]: Andrei Z. Broder, David Carmel, Michael Herscovici, Aya Soffer, Jason Y. Zien: Efficient query evaluation using a two-level retrieval process. CIKM 2003: 426-434
[7]: J. Carbonell and J. Goldstein, “The use of MMR, diversity-based reranking for reordering documents and producing summaries,” in Proc. ACM SIGIR, 1998