Webintelligence - Mini Project 1

Lars Andersen, Mathias Winde Pedersen & Søren Skibsted Als

1. oktober 2014

Crawler

The crawler serves as the obtainer of website links. When doing this, it has to make sure that it follows the robots.txt specification. Furthermore, it should not visit a site too often. In order to not consume too much space, near duplicate detection is also performed here.

We decided to implement the crawler with mercator. However, for the front queues we use a single queue, resulting in a uniform ranking. An example of ranking that could be used there would be to rank news sites who change regularly over more static websites. Furthermore, the crawling is single threaded, as a proof of concept, but could be expanded to a multi threaded one. This is also related to DNS lookup times, as it was found that some DNS lookups delayed the crawler severely.

To check near duplicates we use shingles with sketches. However, Google recommends 84 hash functions, but we only use 19 hashes. Our shingle size is 8, which is recommended.

Furthermore, tackling documents on the website which is not html or txt encoded is not handled, which gives crawled text encoded in a strange manner, e.g. pdf files. Image files are excluded, as some of those formats made the crawler otherwise crash, when trying to convert it to a string. A library could be used to translate such text into a readable format, to be used for the indexing, and also to make more sensible near duplicate detection.

For the seed we used `http://aau.dk/` and `http://stackoverflow.com/`

Indexer

The indexer has the purpose of constructing the inverted index, in that regard it is important to keep statistics of term and doc frequency, such that this information can be used by the ranker to rank pages.

For the indexer we cut corners as follows. We only stem on the English language. For stopwords it is used for English and Danish, As we encode weird strings due to extracting pdf files without a library, we get some strange terms that fill the index unnecessarily.

Furthermore, the postingslist are all kept in main memory. This works fine for a small size of websites crawler, however, if you were to crawl the whole web, you would have to have servers farms or write to disk(making it much slower).

Ranker

The ranker serves the purpose of ranking the pages, given a search query. This is meaningful, as it contrary to boolean matching can weight different documents, and as of such can rank the results on what matches the query best, and not just if the terms occur in the document or not.

For ranking we used the algorithm provided from the slideshow, which means we used the CosineScore, based on the tf-idf weighting. For contender pruning we used a precomputed champion list. This made it possible to pre-compute this, and significantly reduce the computations needed when ranking, as the champion list limits each posting to a list of r docs, the docs with the most occurrences of the given term the posting is associated with.