Information Retrieval – Project 2

Group ID: 24 December 12, 2016

Paulus Max
paulusm@student.ethz.ch
15-945-223

Raiskin Yarden
raiskiny@student.ethz.ch
15-927-189

## General notes

- Data preprocessing: Text was extracted from the XML <head> and <text> tags and all special non-alphabetical characters were removed. Then, tokenization, stemming and stop words removal were applied. Finally, we applied custom hashing the tokens, from string to integers, in a manner that ensured no false collisions. After acquiring the entire collection, we removed all tokens that had a document frequency of less than 5. The pruning step reduced the number of distinct tokens from 1,356,183 to 176,866.
- Data quality and anomalies: While manually inspecting the tokens we have noticed that misspelled words were more frequent than we had expected. We ruled out bugs from the tokenization process by inspecting the raw XML files, which were the source of the problem. We hoped to alleviate this issue by the pruning step.
  We have also noticed that 2,478 documents appeared to empty. After closer inspection we concluded that these documents had irregular XML tag structure, where at least some documents seemed to be product descriptions. Due to the goal and scope of this project, we decided not to further investigate this anomaly.
- Candidate documents policy: One possible way to use the inverted index would be to consider all documents that contain any token from the query, i.e. the (distinct) union of all documents returned by the inverted index. Since this set is relatively large for some queries (up to ~40k documents), we applied a heuristic to weed out less relevant documents. For each query, we retrieved the inverted index set of documents and count the number of occurrences of each document in this set. We then iteratively look at subsets of documents that contain at least k tokens from the query, in descending order, stopping when the subset contains at least m documents ($m \geq 100$).
- Memory foot-print: 4,117 MB, computed as total memory – free memory.
- Average running time per query: estimated with and without indexing on the query test set. We still assumed access to information derived from preprocessing, such as collection and document frequencies (but not to the inverted index). Results varied substantially between different runs on our machines are reported on the test set. We observed faster results on the training set than on the test set.
  - Term-based model: 14.1 – 19.8 seconds / query with indexing 21.3 – 29.2 seconds / query without indexing
  - Language model: 14.8 seconds / query with indexing, 24.1 seconds / query without indexing

## Term-frequency-based model

- **Simplest tf-idf model:** Our exploration began with implementing a simple tf-idf model. We computed the inverse document frequency to assess how much information different terms in a query provided. This was deemed necessary as the number of query tokens ranged from 2 to 6 tokens across queries and different tokens could be relatively more or less important than others. In the simplest model, we computed the tf-idf based on the raw term-frequency and used the sum to create a ranking. Because the computation was relatively cheap, all documents in the union (see above) were assessed and candidate document policy was not applied. The results of this simple model were not satisfactory with 9 out of 40 training queries receiving no relevant document at all in the selection of top 100 and consequently a low MAP just slightly above zero. We identified several potential sources of this poor performance, for example we noticed that the length of documents varied substantially between a single (0 respectively) token and 182,367 tokens, such that the term-frequency count should be normalized to take the document length into account.
- **Okapi-BM25 model:** While looking for possible ways to improve the simple term-frequency model and more sophisticated information retrieval systems, we encountered the Okapi-BM25 ranking function and implemented a variant of it (as described in the accompanying textbook, Manning et al., 2008). The raw ingredients for this model are still the idf and the tf, but both are normalized to take account of varying document lengths. The model contains two hyper-parameters that make this normalization explicit. After we experimented with different values of these hyper-parameters, but noticed that the performance was changing only marginally, we opted for the standard parameter setting ($k = 1.2$, $b = 0.75$, see book or Wikipedia), which gave a (bounded) MAP performance on the training set of 0.323 on the training set, which was a significant improvement over the simple tf-idf model. As a final note, notice that the query answering time of our term-frequency based model is larger in a few cases than that of the language model, but this is in large part attributable to the term-frequency based model operating on the entire document union for a query while the language model operates on the candidate documents policy.

**Language model**

- Model description: We have implemented a MLE, combined with linear smoothing with the collection frequencies (Jelinek-Mercer smoothing). We constrained the smoothing parameter to be constant across documents, $\lambda_d = \lambda$. We allowed for two different possible representations of term frequencies and document frequencies; raw term counts and a log(1+x) transformation. Furthermore, we considered different threshold values for our candidate documents policy.
- Model hyper-parameters and configuration: We have estimated the (bounded) MAP score over different values of the smoothing parameter, $\{10^0, \ldots, 10^{-4}\}$, different candidate documents thresholds {100, 1,000} and different term counts transformations {linear, log(1+x)}. Unsurprisingly, $\lambda$=1 performed the worst over all configurations. However, the linear term frequency representation slightly but consistently over performed the log term frequency representation. The highest MAP score achieved was 0.289, with 1,000 document candidate size, linear term frequency representation and $\lambda$=0.1 (see figure below). These were the model configurations chosen to rank the test queries.