Information Retrieval – Project 2

Group ID: 24 December 11, 2016

|  |  |
| --- | --- |
| Paulus Max  [paulusm@student.ethz.ch](mailto:paulusm@student.ethz.ch)  15-945-223 | Raiskin Yarden  [raiskiny@student.ethz.ch](mailto:raiskiny@student.ethz.ch)  15-927-189 |

report-[groupid].pdf : a short report (max. 2 pages) that describes your system, your retrieval models as well as your training set performance in terms of MAP. You should also report average (per query) running times when using an inverted index versus when passing the document collection sequentially for each query.

**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 we more frequent than we have 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 ≥ 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).
  + Term-based model
  + Language model: 14.8 seconds / query with indexing, 24.1 seconds / query without indexing

**Term-based model**

* Prediction: bla bla bla

**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, λd = λ. 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 MAP score over different values of the smoothing parameter, {100, …, 10-4}, different candidate documents thresholds {100, 1,000} and different term counts transformations {linear, log(1+x)}. Unsurprisingly, λ=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 λ=0.1. These were the model configurations chosen to rank the test queries.

