Tutorial Information Retrieval

Universität Innsbruck - Department of Computer Science

E. Zangerle

2019-04-03

Exercise Sheet 3

Exercise 1 (n-gram Indexing) [3 Points]

Extend your fulltext index of the previous exercise sheet with n-gram functionality (i.e., adapt the way your keywords are represented in the index structure). Compare the performance of n-gram sizes with n = 1, 2, or 3 to your previous versions of the indices created in exercise sheet 2 (both in terms of index creation and query processing time).
Please extend your evaluation procedure of the previously used ground truth file `queries.csv` for benchmarking the ranking functionality. For the implementation of the similarity measure for a given query q and a given document d, re-use the similarity measure $sim(q, d)$ as defined on the last exercise sheet. Use this similarity as a measure for relevance and hence, use it for ranking documents for a given query.
What happens if you cap the number of relevant items to be considered for the evaluation at a result list position k (i.e., only use the top-k documents for the evaluation; often referred to as @k) in contrast to the setup where all relevant items (given in the ground thruth file) are considered (as performed in the last exercise sheet)? How do recall@k and precision@k behave when increasing the number of evaluated documents k?

Exercise 2 (Vector Space Model) [3 Points]

The next step for our fulltext index is to implement retrieval based on the vector space model. Here, we rely on the term frequency–inverse document frequency vector representation for texts. Therefore, please create tf-idf-weighted vectors for the documents and queries and use cosine similarity for the computation of the similarity $sim(\vec{q}, \vec{d})$ of a document d and a query q. Subsequently, use the similarity score sim for the ranking of relevant documents.

$$sim(\vec{q}, \vec{d}) = \frac{\vec{q} * \vec{d}}{\|\vec{q}\|\|\vec{d}\|} \tag{1}$$

Please compare this index to the previously created n-gram indices created in Exercise 1.

Exercise 3 (Elasticsearch) [4 Points]

To end our block on traditional text retrieval approaches, we will look at a state-of-the-art text search and retrieval engine. The goal of this exercise is to get a first hands-on-experience with ElasticSearch[1] (and the underlying Apache Lucene engine[2]) and to look into the retrieval methods employed by ElasticSearch.

[1] https://www.elastic.co/products/elasticsearch
[2] http://lucene.apache.org/

Please note that we do not aim to actually benchmark Elasticsearch here, but rather, we aim to get a first impression of how such a system can be used and configured. Hence, for answering the following questions, you should play around with and make use of Elasticsearch's functionalities, but also look into the documentation to get a deeper understanding on how these functionalities are actually realized.

a) *1 Point* Please download and set up Elasticsearch[3]. Fill the index with your benchmark documents or alternatively, you may also add any other new documents of your choice. Use the previously used benchmark queries on it and make sure to test both the boolean and the vector-space capabilities of ElasticSearch.

b) *1 Point* How does scoring in Elasticsearch resp. Lucene work? Which options are available and how do these work?

c) *1 Point* How does Elasticsearch resp. Lucene ensure performance? What are the critical aspects for performance in such a text retrieval system and how can these be overcome?

d) *1 Point* How does Lucene compute answers to wildcard and fuzzy queries efficiently?

Important: Submit your solution to OLAT and mark your solved exercises with the provided checkboxes. The deadline ends at 23:59 on the day before the discussion.

[3] https://www.elastic.co/products/elasticsearch