Simple Search Engine

Bulat Nasrulin

Innopolis University

Our approach

We implement simple search engine using inverted indexes, tf-idf.

The first issue that we faced when implementing an search engine is the lack of datasets for search evaluation. To evaluate the search results we use NPL dataset [1]. The NPL (also known as the VASWANI) collection is a collection of around 10,000 document titles. The dataset has 94 queries, each query is associated with document ids that are related to that query.

To unify the retrieved results we first use word_tokenize and then apply Porter stemmer. For tf-idf log normalization was applied.

Query processing steps:

1. Map each word documents ids using inverted index
2. Calculate tf-idf for word, document pair
3. Sum up tf-idf for all words in a query - in this work we use this number as a relevance representation of a document
4. Rank documents by their tf-idf

Dataset evaluation

For each query dataset provide some small number of assigned documents (from 2 to 15). The assigned documents are sorted by their ids, and are not ranked. In this work F-score is used as a main score function for evaluation. We say that we correctly predict the document if the document id is presented in the assigned documents set.

To extend beyond simple tf-idf, we propose using normalized tf-idf, tf-idf build on bigrams and a combination of simple tf-idf and tf-idf build on bigrams (bi-tfIdf).

Moreover, we compare the score of the model when releasing the whole set of related documents with the score of releasing only a part of the set. To find the best fraction to release, we use GridSearch, trying to find the fraction, for which the F-score is maximized.

Experiments

In this section, we present the results for accuracy of different models.

Simple tf-idf

Simple tf-idf when releasing all documents give F-score = 0.02. Although we may identify all the documents, we also provide 98% of irrelevant documents. If we rank the documents, and take only the required number of documents (for a specific query), the mean accuracy is 0.44.

Tf-idf on bigrams

This approach showed worse results than simple tf-idf, with the mean accuracy 0.23. But during the experiments we notice that it correctly identifies documents (high rank), which are underranked by simple tf-idf. This notion leads to the idea of using a combination of this ideas.

Combination of simple Tf-idf and Bi-tfIdf

This approach showed the best results achieving the mean accuracy of 0.459.

Increasing the number of published documents also increases the chance of getting the right document, but at the same time it also increases the amount of incorrect results (noise). To find the optimal value of the fraction for released documents, we implement a simple GridSearch, in which we try to find the fraction that maximizes the F-score.

The experiments showed that if we release 2% top ranked documents, it would maximize the F-score. But in fact, this value we only achieve mean F-score = 0.18. So in fact it is better to use N top-ranked documents, where N is the number of assigned documents in the test dataset.

Final notes

The experiments also showed, that some queries don't even contain the keywords that are presented in the given dataset. For example, the best response for query 'humans and other species' is the documents with words: mammals, animals, etc. To build a good search engine we need to consider the context, the sense and the synonyms.

References

1. Glasgow IR test collections, $http://ir.dcs.gla.ac.uk/resources/test_collections/$