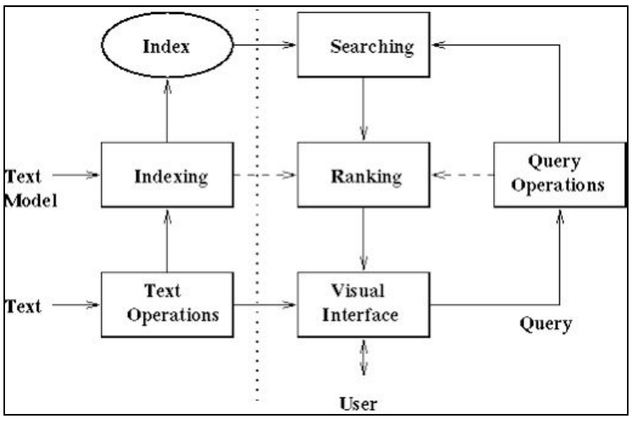
Summer Semester 2020 SNLP Final Project Report

Name: Awantee Deshpande
Id: 2581348
Email: s8awdesh@stud.uni-saarland.de

Name: Lakshmi Rajendra Bashyam
Id: 2581455
Email: s8laraje@stud.uni-saarland.de

[I] Introduction to IR

An information retrieval task is an end to end system that takes a query from a user and scans over a corpus of documents to return the documents relevant to the query in a ranked manner. It is end-to-end in the sense that there must be a well-defined interface for entering the query, processing it to get the sense, doing a lookup and retrieval over a collection of documents (which must be meticulously assimilated as well), ranking the documents by their relevance to the query, and then returning this ranked list.
While the current trend is that of Question Answering i.e. retrieving the direct answer to the query in the form of a phrase, even a few QA systems have an underlying basis of information retrieval to them. Some IR systems also incorporate relevance feedback from the users to improve the ranking process. The general flow of an IR system is given below.



source: http://people.ischool.berkeley.edu/

At each phase, the model can base the task on different kinds of metrics and retrieval model choices. E.g. Indexing can be done using Boolean retrieval, tf-idf, ranking can be done using a learning to rank framework using ML, neural models etc, the final result can be evaluated using scores like precision, recall, f-measure, MRR, MAP and so on.

[II] Motivation

Information Retrieval can be highly observable. Information is the need to every problem and this makes designing highly fast and efficient IR systems very important. It is an amalgamation of different techniques w.r.t to storage, text processing, memory optimisation, retrieval, scoring, ranking, feedback analysis and so on. The key point here is that there is a lot of data available to us in unstructured format. Information retrieval is about satisfying diverse information needs using this unstructured data. While the focus in this project is on a very small collection of xml formatted documents, in reality, the entire Web could be a corpus with differently structured documents. Other data formats like images, videos etc. also exist. In a general IR system, the information is crawled over, extracted and cleaned, preprocessed, and indexed first before doing a similar process over queries and returning a ranked set of answers. Each step in this pipeline is an optimisation task by itself. Information needs are also

quite diverse - queries could be ambiguous, phrases with synonymous interpretations, time-aware, quantity-based etc. Capturing all these aspects while designing an IR system is important and challenging.
Even in real world tasks, every time we look for something, it is akin to an information retrieval system in its own sense. The current project not only involves a baseline IR task by designing an indexing and ranking scheme over documents, but also offers the implementation of comparative methods to improve the ranking and making IR more efficient. Thus, it is a highly interesting aspect of language processing models to work on.

[III] Project Description

The current project is based on three aspects of a general information retrieval model

1) Indexing and retrieval -
Termed as the baseline model, this step is carried out using a standard tf-idf vector space model. The tf and idf values are calculated using

$$tf(term_i, doc) = \frac{n(term_i, doc)}{max(term_j, doc)}$$

$$idf(term_i) = log(\frac{N}{n(term_i)})$$

2) Reranking
To get a better ranking of the documents, we utilise the Okapi BM25 function to rerank the documents per query according to the query terms. The BM25 score of a document is given by

$$\text{score}(D, Q) = \sum_{i=1}^{n} \text{IDF}(q_i) \cdot \frac{f(q_i, D) \cdot (k_1 + 1)}{f(q_i, D) + k_1 \cdot \left(1 - b + b \cdot \frac{|D|}{\text{avgdl}}\right)}$$

where f(qi, D) = term frequency of query term qi in D
|D| = length of D in words
avgdl = average length of documents
k1, b = free parameters

$$\text{IDF}(q_i) = \ln(\frac{N - n(q_i) + 0.5}{n(q_i) + 0.5} + 1)$$

where N = total no. of documents
n(qi) = idf of query term qi in document collection

The documents are re ranked by this score and the top 50 documents are returned.

3) Sentence based ranking
In the 50 documents returned in step 2), each sentence is considered as a separate document and their order in the documents is assumed to be the default rank of the sentence. Then, these sentences are ranked again by the BM25 score, and the top 50 sentences per query are returned.
The performance is evaluated by using Mean Reciprocal Rank (MRR) as a metric.

$$MRR = \frac{1}{|Q|} \sum_{q=1}^{Q} \frac{1}{rank(q)}$$

Finally, a brief evaluation and analysis of the observed and expected performance are done with the help of the obtained results and graphical illustrations.

[IV] Project Modules

IV. a) Preprocessing

a) Document preprocessing - The documents are in the file ‘trec_documents.xml’ in the format with the text captured in <TEXT></TEXT> and <P></P> tags. Each document is uniquely identifiable with <DOC></DOCNO> tags. The subsequent text is retrieved and stored as key-value pairs with the document ids with the use of BeautifulSoup for xml scraping.

b) Query processing - Queries are present in a similar xml format in ‘test_questions.txt’. The corresponding answer patterns are in the file, ‘patterns.txt’. These two files are jointly processed to capture all the aspects of a query to get a format of the form <query id, query text> and <query id, query pattern> to make it suitable for further tasks.
The actual text in both these tasks is preprocessed by standard operations like punctuation removal, sentence splitting, lowercase conversion etc.

IV. b) Models

a) Baseline - The tf-idf model is implemented as a baseline method with the corresponding class of the same name. Using the formulae mentioned above in section [III](1), the tf, idf and tf-idf values are calculated over the processed document corpus (see files ‘tf.json’, ‘idf.json’). Then the queries are processed one by one, and the documents are retrieved by the corresponding cosine similarity scores between the query tf-idf and document tf-idf values. The baseline.py file contains the class TF-IDF with its method to implement it from scratch.

b) Okapi BM25 - (BM = Best Match) ranks the documents based on the query terms appearing in them irrespective of their positional proximity in the document. The top 1000 documents returned by the baseline model are reranked using the BM25 formula mentioned in [III](2), of which the top 50 documents are then returned. BM25 has its own underlying implementation of tf and idf which are different from those implemented in the tf-idf models. The gensim library implementation of BM25 has been utilised here. The class bm25.py contains the class BM25Model, which defines the functions to implement re-ranking using bm25.

c) Sentence ranking - Here, every sentence in the ranked documents is considered as a separate document in itself and reranked by the implemented BM25 scheme. Then, the rank of the first relevant sentence for the query is found and is used to find the MRR for all hundred test queries as per the formula in [III](3). The file ‘bm25_sentence.py’ contains the BM25SentenceModel class that reimplements bm25 for sentence ranking.

d) Metrics - The documents retrieved in baseline TF-IDF and BM25 models are evaluated with the precision scores. The sentences returned by BM25 sentence ranking model are evaluated with the Mean Reciprocal Rank. The file ‘baseline.py’ contains the class Metrics for evaluating the precision and MRR metrics.

Dataset:

The corpus and dataset comprise the following documents:
dataset - trec-documents.xml, test_questions.txt, patterns.txt (contain the trec dataset and queries with which the documents to be retrieved)
generated - contains the intermediate json files generated for further score based calculations

Technical Requirements Specification:

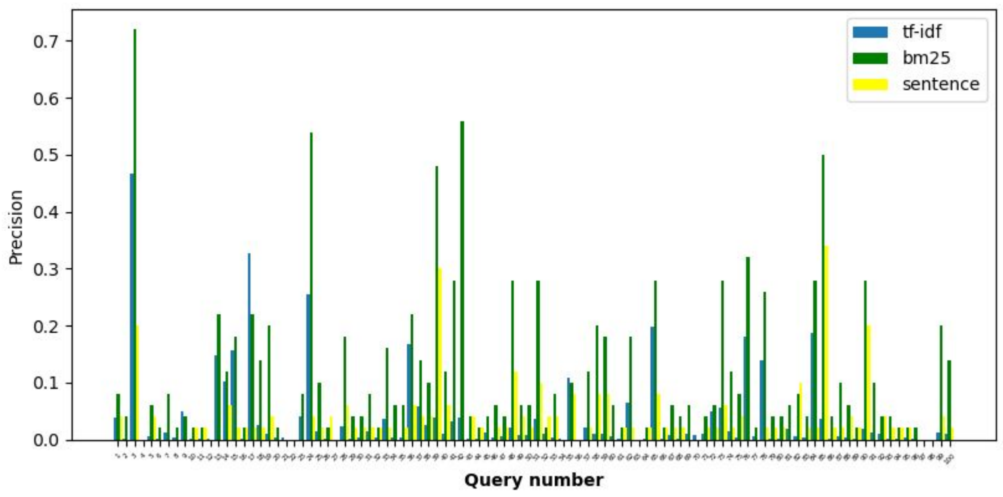
Python 3.0+

Special Libraries -

```
BeautifulSoup (for document and query formatting)
json (to save dictionary results for different scores and storage)
gensim.summarization.bm25 (implementation of bm25 ranking)
nltk (tokenization and preprocessing)
re (for regular expression processing)
```

[V] Evaluation

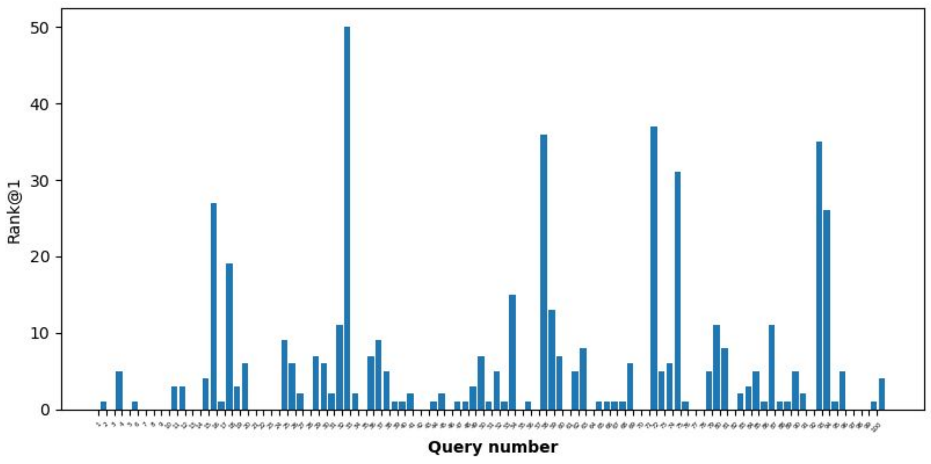
The tf and idf scores obtained from the baseline model are saved in the 'generated' folder.
The combined precision results of the baseline tf-idf and two bm25 models have been illustrated in the plot below.



The blue bars indicate the precision values of the baseline model. The green bars indicate the BM25 scores, and the yellow bars indicate the precision scores for the sentence ranking model for the same query. It can be seen that the precision scores returned by the BM25 model are equal to or higher than the corresponding tf-idf scores for each query. Sentence ranking typically does not perform so well in most cases.

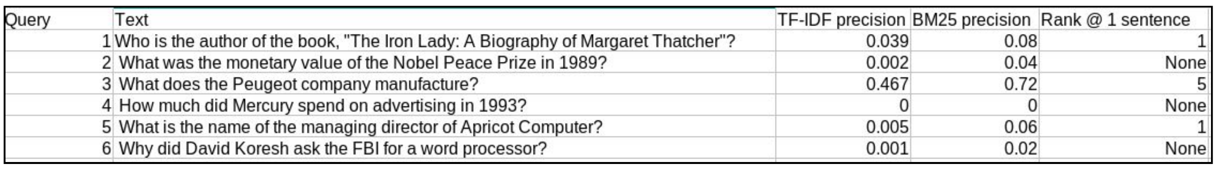
The mean precision for the tf-idf model is: 0.034909999999999997
The mean precision for the tf-idf model is: 0.11119999999999988

The rank of the first relevant sentence per query has been plotted with the query index below (zero indicates no relevant sentence found in the top 50).



The corresponding mean relevance ranking score MRR over the 100 queries is
MRR: 0.31023732262306114

The summarised results for all the queries are present in ‘results.txt’’ in the format <query index, tf-idf precision, bm25 precision, rank@1 sentence> (head illustrated in Figure below)



Query	Text	TF-IDF precision	BM25 precision	Rank @ 1 sentence
1	Who is the author of the book, "The Iron Lady: A Biography of Margaret Thatcher"?	0.039	0.08	1
2	What was the monetary value of the Nobel Peace Prize in 1989?	0.002	0.04	None
3	What does the Peugeot company manufacture?	0.467	0.72	5
4	How much did Mercury spend on advertising in 1993?	0	0	None
5	What is the name of the managing director of Apricot Computer?	0.005	0.06	1
6	Why did David Koresh ask the FBI for a word processor?	0.001	0.02	None

[VI] Analysis

From the evaluation results in section [V], it can be seen that BM25 precision values for each query are equal to or much higher than those of the corresponding tf-idf precision scores. Unlike tf-idf, which is purely term score specific, BM25 is term scoring *in relation to the query*. They are not technically directly comparable, though. BM25 is based on a probabilistic retrieval framework wherein a randomly picked document D would have term t with a probability of idf(t)/N (N = corpus size).
BM25 also gives better relevance for shorter or longer documents than tf-idf+cosine similarity.

The commonalities in these are that both are Bag-of-words models, and they do not capture the position of the terms in text, contextual proximity, and the semantics. There are various variants of both these models according to how they calculate the respective tf, idf and total scores.
In the current project, better precision is obtained by the BM25 ranking (note though that only 50 documents are returned by BM25, so perhaps average precision till rank 50 might be a better metric for comparison).

As mentioned above, one of the shortcomings of the tf-idf and BM25 models is that it does not consider the context and proximity of the term occurrence. Hence, this can be analysed by a sentence-based retrieval system. The idea behind this is that a sentence containing all (or most of) the query terms will be more contextually relevant than a document containing the same terms because of a higher word overlap. But because sentences are smaller, the performance of sentence retrieval is typically found to be worse. Different techniques like smoothing, pseudo-relevance feedback etc. have been researched in sentence retrieval to improve the model performance.

Overall analysis:
It is observed that the precision scores for all the three implementations are by themselves very low. This, however, can be attributed to the corpus more than the retrieval method. We have a very limited corpus where the query answers might probably not even be present. Also, the query evaluation that utilises regular expressions restricts the format of the answer, thus reducing the precision even more.
In conclusion, we have managed to develop a baseline model for information retrieval and compared it with two other methods to obtain a better relevance and ranking of the retrieved documents.

***** THANK YOU *****