IPR project I

Group 24, Diogo Moura 86976, Pedro Pereira 90766, Vasco Morganho 81920

1 OUR SOLUTION

The inverted index was implemented as a python dictionary, with the several terms as the keys, and list postings as the values. The postings are just conventional postings, in which the first two positions in the list are used to store the term's total frequency across all documents and the term's inverse document frequency respectively. This was done because we wanted to increase efficiency in retrieval operations.

We also decided to store each topic in a dictionary, in which the topic numbers are the keys, and the values are a list of pairs (term, term frequency). We did this to increase the accessing speed to the topic information. Having a topic in disk would require an I/O operation every time we wanted get topic information very slow and impractical. We also store the information regarding the relevance feedback, for each topic in a dictionary.

As suggested in the project FAQ, we only indexed the documents present in the qrels files to reduce the ammount of time it takes to index the collection and query retrievals for a specific topic. We consider the judged documents in the qrels files as the whole universe for that topic. It also means that metrics such as BPREF, MAP, precision and recall will be in function of the documents in qrels.

The text processing options used are: lemmatization, stop word removal and number removal. For the rest of report whenever text processing is mentioned, it refers to applying all of the options mentioned above.

We used the following scoring function for ranked retrieval: vector product of the TF-IDFs, BM25 and RRF. In the evaluation function we calculate: precision recal, MAP, BPREF and precision-recall curve.

2 QUESTIONS TO EXPLORE

a) Characterization of the document collection *D* and topic collection *Q*. Distribution of informative terms in *D*

.

The text processing removes stop-words, punctuation and lowercases all the words. Therefore, the text-processing doesn't increase the absolute frequency of the relevant words-distribution, it only increases their relative frequency since we remove stop-words, numbers and punctuation.

Before text processing, the most frequent terms in *D* are punctuation and stop-words. This may reduce the performance of our information retrieval system since it uses stop-words and punctuation with no actual relevance, introducing random noise in our query that may alter our results in a negative way (figure 1).

We can see at figure 14 that this problem is solved with text processing.

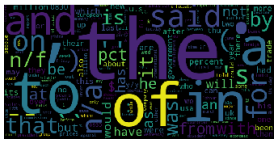

Figure 1: Word Cloud without text processing

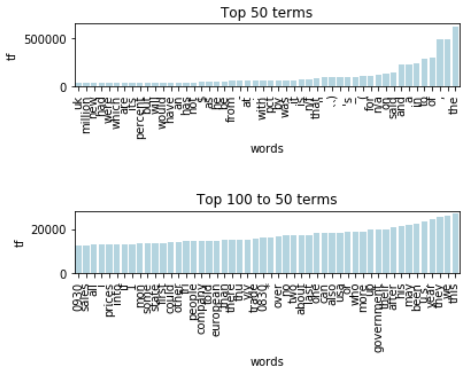

Figure 2: Term frequency distribution without text processing

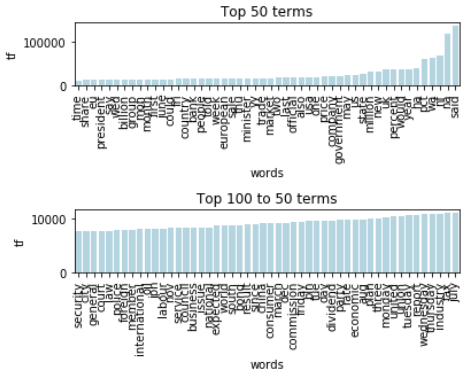

Figure 3: Distribution of term frequencies with text processing

We can also see at figure 2 that the top terms without text processing are much more frequent than without text processing at figure 3.

Even thought this may not affect largely our ranking retrieval model, since we use measures that compensate for this, such as

1st delivery of the Information Processing and Retrieval Class (IST 2020), D. Moura, P. Pereira, V. Morganho, October 2020, Lisbon, Portugal

https://https://tecnico.ulisboa.pt

TF-IDF, this will have a great impact on our boolean retrieval model. Even with 20% of relaxation, much more documents will be retrieved this way, many of those may not even be relevant for that topic since the boolean retrieval is being based on stop-words/punctuation.

At figures 4 and 15 (Appendix), we can also visualize that the memory occupied by our index will be much smaller with text processing since the number of postings drastically decreases, particularly for terms with frequency 1-2 but also for the ones with the biggest frequency.

In summary, the term distribution with text processing is drastically changed which accounts for benefits in performance and memory occupied of our IR system.

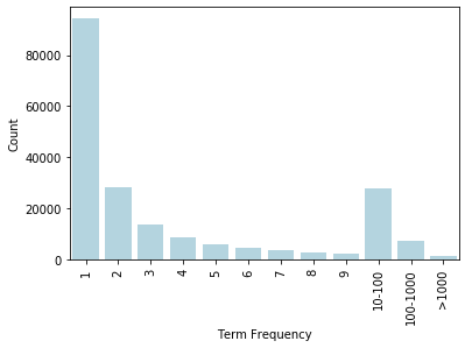

Figure 4: Count of term frequencies distribution without text processing

Another fact is that there are also overlapping terms among the different topics. We counted in how many topics each term appears. The we calculated the average number of overlapping terms, 23.4.

b) Characterization of the performance of the IR system.

For the full test collection, with preprocessing we obtained the following results:
- Processing Time: 6113s
- Processing Space: 0.0MB
- Indexing Time: 130s
- Indexing Space: 6133MB
- Total Time: 6328s
- Index Space: 478021 terms

The fact that the processing space is 0MB is due to the fact that all memory allocated during processing phase is freed afterwards. The following values are from the pre-filtered collection, with only the documents in qrels, for k = 10, and the query used is one containing all topics.
- Processing Time: 714s
- Processing Space: 0.0MB
- Indexing Time: 9s
- Indexing Space: 500MB
- Total Time: 730s
- Index Space: 118985 terms
- Average Ranked Query Time with TF-IDF: 0.0537s
- Standard Deviation for Ranked Query Time with TF-IDF: 0.0229s
- Average Ranked Query Time with BM25: 0.093s
- Standard Deviation for Ranked Query Time with BM25: 0.021s
- Average Ranked Query Time with RRF: 0.162s
- Standard Deviation for Ranked Query Time with RRF: 0.032s
- Average Boolean Query Time: 0.206s
- Standard Deviation for Boolean Query: 0.011s

The choice to obtain these values by querying the pre-filtered collection was made because, the entire was taking too much time and memory to index, which degraded the testing PC performance drastically.

c) Analysis of the impact of *k* on the retrieved documents solutions.

Looking at the graph of figures 5 and 6 that relate the average boolean query result size with different values of for topics R106 and R108, it is easy to see that, for topic 106, the queries stops returning documents at k=24, whereas, for R106, documents stop being retrieved at k=7. Therefore there k should depend on the topic. Imagine that a fixed k threshold was used (e.g.k=7 by looking at figure 16), then for a topic like a lot of documents would not be retrieved.

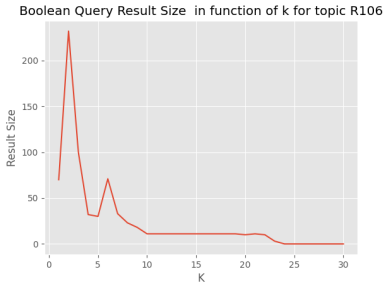

Figure 5: Boolean query result size for k for topic R106

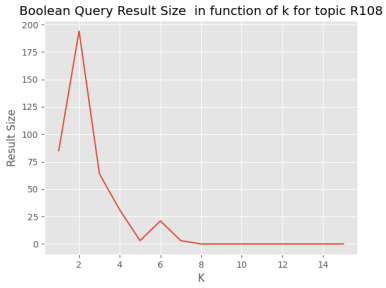

Figure 6: Boolean query result size for k for topic R108

d) Analysis of the IR system with precision and recall given a specific *p*

Recall is the percentage of the relevant documents (positives) that are retrieved (true positives). (tp/(tp+fn))

Precision is the percentage of retrieved documents (tp+fp) that are relevant (tp). (tp/(tp+fp)).

For a big p the precision decreases since we are retrieving more documents with smaller ranking, so they are less likely to be relevant. On the other hand, since we are retrieving more documents we will retrieve more true positives so our recall will increase.

For a small p the precision increases and recall decreases, since we will retrieve documents that are almost for sure positive (the ones with the best ranking) but on the other hand we will retrieve very few documents so the recall will decrease due to being retrieved a smaller quantity of documents.

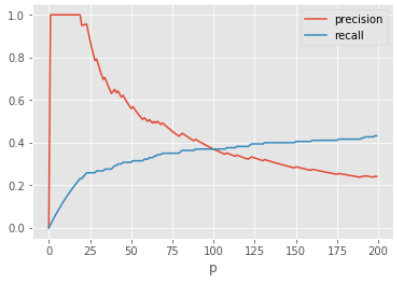

Figure 7: Precision and Recall as a function of p for topic R115

e) Analysis of p for minimizing false positives, minimizing false negatives, or maximizing true positives.

To minimize false positives, a small p should be used, in order to only retrieve the documents that are more likely to be relevant (only the ones with the best ranking).

To minimize false negatives, a big p should be used. With a big p there will be less documents classified as negatives, even if they have a very low score, and therefore the false negatives will decrease.

To maximize true positives we should use a big p. Since we retrieve a lot of documents with a big p, the probability of including most of the positives increases, despite the fact that we may also include many more false negatives in this process and decrease precision.

Minimizing false positives is equivalent to maximizing precision.

Minimizing false negatives and maximizing true positives is equivalent to increasing the recall.

In figures 8, 9, 10 and 11, we can observe precision increases with a small p and decreases with a big p. Recall increases with a a big p and decreases with a small p.

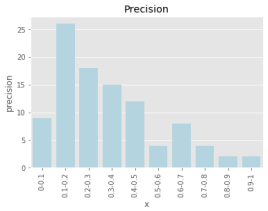

Figure 8: Precision distribution among topics with p=100

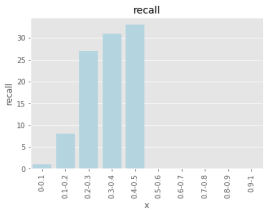

Figure 9: Recall distribution among topics with p=100

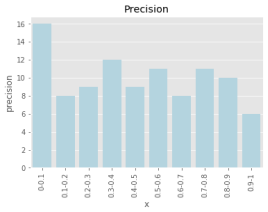

Figure 10: Precision distribution among topics with p=10

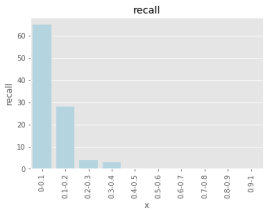

Figure 11: Recall distribution among topics with p=10

f) Analysis of the ranking function and performance among topics, given the Relevance Feedback.

Yes, the performance varies strongly across topics in Q. There are many topics where the precision is 1 up to 9 retrieved documents, but there are also some for which the retrieved documents are all irrelevant and therefore the precision is 0 for all the values between 1 and 10 retrieved documents. This situation is very well illustrated in figures 12 and 13.

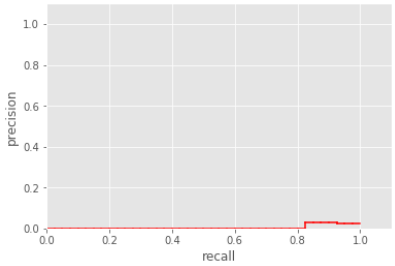

Figure 12: PrecisionRecall-curve for topic R116 with p=40

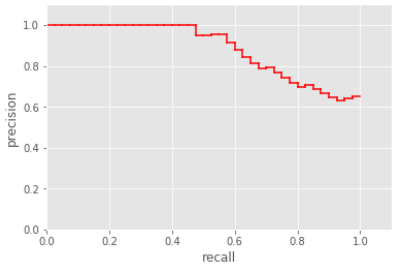

Figure 13: PrecisionRecall-curve for topic R115 with p=40

Given the available relevance feedback we are getting very good results. With vector product of TF-IDF as the scoring criterion and k=10 we obtained an average precision across all documents of 0.498, meaning that more or less 50% of the retrieved documents were actually relevant, but however we only have an average recall of 0.087.

These results are not bad since we are not using an aided retrieval system and a low p, which account for the low recall. However, this is one of the simplest ranking functions, computed by summing the tf–idf for each query term. This solution is biased towards long documents where more terms will appear. There are much better ranking functions which get this into account, most notably Okapi BM25. In the next question we discuss the adequacy of this ranking function in more detail.

g) Analysis of how different text processing and scoring options affect retrieval and of the usefulness of reciprocal ranking to place ensemble decisions

Table 1: Average Measure Values with/without preprocessing for TF-IDF

Measure	Preprocessing	No Preprocessing
MAP	0.369	0.371
BPREF	0.155	0.157
Recall	0.087	0.088
Precision	0.498	0.493

Considering TF-IDF as the scoring method for ranked retrieval. Table 1 shows that having preprocessing or not does not have a big impact on the values of the measure, since they are very similar. This can be explained by the fact that TF-IDF is good at not giving too much importance to values that show up in the documents in excessive quantities (e.g. stopwords), therefore having pre-processing is very similar to not having pre-processing, in terms of retrieval.

Table 2: Average Measure Values with/without preprocessing

Measure	TF-IDF	BM25	RRF
MAP	0.369	0.457	0.446
BPREF	0.155	0.191	0.187
Recall	0.087	0.098	0.098
Precision	0.498	0.557	0.545

Considering the pre-processed index as k = 10 as fixed parameters, we can see that BM25 and RRF perform much better than TF-IDF. Therefore, this can be considered a much better ranking function.

Reciprocal rank fusion is a simple method for combining the document rankings from multiple IR systems. That being said, it is very useful for ensemble decisions since it gives us a consensus between multiple scores.

Under this consensus, we can assess whether consensus improves retrieval or, in alternative, there are specific processing options and retrieval models that yield better performance.

Nevertheless, even tough the consensus may not always yield better results, as in this case, the RRF provides a way of averaging these scores, so we will arrive to a more robust score.

The consensus is a better option than arbitrarily choosing a ranking function between several options, . With RRF we can get a good robust score independently of the document collection and this can even be used as sort of a "black-box" since it will most likely yield good results, independently of the data and reduce the number of situations where our system performs in a very bad way. As a downside is the fact that our queries will take slightly longer to be retrieved, due to the computation of several scores.

3 APPENDIX

Figure 14: Word Cloud with text processing

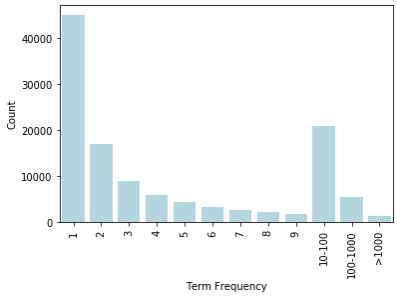

Figure 15: Term frequency distribution with text processing

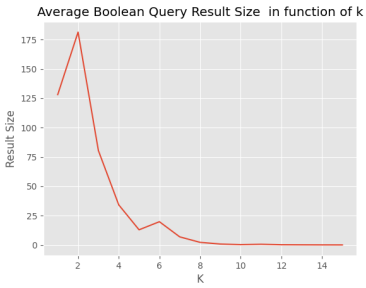

Figure 16: Average boolean query result size for k among all the topics