PRI Project Report

Group 09

Dinis Araújo, ist186406
dinis.araujo@tecnico.ulisboa.pt

Inês Lacerda, ist186436
ines.lacerda@tecnico.ulisboa.pt

Maria Duarte, ist186474
maria.a.duarte@tecnico.ulisboa.pt

Keywords
Information Processing and Retrieval, IR evaluation, indexing, Boolean and vector space models, ranking, and text processing.

1. INTRODUCTION
Filtering has been contributing successfully for document navigation and improving complex querying. It consists of providing a document of interest into a search engine to guide the retrieval of documents from a given collection and our project will be focused on this problem. This first delivery consists of developing an IR system that will rank documents from a specific collection using topics as queries.

2. IMPLEMENTATION
The implementation involves five main structures: the collection, the preprocessing, the indexing, the retrieval models and the evaluation.

2.1 Collection
We were given *D*, *the Reuters Corpus Volume I* (RCV1) Collection. For this delivery we only use *Dtest*, which corresponds to all documents with dates posterior to September 30, 1996. For each document, we extracted the id of the document, the *headline* and the *byline*. For each topic we extracted the *title*, the *description* and the *narrative*.

2.2 Preprocessing
To make our data cleaner and more relevant we preprocessed all topics and documents. First, we removed all the punctuation, then, we lowercased each word, tokenized it, removed all stop words and applied stemming/lemmatization. We decided to use the stemming model because we saw it improved the overall performance of our system.

Table 1 and 2 – 5 most relevant terms for 10 documents in the collection with and without preprocessing

119150	gm	vw	lopez	case	suit
106273	sentenc	jone	year	california	rapist
112549	dominican	puerto	rico	40	fear
106431	connerott	case	remov	prosecutor	children
139503	quaker	gatorad	sale	said	see

119150	,	in	the	GM	.
106273	.	the	in	,	was
112549	.	the	to	in	on
106431	the	.	,	of	to
139503	Quaker	.	Gatorade	sales	said

Table 3 - Average Precision of all documents for each topic with and without preprocessing

	321 – PREPROCESSING	321– NO PREPROCESSING
Boolean Model	0.434	0.262
SV Model	0.413	0.332
BM25 Model	0.464	0.394
RRF	0.457	0.409
CombSUM	0.479	0.375
CombMNZ	0.247	0.155

Based on tables 1 and 2, we can see how our preprocessing impacts our system. In table 2 most terms correspond to either stop words or punctuation, which are not informative terms. These are not helpful for our system in identifying relevance and just jeopardize the system overall performance, as can be seen on table 3.

Furthermore, we also tried removing all small words (i.e., words smaller than four letters). We wanted to see if eliminating these small words that could be irrelevant (e.g., “see”, “hear”) would improve the performance, but unfortunately it did not improve our model. We also tried adding synonyms in order to expand queries, to do this we used the *PyDictionary* module, but unfortunately it also did not improve our model.

2.3 Indexing
The *Indexing* function returns the inverted index, the indexing time and space required. For the inverted index, we created a dictionary containing the documents of the collection and for each document the number of times a term appears. We calculated the index manually without using any python library. For the indexing time, we used the python module *time*, and for the space required we simply used sys.*getsizeof(index).*

2.4 Retrieval Models
2.4.1 Boolean Model
The *Boolean Query* function returns the matching documents for a query of a given topic, using the *Boolean IR model*. The function *extract topic query* returns a query that corresponds to the top-k informative terms of a topic *q*. This function creates a *TFIDF* using the terms of the query as vocabulary and indicates the importance of a term in the topic in relation to the collection. The creating of the *TFIDF* was done using *TfidfVecorizer* from the *Sklearn* library. After creating the *TFIDF*, we picked the top *k* informative terms, created a query and returned all document ids that contained at least 80% terms from the query.

2.4.2 Ranking models

As ranking models, we decided to implement the *Vector Space Model*, *BM25*, *RRF*, *CombSUM* and *CombMNZ*. For each we analyzed the results to understand which model would improve the overall performance of our system.

2.4.2.1 Vector Space Model

VSM is an algebraic model for representing text document as vectors. This model is implemented in the *ranking* function. First, we created the *TFIDF*, the same way we did in the *extract topic query* function, but now our collection also contained the query terms. After, we calculate the *cosine similarity scores* between the topic and each document in the collection based on the *TFIDF* and returned the top *p* documents with the highest scores.

2.4.2.2 BM25

To implement BM25 we simply used the *BM250Kapi* from the *rank_bm25* python library.

2.4.2.3 RRF

To implement *RRF* we simply used the formula given in the project assignment:

$$RRF_{SCORE}(d \in D) = \sum_{f \in D} \frac{1}{50 + rank(f_d)}$$

2.4.2.4 CombSUM

To implement *CombSUM* we simply did:

$$CombSUM(d) = \sum_i s_i(d)$$

Where *i* is each one of our models (*Space Vector Model*, *BM25* and *Boolean Model*). Using all the three models gave the best results.

2.4.2.5 CombMNZ

To implement *CombMNZ* we simply did:

$$CombMNZ(d) = |\{i | d \in Rank_i\}| \cdot \sum_i s_i(d)$$

Where *i* is each one of our models (*Space Vector Model* and *BM25*).

3. EVALUATION

Since this collection has a computational expensive size, we decided to use subsets during the development and evaluation of our system. For this purpose, we have developed a python script, *script.py*, which automatically retrieves all documents' subsets.

We decided to create a subset for each topic that only contains the documents classified as non-relevant and relevant in the *Rtest* collection, because we wanted our system to return only documents that could be considered as relevant or irrelevant for evaluation purposes. This way, we could improve our system based on the evaluation results for a given topic and compare results from different topics. These subsets also made it possible to run our program in an acceptable time because otherwise it would take too long to evaluate the complete *Dtest* collection.

3.1 Initial Parameters

We ran multiple experiments with the following parameters[1]: 1) weights for the topic information based on its structure (i.e., weights for the title, the description and narrative). We did this because we wanted to see if giving more importance to a specific group of terms of the topics would result in better results; 2) a flag for the removal of stop words, to see if removing the stop words would benefit the system; 3) the number of ranked documents; and 4) a flag for deleting the terms of sentences that included the word "irrelevant" in the topics' descriptions. We did this because we did not want our model to associate those terms to the documents as they were classified as irrelevant.

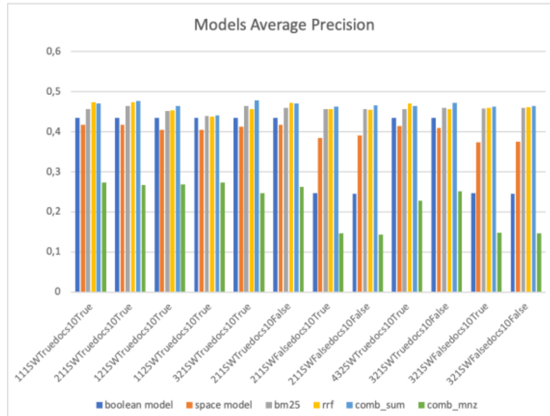

Figure 1. Average of the precision (of each topic) for every model

In figure 1, we have the most relevant results from running multiple experiments with different parameter. For all experiments we used 10 as the number of returned ranked documents, and 3 as the number for the *k* terms in the Boolean query. We used these constant values in order to compare how other parameters influenced the final performance of our system. By choosing small values we were able to run a higher number of experiments without it taking too much time.

Based on figure 1, can be seen that the model and parameters with the highest precision are (i.e., 48%): *CombSUM* model; removing stop words; giving higher priority to the title, then the description and finally the narrative (e.g., 3, 2, 1); and removing the topic terms that appear in sentences with "irrelevant". We believe *CombSUM* outperforms the other models because it combines knowledge from the output of three models (i.e., the Boolean model, Vector Space model and BM25) without considering the position in the ranking but instead the score of the ranking. From the three baseline models, *BM25* was the best model. BM25 is a probabilistic model that supports ranking, thus, it performs better than a simple Boolean model. Furthermore, BM25 incorporates length normalization and term frequency and is based on probability theory, thus, has better theoretical foundation than the vector space model.

After knowing the best parameters for our models, we decided to analyze in more detail each model using the same parameters.

3.2 All Models: Precision for Each Topic

When running our IR system, we realized that the performance strongly varied across topics. We decided to analyze in detail how each baseline model behaved for each topic. After, realizing each baseline model was not constant throughout the topics, we

[1] we also added a parameter that would insert synonyms for each term in the topic, but we did not add it as a parameter for the analysis (e.g., graphs) as it decreased the performance our system

decided to add models that could handle ranking differences and plan consensus. We implemented *RRF*, *CombSUM* and *CombMNZ*, like it was described in sections 2.4.2.3, 2.4.2.4 and 2.4.2.5 respectively.

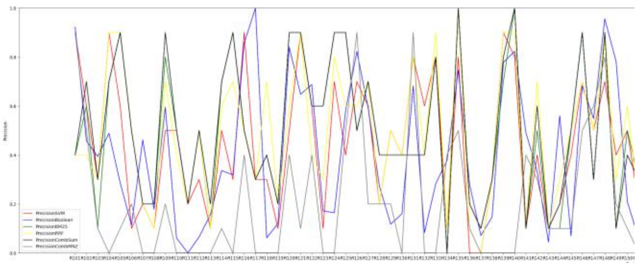

Figure 2. Models Precision for Each Topic: First 50 topics

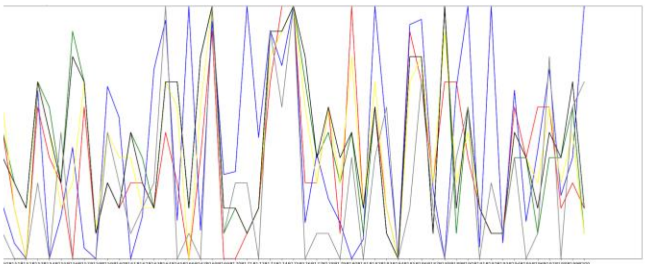

Figure 3. Models Precision for Each Topic: Last 50 topics

As can be seen in figures 2 and 3, the precision in all models varies a lot. No model seems to work for all topics, however, *CombSum* had the best performance, with an average topic precision of 48%.

As can be seen in Figure 3, topic 75 has the maximum precision (i.e., 100%) and topic 84 has the minimum precision (i.e., 0%) for all models. After analyzing the content of both topics and some documents that are considered relevant by the *Rtest*, we can conclude: 1) Some topics have descriptions that describe which documents' themes are considered relevant and irrelevant (e.g., "Documents about military or political espionage would be irrelevant"). Based on this information, we deleted the sentences that describe the irrelevant documents so that our models would not consider these as relevant. 2) Some documents' descriptions use synonyms or words related to other words (e.g., the words "Moslem" and "Christianity" to describe a document that is relevant for a topic about "religion"). Therefore, we decided to add synonyms for each term in the topics in order to expand our query but it did not improve the precision of our models, in fact it decreased it.

3.3 Boolean Model: Precision, Recall, F-score in Function of K Terms

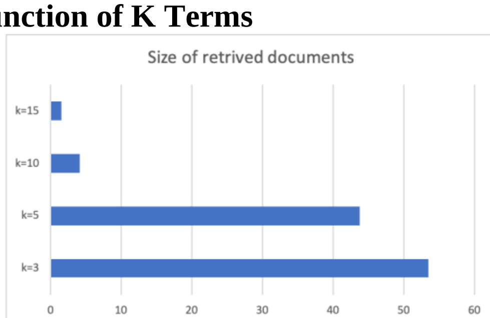

Figure 4. Size of Retrieved Documents in Function of Number of K Terms

We decided to see how the number of *k* terms affects the size of the retrieved documents, as can be seen in figure 4.

Just like expected the higher the number of *k* terms, the lower is the size of retrieved documents. The reason behind this is that the higher the number of *k* terms, the more specific is the query and the system finds less documents for that specific query.

To further analyze our system, we decided to see if we should use *k=3* (fixed threshold) or if it should depend on the provided topic document (*q*-specific). Figure 5 shows the precision results for each topic when *k* is 3, 10 and 15 and we concluded that using a higher *k* only improves very few topics (e.g., from k=10 to higher, the Boolean model does not have a good performance) and sometimes the improvement is really small.

We can conclude that changing the k value based on the topic would not be efficient, since it would require much more processing to find the best *k* value, and the *Boolean Model* only has satisfactory results with *k* as 3 and 5.

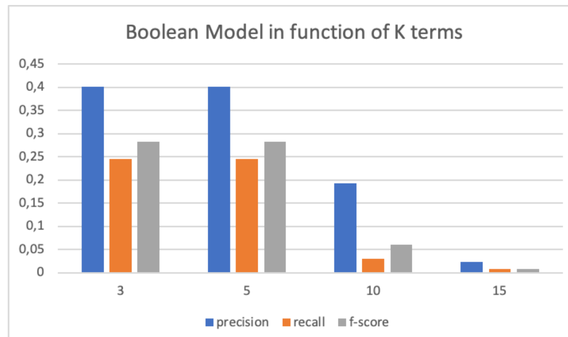

Figure 5. Precision, Recall and F-score of the Boolean Model for different K terms

Based on figure 5, can be seen that if *k* keeps increasing, then the precision, recall and f-score decrease. Finally, we can conclude that the optimal value for *k* is 3, because this way the system can process shorter queries and has a higher performance.

3.4 BPREF

We decided to analyze whether our IR system was ranking relevant documents above irrelevant ones. Due to timing and memory restrictions, we decided to execute our system with the subsets and not the full *Dtest* collection.

To analyze *BPREF* we ran our system for different *p* values, so that we could see how the number of retrieved documents affected our *BPREF* score, as can be seen in figure 6.

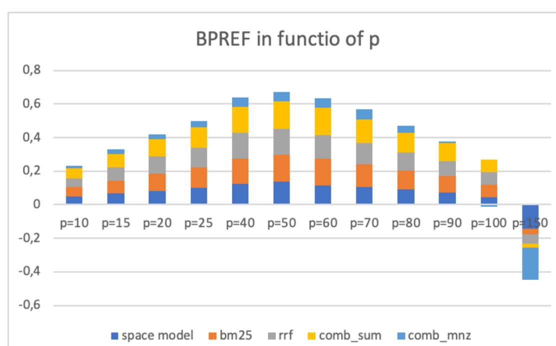

Figure 6. BPREF scores for different *p* values

The average number of relevant documents per topic is 74.73 and the average number of documents per topic is 375.56.

As can be seen in figure 6, when *p* value is between 10 and 50 (i.e., our models return 10 and 50 documents), the BPREF scores increase. Until *p* is 50, our BPREF scores increase which means that our system ranks more relevant documents above irrelevant ones. Furthermore, the model with the best BPREF scores is the *CombSUM* model.

When *p* is equal and higher than 60, the scores decrease. The reason behind this is that at some point the *count* value gets higher than the *min(number of relevant documents, number of irrelevant documents retrieved)*, thus, the value subtracting 1 will be higher than 1, which results in a negative value overall:

Bpref += 1 - (count / min(relevant_len, len(docs_id) – len(rel_retrieved_rank)))

There are cases where the BPREF scores are negative because it will be the sum of multiple negative values.

3.5 MAP

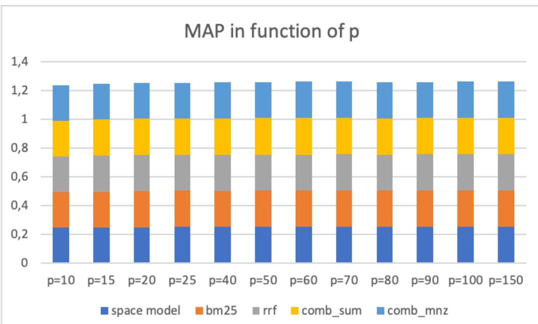

Figure 7. MAP scores for different p values

Figure 7 shows the Mean Average Precision for all topics of each model. For each *p*, the MAP is constant because normalizes the average precision and is used to prevent an unfairly suppression of our ranking scores when the number of retrieved documents surpasses the number of relevant documents.

3.6 *P* relation with Precision and Recall Measures

Table 4 - Precision in function of p

Model / p=	1	3	5	10	15
Boolean	0.43	0.43	0.43	0.43	0.43
Vector Space	0.47	0.46	0.44	0.41	0.40
BM25	0.49	0.47	0.48	0.46	0.44
RRF	0.53	0.49	0.50	0.47	0.45
CombSUM	0.53	0.51	0.50	0.47	0.46
ComMNZ	0.33	0.28	0.27	0.26	0.24

Table 5 - Recall in function of p

Model / p=	1	3	5	10	15
Boolean Model	0.26	0.26	0.26	0.26	0.26
Vector Space	0.01	0.02	0.04	0.08	0.11
BM25	0.01	0.02	0.05	0.09	0.13
RRF	0.01	0.03	0.05	0.09	0.13
CombSUM	0.01	0.03	0.06	0.09	0.14
CombMNZ	0.00	0.01	0.02	0.03	0.05

To any specific *p* value, our IR system is more optimal at providing precision guarantees than recall ones. The reason behind this is that our IR system generates more false negatives than false positives. With these two tables at hand we are able to generate a system based on user's preferences for minimizing false positives, minimizing false negatives or maximizing true positives. If a user chooses to minimize false positives, then we are looking for the value of *p* where the precision is maximized (i.e., when *p* is equal to 1). If a user chooses to minimize false negatives, then we are looking for the value of *p* where the recall guarantee is maximized (i.e., when *p* is equal to 15). In order to calculate which value of *p* would be optimal for maximizing the number of true positives, we have created another auxiliary table with the value of true positives for each of the p values. With this information we can conclude that 15 is the p value for which a user can maximize the number of true positives.

Table 6 - True Positives

	1	3	5	10	15
True Positives	2412	2850	3275	4273	5211

3.7 Time and Space Complexity

After thoroughly measuring our IR system processing all our documents' subset, with a total of 37556 documents for 100 topics, we have generated the following table:

Table 7 - Time and Space of Models

	Time / Sec	Space / Byte
Boolean Model	11.522	6400
Space Vector	11.160	7200
BM25	5.097	7200
RRF	0.300	7200
CombSUM	0.585	7200
CombMNZ	0.585	6400

4. Conclusion

From the results obtained, we can conclude that the best baseline model is the *BM25* and the best model that handles ranking differences and plans consensus is the *CombSUM*.