Assignment 3

Filipe Pires [85122], João Alegria [85048]

Information Retrieval
Department of Electronics, Telecommunications and Informatics
University of Aveiro

December 10, 2019

1 Introduction

This report was written for the discipline of 'Information Retrieval' and describes the implementation and evaluation of a ranked retrieval method that uses the indexes created with the solutions developed for the previous assignments.

We include the correction of design flaws of the delivery done prior to this one and the updates applied both to the text corpus indexation and to our class diagram. We also provide the instructions on how to run our code.

Along with the description of the solution, we also present the results of our calculations to evaluate the solution and determine its efficiency according to the metrics proposed for this last assignment (*I*). All code and documentation is present in our public GitHub project at `https://github.com/joao-alegria/RI`.

2 Re-Indexing the Corpus

The first big challenge in this iteration was the fact that now it was necessary to index not only the titles of all the documents, but also their abstracts, a short description of the document content. This addition turned out to be quite challenging in terms of computational power necessary to accomplish this feat, since the abstract is considerably larger than the title, being many times 4 times as large.

There was 2 approaches when adding the abstract processing to our indexing process. The first one consists in processing the title and the abstract in different fashions, treating each one as independent and creating on index for the title and another for the abstract. This approach has the advantage that is much more simple and direct the process of verifying if a given term appeared in the title, in the abstract or in both, since they are stored in different structures. A clear disadvantage is the disc space necessary to store all this information, those different structure will generate some redundancy and repetition unnecessary.

The second strategy is the concatenation of both fields and the processing of the resulting text as a whole. The advantages and disadvantages of this strategy is almost the inverse of the precious one, since the advantage is no redundancy and the disadvantage is the impossibility of distinguishing where the term occurred. It's obvious that when combining the fields in and using only one structure to store all the information the redundancy and repetition mentioned in the previous approach is removed by this method. On the other hand, if the distinguishing if the term occurred in the title,in the abstract or in both is necessary, this approach is not the ideal since that information is lost when adding one field to the other.

The implementation strategy that we used to accept the abstract in our index was the second approach, the simple concatenation of the title with the abstract, since in our context the position where the terms occurred was not necessary. This enabled us to process both at the same time without changing anything in the already created indexing pipeline. This new feature, as already mentioned, needs significantly more computation power, which is confirmed from the fact that previously the indexing process of the entire corpus took approximately 20 minutes, while with the abstract takes close to 80 minutes, a 4 fold increase that was to be expected since the text to process in this iteration is also close to 4 times larger.

3 Ranked Retrieval of Relevant Documents

.....

4 Evaluation and Results Discussion

After calculating the relevant documents for each query, a assortment of performance metrics were calculated. The metrics chosen for this purpose was Precision, Recall, F-Measure, Mean Precision at rank 10, Mean Precision, Normalized Discounted Cumulative Gain as well as all the averages in between the queries performed for all the previous metrics. Time related metrics were also obtained, such as Query Throughput and Median Query Latency, for these we choose to use an auxiliary linux command line program called *time*, which is able to summarize the system resources needed to run a specific program.

4.1 Metrics

This section aims to give a brief explanation on what each metric is and what's the true meaning of the result it provides.

Precision is one of the most used metrics in information retrieval and is characterized by dividing the correct retrieved documents by the totality of retrieved documents. This metric ends up providing the percentage of the retrieved documents that are really relevant for the user.

Recall is another quite popular metrics when accessing systems performance, specially in the information retrieval area. This metric consists in dividing the correct retrieved documents by the totality of ideal correct documents. The result is the percentage of correct documents that the system could present.

F-Measure is a metric many times used to represent the system performance with only one value. This is accomplish by the combination of the two previous metrics, performing the harmonic mean between the two metrics ($\frac{2*P*R}{P+R}$).

Mean Precision is a variation of the already mentioned Precision metric. The formula is exactly the same, the difference is that in this case the precision is calculated in the various ranks, i.e., every new retrieved document the precision is calculated and then the average of all the intermediate precisions is calculated, giving the Mean Precision of the query result.

Mean Precision at rank 10 is yet another specification of the Precision metric, being the Mean Precision metric just explained but calculated just until rank 10, meaning just until the 10^{th} value.

Normalized Discounted Cumulative Gain is a more complex metric that uses a graded relevance scale of documents to evaluate usefulness/gain of the returned results. The core idea with this metric is that relevant documents that are lower in the list should be penalized, since the important documents should be the first results provided. This is accomplished by the formula:

$$DCG = \sum_{i=1}^{p} \frac{rel_i}{\log_2(i+1)} \tag{1}$$

The normalization is then achieved by dividing this value by the ideal one, i.e., having the relevances in order, the first documents being the most relevant and so on.

Query Throughput is a time related metric that indicates how many queries can be processed in one second. It's obvious that in the retrieving process it's ideal that the information gathering and ordering should take the least amount of time possible, so the higher the query throughput the better.

Finally, the Median Query Latency, as the name suggests is the average time is query took to process. Following the same logical line of the previous metric, the lower this number is the better, since it means that more queries can be processed in less time.

4.2 Results

4.3 Discussion

5 Implementation Limitations

6 Conclusions

After completing the assignment, we drew a few conclusions regarding our solutions and the whole concept of .....

The biggest challenge we faced was .....

From this assignment, we take .....

The overall perspective of our performance regarding the project is .....

References

1. S. Matos, *IR: Assignment 3*, University of Aveiro, 2019/20.