Project 2: MuGle

SUMMARY REPORT

OVERVIEW

This report shows the backgrounds of our Jaccard, TF-IDF, and BM25 (Bonus Credit) implementation.

Members

1. **Krittin** **Chatrinan** ID 6088**022**
2. **Anon** **Kangpanich** ID 6088**053**
3. **Tanawin** **Wichit** ID 6088**221**

ANSWERS

QUESTION 1

TF-IDF w/ Cosine Similarity VS. Jaccard Similarity

Metric: **Relevance**

From *Figure 2* and *Figure 4*, it can be observed that at the same k, TF-IDF yields greater precision and F1 values. This means that for each k value TF-IDF search engine can retrieve more relevant documents proportional to all retrieved documents. Moreover, the average of interpolated Precision Curve of TF-IDF, in Table 2, stays at 0.2972108 while Jaccard stays at 0.1759865 which is significantly lower; therefore, with both indications, the TF-IDF searcher tends to serve the information need of the user better in terms of relevance than the Jaccard one. The statement is bold but it is true because the benchmark data provides the ground truth of relevant documents for each query which is done by humans. Search engines are tested by using benchmark queries. The result will be tested against the ground truth using precision, recall, and F1 values.

The reason why TF-IDF with cosine similarity is better than Jaccard Similarity lies within its characteristic. TF-IDF with cosine similarity does not take each term equally by using Inverted Document Frequency (IDF). In another word, some terms may have higher weight because of its rarity in a corpus.

$$idf_t = \log_{10}(1 + \frac{N}{df_t}) \quad tf_{t,d} = 1 + \log_{10} tf_{t,d}$$

The other reason is that TF-IDF with cosine similarity also considers the weight of each document based on the frequencies of terms that occur in the query. Multiplying both TF and IDF creates a score for a term within a document. Lastly, by treating each document as a vector, TF-IDF with cosine similarity is also advantageous when it comes to comparing documents and a query. If we construct both document and query vectors, we can calculate its norm and calculate the angle between those vectors which is a similarity score between document and query.

However, in Jaccard Similarity, it only works on binary values indicate whether a term is present or not present in a document. Moreover, Jaccard Similarity does not take term frequency and document length into account; therefore, every term and document are the same if the term presents in the query. Thus, Jaccard Similarity is inferior when it comes to relevance because it cutting too many corners. The only advantage it has is the amount of implementation required.

Metric: **Time Consumption**

As illustrated in *Table 1*, the average computation time of Jaccard Similarity is the highest. This may be caused by the fact that the Jaccard coefficient calculation cannot be precalculated when indexing. Moreover, Jaccard calculation also requires both union operation and intersect operation all of which require a tremendous amount of comparisons that scale up with the factor of the number of terms in a document.

On the contrary, TF-IDF with cosine similarity has an advantage by creating an inverted term index and precalculating both TF and IDF scores before searching time. By simply multiplying TF and IDF together, TF-IDF searcher can compute every score for every term and every document before searching time. When searching, the TF-IDF searcher will calculate the TF-IDF score for only the query. Then it will focus on creating document vectors focusing on terms that present in both query and documents. With all this, search time improves significantly when compare to Jaccard searcher.

Similarly to the BM25 searcher, to calculate RSV in searching time, BM25 requires only the precalculation of IDF of terms. BM25 also requires only inexpensive operations such as multiplication, addition, and division to get its score similar to TF-IDF.

To conclude, the Jaccard searcher is also the worst in terms of time. Both TF-IDF and BM25 are similar in terms of some processes and use less time than Jaccard.

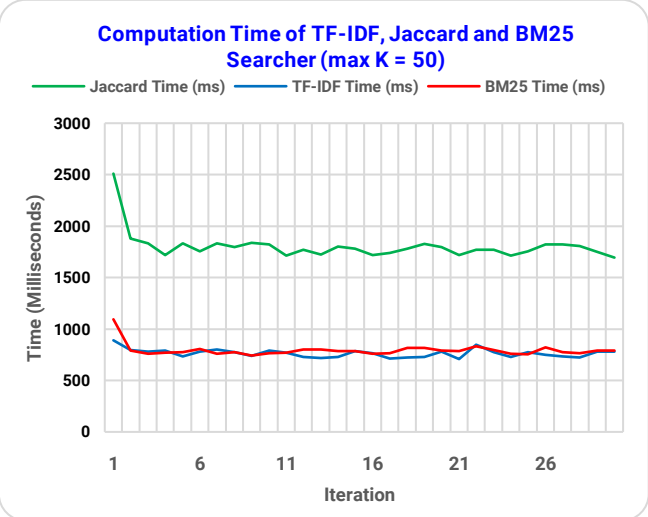

Figure 1. Time Consumption of TF-IDF, Jaccard and BM25 Searcher with the total iteration of 30.

TABLE 1. AVERAGE COMPUTATION DURATION OF TF-IDF, JACCARD & BM25 SEARCHER.

Searcher	Jaccard	TF-IDF	BM25
Average Duration (ms)	1802.466667	763.6333333	793.3333333

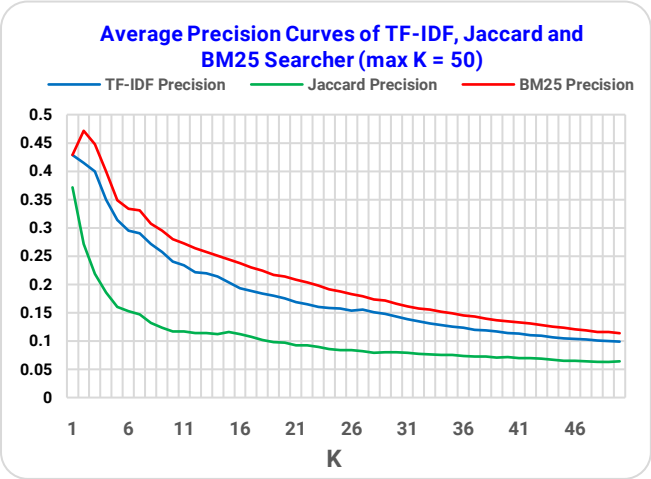

Figure 2. Average Precision Curves of our implementation of TF-IDF, Jaccard, and BM25 limited at K = 50. Tested by using LISA dataset. Showing differences in the precision decrease rate of three systems.

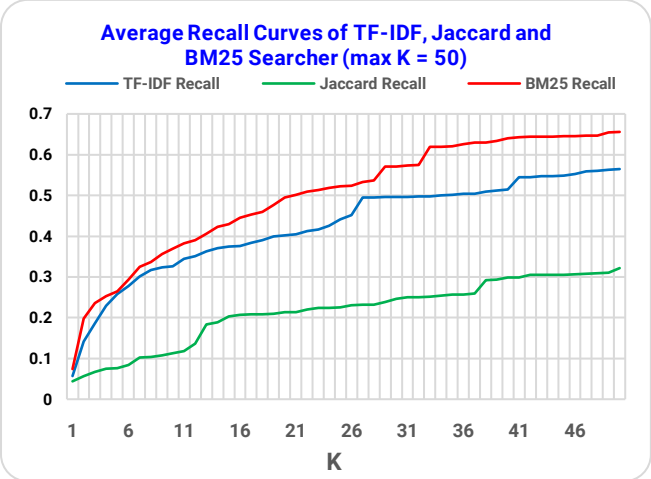

Figure 3. Average Recall Curves of our implementation of TF-IDF, Jaccard, and BM25 limited at K = 50. Tested by using the LISA dataset.

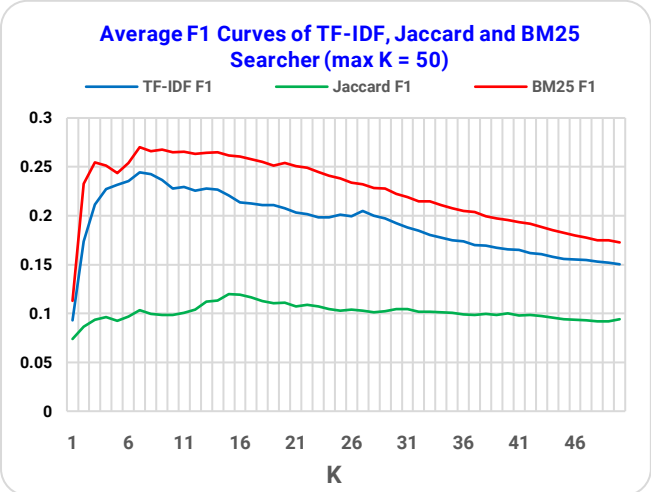

Figure 4. Average F1 Curves of our implementation of TF-IDF, Jaccard, and BM25 limited at K = 50. Tested by using the LISA dataset.

QUESTION 2

Precision, Recall and F1 Curve

Here are our observations from Precision curves in *Figure 2*, Recall curves in *Figure 3*, and F1 curves in *Figure 4* from all three search systems.

1. **K value is a direct variation to Recall value** – As illustrated in *Figure 4*, the recall value increases as K value increases. This implies that both values are a direct variation to each other. Likewise, in the definition of Recall which is a proportion of retrieved relevant documents to all relevant documents in the corpus, retrieving more documents could increase the chances to get relevance documents; therefore, Recall always increases when retrieving more documents.

$$Recall = \frac{Total\ Retrieved\ Relevant\ Documents}{Total\ Relevant\ Documents}$$

2. **Different Precision at the same Recall** – As illustrated in both *Figure 2* and *Figure 4*, it can be observed that the precision values at the same recall are significantly different between Jaccard and TF-IDF. Moreover, both also show that TF-IDF consistently yields better precision in every recall value which means that TF-IDF's search results have more retreived relevant documents proportional to all retrieved documents.

$$Precision = \frac{Total\ Retrieved\ Relevant\ Documents}{Total\ Retrieved\ Documents}$$

3. **TF-IDF's F1 is consistently greater than Jaccard's F1** – Due to the fact that F1 is a harmonic mean between precision and recall, in other words, it quantifies both precision and recall into a value, larger F1 value means larger precision and recall. Therefore, from *Figure 4* alone, it can conclude that TF-IDF search results yield higher F1 values.
4. **Increasing Recall when k is increased** – From *Figure 3*, it can be observed that recall curves from all search systems are initially low but later higher due to more document retrieved. The slop of all recall curves is initially high; however, at higher K values, the slope of all curves is going closing to 0 or flatter.
5. **Decreasing Precision when k is increased** – From *Figure 2*, it can be observed that the precision curve of Jaccard is initially steep which means that it has a higher decrease rate than TF-IDF. Similarly to Recall, both TF-IDF and Jaccard, in high k values, have a flatter slope of the precision curve which can be inferred that the decreasing rate is diminishing with higher K value or Recall.

QUESTION 3

Precision-Recall plot and Search Engine Performance Evaluation

Normally, the performance test of a search engine is observed by the same precision values at the same recall values. From *Figure* 5, it can be observed that the precision values of the TF-IDF model are

higher than the precision values of the Jaccard model. Therefore, it can be summarized from the graph that the search engine using the TF-IDF model can retrieve more relevant documents than the search engine that uses the Jaccard model at the same rate of retrieving documents.

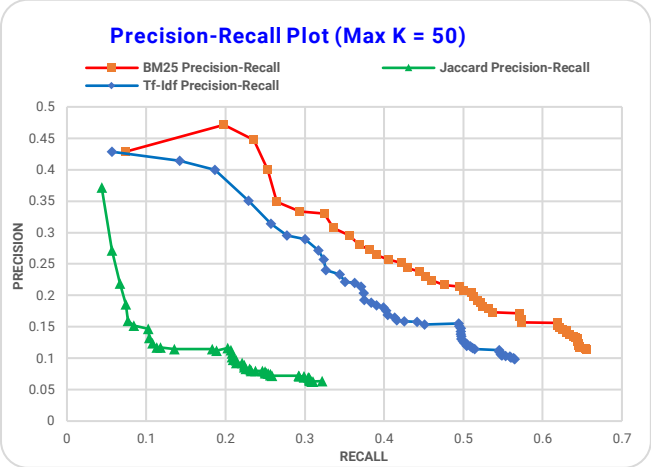

Figure 5. Average Precision Curves of our implementation of TF-IDF, Jaccard, and BM25 limited at K = 50.

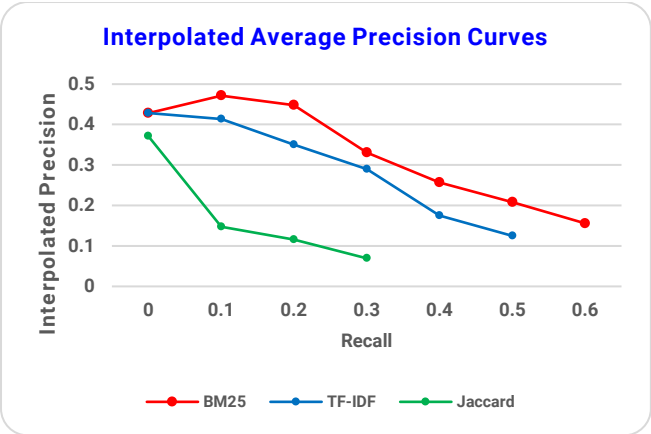

Figure 6. Interpolated Average Precision Curves of our implementation of TF-IDF, Jaccard, and BM25 limited at K = 50.

TABLE 2. INTERPOLATED AVERAGE PRECISION WITH THE AVERAGE OF THE INTERPOLATIONS.

Recall	Ranking Methods		
	BM25	TF-IDF	Jaccard
0	0.428571	0.428571	0.371429
0.1	0.471429	0.414286	0.146939
0.2	0.447619	0.35	0.11619
0.3	0.330612	0.289796	0.069388
0.4	0.257143	0.175714	
0.5	0.208163	0.124898	
0.6	0.155844		
AVG	**0.328483**	**0.2972108**	**0.1759865**

EXTRA CREDIT QUESTION

MyCoolSearcher

Our implementation of MyCoolSearcher is based on BM25, a probabilistic ranking approach. Our implementation starts from `Indexer` class, resides in `TFIDFSearcher.java`, which involves indexing all terms in all documents as well as collecting term frequency. `ProbabilisticIndexer` class, a subclass of `Indexer` class, which involves pre-calculating IDF. The `MyCoolSearcher` class will use an indexed term incidence matrix from the indexer object to calculate Retrieval Status Value (RSV) for every document. With RSV values, the Searcher then sort documents by its RSV. The higher the RSV, the greater. To calculate BM25, we specify 3 tuning variables as follows: $k_1 = 1.2$, $b = 0.75$, $k_2 = 2.0$.

$$RSV_d = \sum_{t \in q} \left[\log\left(\frac{N}{df_t}\right) \times \frac{(k_1+1) \times tf_{t,d}}{k_1\left((1-b)+b\left(\frac{L_d}{L_{avg}}\right)\right)+ tf_{t,d}} \times \frac{(k_3+1) \times tf_{t,q}}{k_3+tf_{t,q}} \right]$$

Why is it better than TF-IDF with cosine similarity and Jaccard similarity?

According to the similarity model, the Jaccard model looks at both documents and queries in the form of the set and then calculates the rate of the term contained in both documents and queries and total term in both documents and queries. In the case of the TF-IDF model, it looks at both the documents and queries in the form of vector space and calculates the cosine from the angle between the documents vector and the queries vector because both term frequency and document frequency are also taken into consideration. So, it has more effective than the Jaccard model.

However, the model that we use in MyCoolSearcher is a probabilistic model named BM25. BM25 looks at both documents and queries in the form of probability and uses criteria based on the existence of each term in documents and documents relevancy. So that is possible to create a condition in the rank calculation more than the TF-IDF Model. In addition, although both models will have the same TF and IDF calculations, the BM25 has additional document length considerations that are different from the TF-IDF model which does not consider document length. Therefore, it can be concluded that BM25 has the highest efficiency because it has more conditions that are considered than the TF-IDF model and the Jaccard model.