ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Text Retrieval using Natural Language Processing Methods Project Report

Project Work Report

June 30, 2020

Group 1d: Francesco Carzaniga, Lorenzo Barbera,
Qiuhong Lai, Simone Barbaro, Wenjie He

Abstract

In this project, different methods (models) like BM25, FastBM25, Sent2Vec, Bert and their hybrid versions are implemented for different text retrieving tasks with different size of database with or without human rating scores. The text retrieving performance is evaluated with different matrices concluding M@L scores, ROUGE scores and NDCG scores.

It is found that based on size of dataset, M@20 can better indicating model performance on small database. NDCG scores can better indicate the model performance given a big dataset and when human scoring is involved. ROUGE-2 score can also be a good reference for model performance.

BM25 and FastBM25 model (Faster version of BM25) is found to have the best text retrieving performance compared with other models. The hybrid version of all the models seems to have worse performance compared with the un-hybrid version. If the model is ”hybrided” with different keyword filtering strategy, the ”or” keyword filtering strategy seems to perform better in the case of big dataset base and ”and” keyword filtering strategy performs better in the case of small dataset base (around 1000 dataset).

Contents

Chapter 1

Project introduction

1.1 Objects

The overall goal of the project is to explore natural language processing methods for retrieving scientific content based on a query and potential keywords of interest. Information retrieval methods is studied and search engine is built step by step. The performance of the engine is evaluated on a specific dataset.

As shown in the figure 1.1, the project start with basic Sent2Vec + nearest neighbor search. Keyword Boolean filtering processing is further developed into the search engine. In addition, by picking up more than one keywords and using logic strategy to get a subset of corpus, Sent2VecHybrid model is implemented. Finally, more hybrid versions of other models are implemented and evaluated.

1.2 Task Introduction

The project is divided into two parts:

- **Part A:** the BM25, Sent2Vec and their hybrid versions retrieving methods are implemented to retrieve abstract using given article title. Ac-

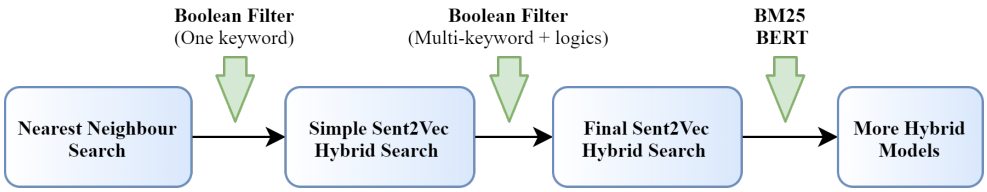


Figure 1.1: Project objects and procedures.

cording to TF-IDF scores, different number of keywords are selected and are used to retrieve with different logic (e.g. "or" and "and" logic). The retrieval performance of various methods is further evaluated with "M@L score" and "ROUGE" score based on a reduced dataset.

- **Part B:** more retrieving methods (like Bert, BertHybrid, FastBM25, etc.) and their hybrid versions are applied in more advanced information retrieval tasks. A new result and discussion pair dataset based on papers from PMC-OA dataset are built. Human rating are made for the result and discussion pairs. Different keyword filtering strategies like "sentence tokenizer", "PuctDigitRemoveTokenizor", simple "split" functions and different number and logic are further applied and investigated. Finally, M@L score, ROUGE score, and NDCG score are used to evaluate and compare the performance of different search engines.

Chapter 2

Text Retrieval Methods Implementation

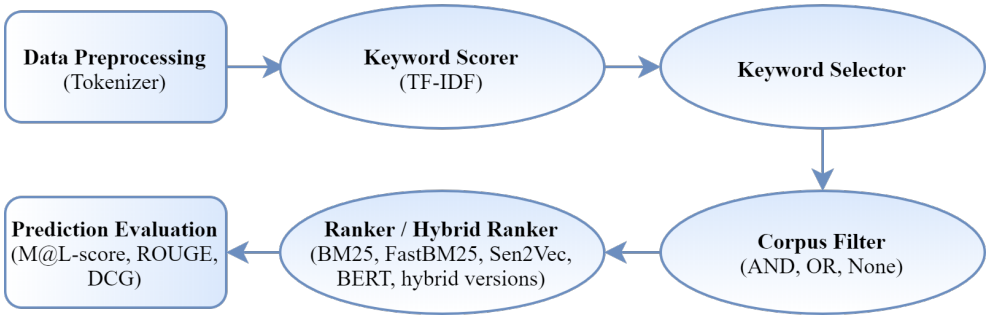


Figure 2.1: Text retrieval methods implementation procedure.

The procedure of text retrieval methods implementation (models) used in this project is shown in figure 2.1.

2.1 Data Preprocessing

Different tokenized is tried for processing the data set before fed into the model like Sentence Tokenizer(by filtering out useless characteristics(e.g. out of the set [0-9A-Za-z]), stop-words (e.g. ”a”, ”the” and so on) and word endings), PuctDigitRemoveTokenizor and simple split function. It seems Sentence Tokenizer performs best compared with other tokenizers.

2.2 Keyword Scorer, Selector and Boolean Filter

By computing the term frequency (TF) of each word in the query and the document frequency (IDF) in the corpus, the product of the term frequency

and document frequency is the TF-IDF score for each word.

By determining the number of keywords N, the words with top N highest TF-IDF scores are selected for further text retrieving usage.

When multiple keywords are selected, an "or" and "and" logic is applied for the retrieving. The "or" logic can be understood as union of all the articles where the keywords appears. The "and" logic can be understood as the intersection of articles lists where each keyword appears (e.g. keyword A appears in article i,j,k and keyword B appears in article i,m,z, the intersection is article i) which can also be understood as selecting the article where all the selected keywords appears.

2.3 Ranker / Retrieving Models

Several rankers or retrieving methods (models) have been implemented concluding BM25, FastBM25, Sent2Vec, BERT (Bidirectional Encoder Representations from Transformers) and the corresponding hybrid models which combine the existing methods with keyword selection and Boolean filtering. It is necessary to mention that none of the rankers (models) are trained on our data. In particular we only use pretrained embedding for sent2vec and BERT model.

2.3.1 Traditional keyword-based information retrieval methods

TF-IDF

TF-IDF [1] (term frequency-Inverse Document Frequency) is a common weighting technique in information retrieval and data mining. It is often used to mine keywords in the article, and the algorithm is simple and efficient, which is often used by the industry for the initial textual data cleansing. TFIDF reflects how important a word is to a document in a collection or corpus. The TFIDF value increases proportionally to the number of times a word appears in the document and is decreased by the number of documents in the corpus containing the word, which helps to adjust for the fact that some words appear more frequently in general.

Term frequency

$$\mathrm{tf}(t,d) = 0.5 + 0.5 \cdot \frac{f_{t,d}}{\max\{f_{t',d} : t' \in d\}}$$

where $f_{t,d}$ is the number of times that term t occurs in document d.

Inverse Document Frequency

$$idf(t, D) = log \frac{N}{|\{d \in D : T \in d\}|}$$

where N is total number of documents in the corpus, $|\{d \in D : T \in d\}|$ is number of documents where the term t appears.

Term frequency-Inverse Document Frequency

$$TFIDF(t, d, D) = tf(t, d) \cdot idf(t, D)$$

BM25

Okapi BM25 [2] (Best Matching) is a typical algorithm in information retrieval, which is used by search engines to compute the estimated relevance of documents to a given search query. The idea is come up by Stephen E. Robertson, Karen Spärck Jones when they built a probabilistic retrieval framework in the 1970s and 1980s. BM25 is a bag-of-words retrieval ranking function which computes the occurrence of query terms in each document and then ranks a set of documents, ignoring their proximity in the document. It is a set of ranking functions with slightly different parameters and components.

Given a query Q, containing keywords $q_1, ...q_n$, the BM25 score of of a document D is:

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

where $f(q_i, D)$ is q_i's term frequency in the document D, $|D|$ is the length of the document D in words, and $avgdl$ is the average document length in the text collection from which documents are drawn. k_l and b are free parameters, usually chosen, in absence of an advanced optimization, as $k_l \in [1.2, 2.0]$ and $b = 0.75$. $IDF(q_i)$ is the IDF (inverse document frequency) weight of the query term q_i. It is usually computed as:

$$IDF(q_i) = log \frac{N - n(q_i) + 0.5}{n(q_i) + 0.5}$$

where N is the total number of documents in the collection, and $n(q_i)$ is the number of documents containing q_i.

It can be seen from the formula that BM25 is an improved algorithm based on TFIDF, which adds a constant k_l in TF to limit the growth of TF. BM25

also introduces the concept of average document length, the influence of a single document length on correlation is related to its ratio to the average length.In the TF formula of BM25, two other parameters are introduced: *avgdl* and *b*. *b* is a constant, and what it does is it specifies how much *avgdl* affects the score.

Fast BM25

BM25F or Fast-BM25 is an improved algorithm of typical BM25. BM25 considers documents as a whole when calculating relevancy. However, with the development of search technology, documents are gradually replaced by structural data.Such as web pages are likely to be cut into the title, content, subject headings, such as field, the field's contribution to the article topics not same, so the weight needs to be adjusted, BM25 did not consider this, BM25F makes some improvements, in which the document is divided into individual, so BM25F is score in each word in each field of weighted sum.

2.3.2 Nearest neighbor search on text embedding

Sent2Vec

Sent2Vec [4] uses a simple but efficient unsupervised objective to train distributed representations of sentences, which allows to compose sentence embeddings using word vectors along with n-gram embeddings, simultaneously training composition and the embedding vectors themselves. The model can be seen as a natural extension of the word-contexts from C-BOW model to a large sentence context, with the sentence words being specifically optimized towards additive combination over the sentence, by means of the unsupervised objective function.

The sentence embedding can be presented as:

$$v_s = \frac{1}{|R(S)|} V_{R(S)} \sum_{w \in R(S)} v_w$$

where S is the sentence, $R(S)$ is the list of n-grams present in S, w is every word in the vocabulary, v_w is the word embedding.

BERT - Bidirectional Encoder Representations from Transformers

Bidirectional Encoder Representations from Transformers (BERT) [3] is a technique developed by Google for Natural Language Processing pre-training. BERT performed well in natural language understanding tasks. BERT originates from pre-training contextual representations including Semi-supervised Sequence Learning, Generative Pre-Training, ELMo, and ULMFit. Different

from previous models, BERT is a deeply bidirectional, unsupervised language representation, pre-trained using only a plain text corpus. Context-free models like word2vec or GloVe generate a single word embedding representation for each word in the vocabulary, where BERT takes into account the context for each occurrence of a given word occurring each time.

2.3.3 Hybrid search

A combination of traditional keyword-based methods and text-embedding based methods. A strategy we use in the project is keyword boolean filtering + embedding based ranking.

2.4 Prediction Evaluation

Three evaluation methods have been implemented for model's prediction evaluation.

- **"M@L score"** is a score which reveals the percentage of the actual article or "real" article that appears in the first "L" model predictions.
- **ROUGE** (Recall-Oriented Understudy for Gisting Evaluation) is an automatic relevance metrics for searched results evaluation where the similarity between query and searched result is quantitatively computed.
- **NDCG** (Normalized Discounted Cumulative Gain) is a quantitative metrics have been implemented for different retrieval methods evaluation where human rating score and model predicted ranking relationship is quantitatively evaluated.

Chapter 3

Text Retrieval Methods Application

3.1 Task A: Retrieving abstract using titles

3.1.1 Task description

Object In this part, different models are intended to be implemented and evaluated using "M@L score" on the task of retrieving an article abstract given its title in the example dataset.

Dataset A reduced dataset which contains 10000 articles concluding the information of doc_id, doi, year, title, journal name, authors, abstract and full body is used in this part.

Models (text retrieving methods) In this abstract retrieving using title task, BM25 model, sent2vec model and hybrid search models combining BM25/sent2vec model with keyword Boolean filtering, with different number of keywords and logic filters are implemented.

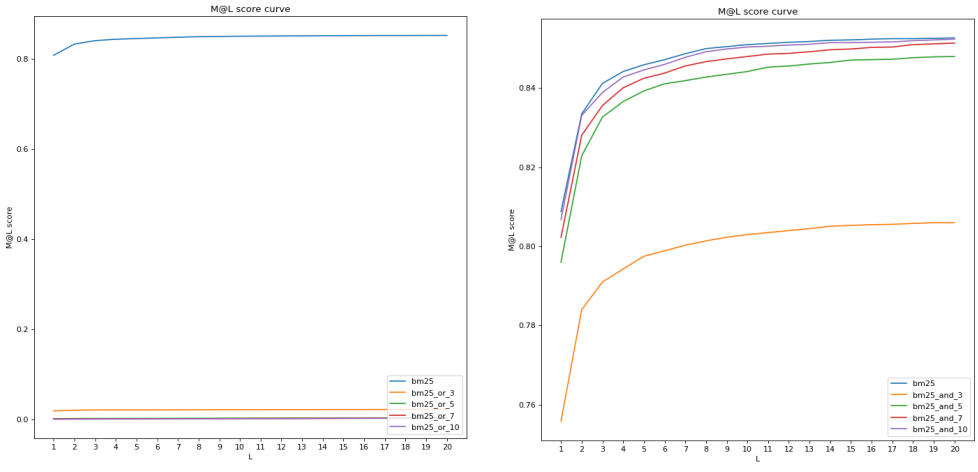
Evaluation By retrieving abstracts using titles, the performance of implemented models are evaluated using "M@L score" and "ROUGE".

3.1.2 Results and evaluation

Retrieving methods comparison

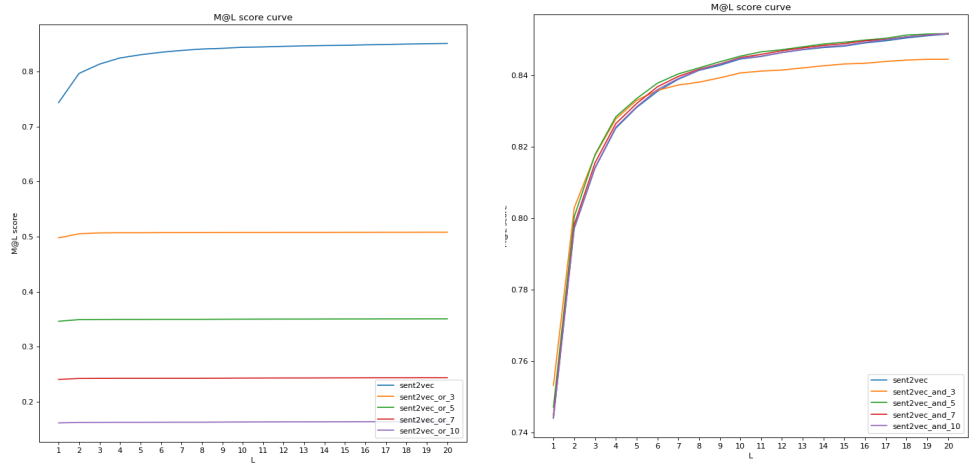
Model comparisons are made between pure model (e.g. BM25 and Sent2Vec model) and hybrid models with different filter strategy (e.g. "and" and "or" logic) and different number of keywords by analyzing M@L scores.

The comparison of BM25 model and BM25 hybrid models with **"or" filter** M@L scores can be seen from the figure 3.1a and the comparison of BM25 model and BM25 hybrid models with **"and" filter** M@L scores can be seen



(a) (b)

Figure 3.1: Comparison of hybrid search with different filters and different number of keywords for bm25 model. (a) With ”or” filter. (b) With ”and” filter.



(a) (b)

Figure 3.2: Comparison of hybrid search with different filters and different number of keywords for sent2vec model. (a) With ”or” filter. (b) With ”and” filter.

L	1	5	10	15	20
bm25	0.8088	0.8459	0.851	0.8522	0.8527
bm25_and_10	0.8068	0.8446	0.8504	0.8515	0.8524
bm25_and_5	0.796	0.8393	0.8442	0.8471	0.848
bm25_and_3	0.7558	0.7975	0.803	0.8053	0.806
sent2vec_and_3	0.7532	0.833	0.8407	0.8432	0.8445
sent2vec_and_5	0.7471	0.8335	0.8454	0.8493	0.8517
sent2vec_and_10	0.7441	0.8312	0.8448	0.8484	0.8517
sent2vec	0.744	0.831	0.8446	0.8482	0.8516
sent2vec_or_3	0.4978	0.5071	0.5077	0.5079	0.5081
sent2vec_or_5	0.346	0.3494	0.3498	0.3502	0.3504
sent2vec_or_10	0.161	0.1621	0.1627	0.1631	0.1634
bm25_or_3	0.0189	0.0211	0.0217	0.0221	0.0223
bm25_or_5	0.0017	0.0022	0.0028	0.0034	0.0036
bm25_or_10	0.0006	0.0009	0.0017	0.0023	0.0026

Table 3.1: Comparison of M@L score for all tests in tabular form

from the figure 3.1a. The comparison between "or" and "and" models is shown in figure 3.1. Similarly, the model comparison of Sent2Vec models is shown in figure 3.2. The full comparison data is shown in table 3.1.

Comparison of pure models and hybrid models with different number of keywords The BM25 model and Sent2Vec model is named as "pure model" here.

Application of keyword filtering As shown in the figure 3.1a and figure 3.1b, the M@L score of hybrid BM25 models are lower than standard BM25 model. Similar tendency is discovered in Sent2Vec models as shown in figure 3.2. However, the M@L score differences between pure model and hybrid models are much less in Sent2Vec model than BM25 model and even in Sent2Vec model, the M@L score is improved for small L which means that the filter helps this model reduce the amount of false positives.

It indicates that adding keyword filtering into the retrieval models could be more efficient improving the performance of Sent2Vec model than BM25 model.

Number of keywords As shown in figure 3.1b, a higher number of keyword tends to lead to a relatively higher M@L score in the hybrid BM25 model with "and" logic. 10 keyword selection would lead to a similar behavior compared with BM25 model. However, the behavior of keyword number

seems to be opposite in the hybrid BM25 model with ”or” logic as shown in figure 3.1a where a higher number of keywords would lead to a lower model performance. Similar tendencies are also found in Sent2Vec models as shown in figure 3.2.

It indicates that a higher number of keywords would help improve the M@L score in hybrid models with a ”and” logic filter and in contrast, fewer keywords would help improve the M@L score in hybrid models with a ”or” logic filter.

This makes sense since with an ”and” logic filter, a higher number of keywords would lead to a higher restriction which means higher correlation and is tend to find the precise content and therefore a better M@L score.

Comparison of "and" and "or" filter logic By comparing the hybrid BM25 models with ”or” and ”and” logic as shown in figure 3.1a and figure 3.1b respectively. It can be found that the ”and” logic will lead to much higher M@L score compared with ”or” logic and ”and” logic will lead to a similar M@L score tendency when increasing L value. Similar behavior is also discovered by comparing the hybrid Sent2Vec models.

It shows that in the task of retrieving abstract using title, hybrid models's ”and” logic can be a better option compared with the ”or” logic.

Comparison of BM25 model and Sent2Vec model By comparing the performance of BM25 model and Sent2Vec pure model as shown in figure 3.1 and figure 3.2, it is found that BM25 model has a higher M@L score than Sent2Vec model with the L value ranging from 1 to 20. However, it is also find that the Sent2Vec model increases much faster than BM25 do which might lead to a higher value when L is bigger than 20.

But it is known that a higher M@L value with a lower L value indicates a more precise prediction. So it can be concluded that in the task of retrieving abstract using title, the BM25 model has a better prediction than Sent2Vec model.

Metrics Comparison

M@L score **BM25** model behavior concluding the M@L score is shown in figure 3.3a with a M@1 score of 0.8088 and a M@20 score of 0.8527.

Sent2Vec model behavior concluding the M@L score is shown in figure 3.3b with a M@1 score of 0.7558 and a M@20 score of 0.806.

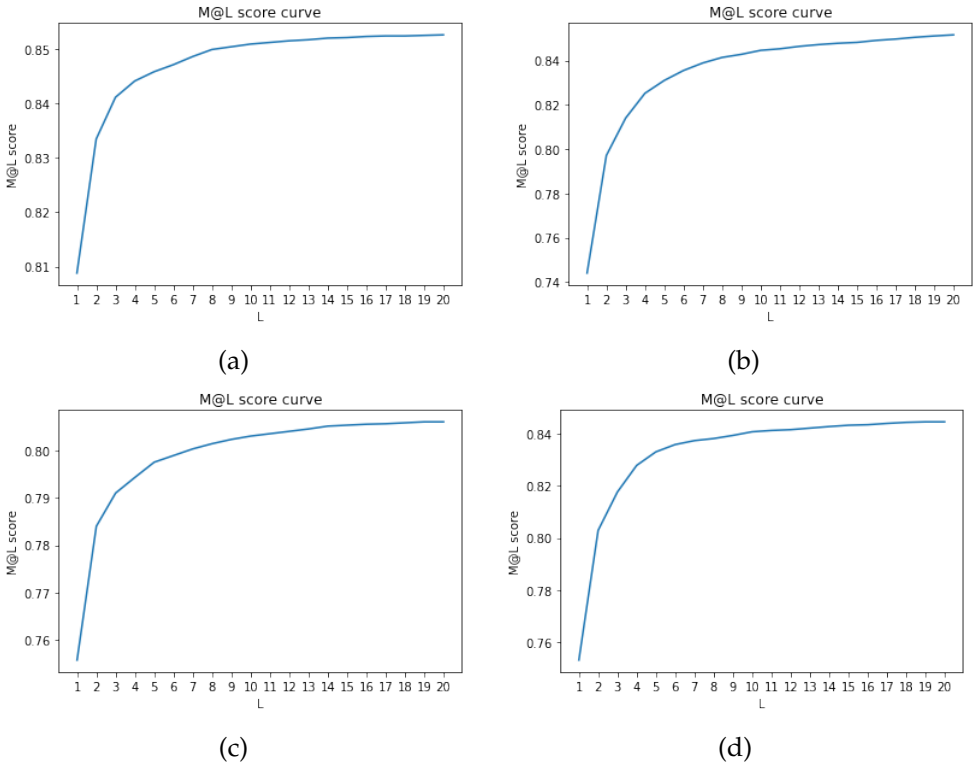


Figure 3.3: Model retrieving performance - M@L score. (a)BM25. (b)Sent2Vec. (c)BM25 Hybrid model (”and” logic and 3 keywords). (d)Sent2Vec Hybrid model (”and” logic and 3 keywords).

Table 3.2: Add caption

models	M@1	M@20
BM25	0.8088	0.8527
BM25_and_3	0.7558	0.806
Sent2Vec	0.744	0.8516
sent2vec_and_3	0.7532	0.8445

Table 3.3: M@L scores of different text retrieval models in task A.

BM25 hybrid search ("and" logic and 3 keywords) model behavior concluding the M@L score is shown in figure 3.3c with a M@1 score of 0.744 and a M@20 score of 0.8516.

Sent2Vec hybrid search ("and" logic and 3 keywords) model behavior concluding the M@L score is shown in figure 3.3d with a M@1 score of 0.7532 and a M@20 score of 0.8445.

The M@1 score and M@20 score of these models are gathered in the table 3.3.

It seems a L value of 20 can be enough and efficient for revealing the quality of retrieving method which can also be understand as model predicted first 20 articles would be efficient to reveal the model's retrieving ability.

In addition, picking a suitable L value for evaluation should be considered only when there is a requirement or limitation for the computation time and computation memory.

Further comparisons between different retrieving models would be discussed in the following sections.

bm25	rouge-1:	P: 32.84	R: 41.65	F1: 36.72
	rouge-2:	P: 9.51	R: 12.07	F1: 10.64
	rouge-l:	P: 18.33	R: 23.25	F1: 20.5
	rouge-w:	P: 18.33	R: 23.25	F1: 20.5
bm25_and_3	rouge-1:	P: 32.36	R: 38.48	F1: 35.16
	rouge-2:	P: 9.37	R: 11.15	F1: 10.18
	rouge-l:	P: 18.73	R: 22.27	F1: 20.34
	rouge-w:	P: 18.73	R: 22.27	F1: 20.34
sent2vec	rouge-1:	P: 33.53	R: 41.41	F1: 37.05
	rouge-2:	P: 9.68	R: 11.96	F1: 10.7
	rouge-l:	P: 18.71	R: 23.11	F1: 20.68
	rouge-w:	P: 18.71	R: 23.11	F1: 20.68
sent2vec_and_3	rouge-1:	P: 31.9	R: 38.7	F1: 34.98
	rouge-2:	P: 9.22	R: 11.19	F1: 10.11
	rouge-l:	P: 18.51	R: 22.46	F1: 20.3
	rouge-w:	P: 18.51	R: 22.46	F1: 20.3

Table 3.4: Rouge score calculated for different retrieving models

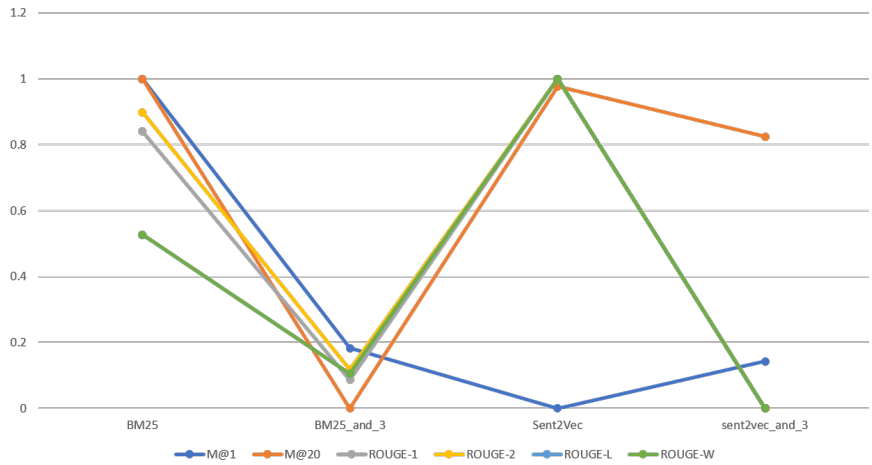


Figure 3.4: Normalized scores of different evaluation matrices using different models.

ROUGE score (Recall-Oriented Understudy for Gisting Evaluation) The calculated ROUGE score is shown in table 3.4 which can also be a reference evaluating the performance of retrieving models. As ”P” indicates precision and ”R” indicates recall and ”F1” is the multiplication-kind operation of ”P” and ”R”, therefore for simplification, ROUGE F1 scores are selected for further analysis.

A higher ROUGE ”F1” value indicates that the model generates a better prediction (predicts more accurate article) compared with the standard reference articles.

It can be found that the ROUGE-1 value is 2-3 times higher than ROUGE-2, ROUGE-L has same value as ROUGE-W and ROUGE-L value is lower than ROUGE-1 value and higher than ROUGE-2 value.

Because of the more restricted ROUGE-2 condition where 2-gram between two string list should correlate and in comparison ROUGE-1 only need 1-gram to be correlated, the ROUGE-1 is much higher than ROUGE-2 value. Since ROUGE-L ”L” indicates the longest common subsequent, it make sense it has a lower value than ROUGE-1 and higher value than ROUGE-2. ROUGE-W is the modification of ROUGE-L, which considers the embedding sequences. Considering our task, it is hard to find the absolute same sequence as title from abstract, so the embedding sequences won't effect the result and it make sense that the ROUGE-W have same value as ROUGE-L.

Evaluation metrics comparison To compare the different evaluation metrics, the scores are normalized to [0-1] for further analysis as shown in figure 3.4.

From this graph, it can be found that M@20, ROUGE-1 and ROUGE-2 have correlated tendency which can help reveal the performance of different models. However, M@1, ROUGE-L and ROUGE-W seem to have different tendency than normal tendency.

It can also be concluded from this normalized figure that the addition of keyword filtering function (hybrid model) in this task doesn't improve the text retrieving performance (prediction). BM25 model seems to have similar performance to Sent2Vec model and has a better performance than other hybrid models.

3.1.3 Conclusion

Retrieval methods In general, BM25 model seems to have similar performance to Sent2Vec model and has a better performance than other hybrid models. It might be because of the nature of the task, since title and abstract are much shorted than the body of the abstracts.

Hybrid tuning methods In this specific task, adding keyword filtering seems (applying hybrid model) to have the potential to improve the Sent2Vec model performance but not for BM25 model.

Hybrid models with **"and" logic** performs much better than models with **"or" logic**.

A **higher number of keywords** in "and" logic and a lower number of keywords in "or" logic would help improve the performance of hybrid models.

Evaluation metrics In this task, it is found that M@20, ROUGE-1 and ROUGE-2 metrics have similar tendency revealing the performance of text retrieving method which is supposed to be better than other metrics like M@1, ROUGE-L and ROUGE-W.

The normalized scores method can also be used for evaluating the performance in other tasks.

3.2 Task B: Retrieving discussions using results

In the second task, the developed hybrid search engines are applied in more advanced information retrieval tasks. A new human rated result-and-discussion-pairs dataset based on papers from PMC-OA dataset are created for 10 different topics. The developed search engines (some new ranking models are added like Bert, BertHybrid, FastBM25, etc.) with different keyword selector methods (SentenceTokenizer, PuctDigitRemoveTokenizer, split) are used to firstly pair result paragraphs to discussion paragraphs ("Pairing Task"), then retrieve discussion paragraphs with given result paragraph ("Retrieval Task"). The performance of the search engines are evaluate and compard using M@L-score, ROUGE score, and NDCG score.

3.2.1 Task Procedure

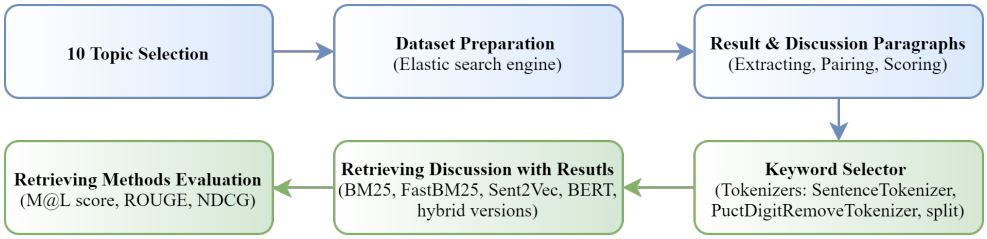


Figure 3.5: Result and discussion pairing task processing procedures.

As shown in figure 3.5, the second task is divided into following major procedures:

Topic Selection: 10 different topics are chosen for article selection in this task. The words from table S1 in the document "Inheritance patterns in citation networks reveal scientific memes – supplementary material" [5] is used as a reference for the topic selection.

The chosen topics concluding: **Reinforcement Learning, Auditory Learning, Bluetongue, Meliodosis, Malaria, West Nile virus, cerebrospinal venous, Buruli ulcer, Listeria and Chronic Cerebrospinal Venous Insufficiency**.

Dataset Preparation: For each topic, Elastic search engine is used to get 1 paper and then 9 most similar papers from PMC-OA dataset is retrieved. In total there are 100 original papers to work on.

Here is a detailed preparing procedure:

- Selecting the keywords like: ”cerebrospinal venous” and ”Buruli ulcer”.
- Using elastic search to find the keywords on title and abstract with one result returned for each.
- For each results, the tile is token and stopwords are removed, with an ”or” query on the elastic search, 9 other paper are returned.
- With the specific papers provided by the search engine, the full body, discussion paragraphs, result paragraphs are further extracted from the PMC-OA dataset.

Result and Discussion Paragraphs Processing and Scoring: With the full body extracted from the 100 papers, the result and discussion paragraphs are further extracted from the full body. The relevant results and discussions are paired and further human scored in the range of [0,1,2,3,4] according to the degree of correlation in between with a score of 0 indicating no relation between selected result paragraph and with a score of 4 indicating the discussion paragraph is exactly talking about the result paragraph.

Finally 2547 human rated scores based on scoring 30 articles out of 100 total articles are generated for result and discussion pairs. Detailed data structure and the code modules would be introduced in Appendix **??**.

Keyword Selector: To select the keywords for the retrieving task, different tokenizors are implemented concluding SentenceTokenizor, PuctDigitRemoveTokenizor and simple split function.

Pairing and Retrieving Discussion with Results: Task B is divided into to tasks, first ”Pairing Task”: the discussion paragraphs are paired with given result paragraphs then the retrieval methods (BM25, FastBM25, Sent2Vec, BERT and their hybrid versions) are evaluated according to M@L scores.

Second, ”Retrieval Task”: by giving a human rating score 0,1,2,3,4 to each retrieved search result according to its relevance to the query, different methods are evaluated by using ROUGE and NDCG quantitative matrices.

Retrieving Methods Evaluation: The search results of different models (model predictions) are evaluated together with human rated scores using M@L score, ROUGE and NDCG in this task.

3.2.2 Results and evaluation

Pairing Task

In the pairing task, different models are applied concluding BM25, FastBM25, Sent2Vec, BERT and their hybrid versions. The discussion paragraphs are paired with result paragraphs, the M@L metrics is used for retrieving method and hybrid effect analysis.

The retrieving performance of different models is shown in the figure 3.6. As can be seen from the figure, the values tend to become same (0.96) when L increases from 1 to 20.

Retrieval methods According to the M@L absolute values, the models' performance are divided into 3 categories: "high M@L score", "middle M@L score" and "low M@L score" as listed in the table 3.5.

From this categorization, it can be find that in pairing task, BM25 and FastBM25 retrieving methods are categorized as "high M@L values" and indeed have better performance compared with other methods. It can be concluded that BM25 and FastBM25 have better performance than other methods in pairing task. In addition, hybrid version of BM25 and FastBM25 models have lower M@L values which means the application of hybrid functions to BM25 and FastBM25 models would lead to reduced performance.

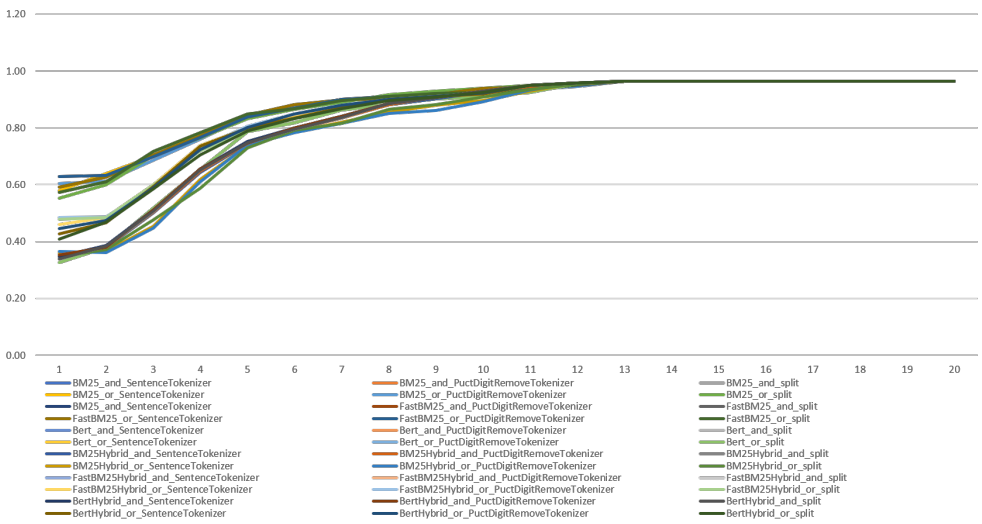
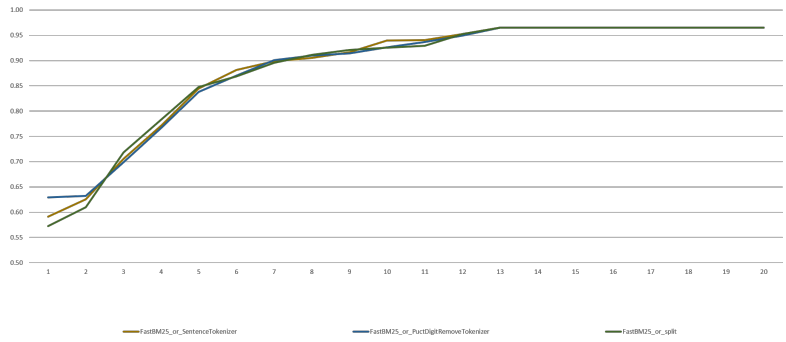


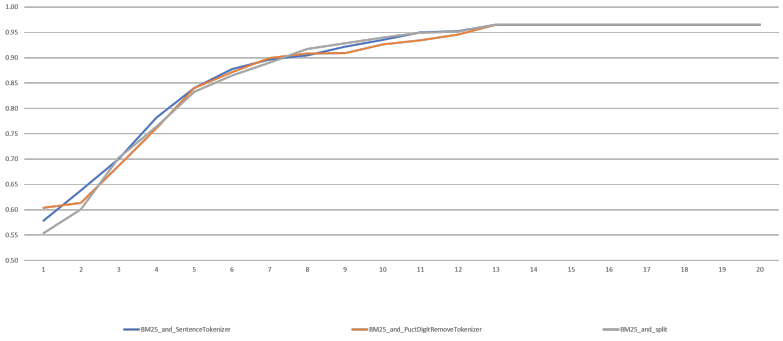
Figure 3.6: Pairing task's M@L score of all the models.

high M@L score	BM25_and_SentenceTokenizer BM25_and_PuctDigitRemoveTokenizer BM25_and_split BM25_or_SentenceTokenizer BM25_or_PuctDigitRemoveTokenizer BM25_or_split FastBM25_and_SentenceTokenizer FastBM25_and_PuctDigitRemoveTokenizer FastBM25_and_split FastBM25_or_SentenceTokenizer FastBM25_or_PuctDigitRemoveTokenizer FastBM25_or_split	low M@L score	Bert_and_SentenceTokenizer Bert_and_PuctDigitRemoveTokenizer Bert_and_split Bert_or_SentenceTokenizer Bert_or_PuctDigitRemoveTokenizer Bert_or_split BM25Hybrid_and_SentenceTokenizer BM25Hybrid_and_PuctDigitRemoveTokenizer BM25Hybrid_and_split BM25Hybrid_or_SentenceTokenizer BM25Hybrid_or_PuctDigitRemoveTokenizer BM25Hybrid_or_split
middle M@L score	BertHybrid_or_SentenceTokenizer BertHybrid_or_PuctDigitRemoveTokenizer BertHybrid_or_split FastBM25Hybrid_or_SentenceTokenizer FastBM25Hybrid_or_PuctDigitRemoveTokenizer FastBM25Hybrid_or_split		FastBM25Hybrid_and_SentenceTokenizer FastBM25Hybrid_and_PuctDigitRemoveTokenizer FastBM25Hybrid_and_split BertHybrid_and_SentenceTokenizer BertHybrid_and_PuctDigitRemoveTokenizer BertHybrid_and_split

Table 3.5: Pairing models categorization according to M@L scores.



(a)



(b)

Figure 3.7: Effect of tokenizor. (a)FastBM25 with ”or” filter with different tokenizors. (b) BM25 with ”and” filter with different tokenizors.

Keyword filtering **The Boolean logic filter function** (”and” filter and ”or” filter) has no effect on non-hybrid models. By comparing the Bert Hybrid model performance, model with ”or” filter function, its' performance is categorized as ”middle M@L score”, model with ”and” filter is categorized as ”low M@L score”. Similar tendency is also discovered in other hybrid models. It can be concluded that hybrid model using ”or” logic filter has better performance compared with hybrid model with ”and” filter.

Tokenizers (SentenceTokenizer, PuctDigitRemoveTokenizer, split) used for filtering the keywords is found has no big effect in influencing the models behavior as can be seen from figure 3.7.

Retrieving Task

In the retrieving task, different retrieval models are applied concluding concluding BM25, FastBM25, Sent2Vec, BERT and their hybrid versions. A human rating score 0,1,2,3,4 is given to each retrieved search result according to its relevance to the query, quantitative metrics like ROUGE and NDCG are used for retrieval performance evaluation.

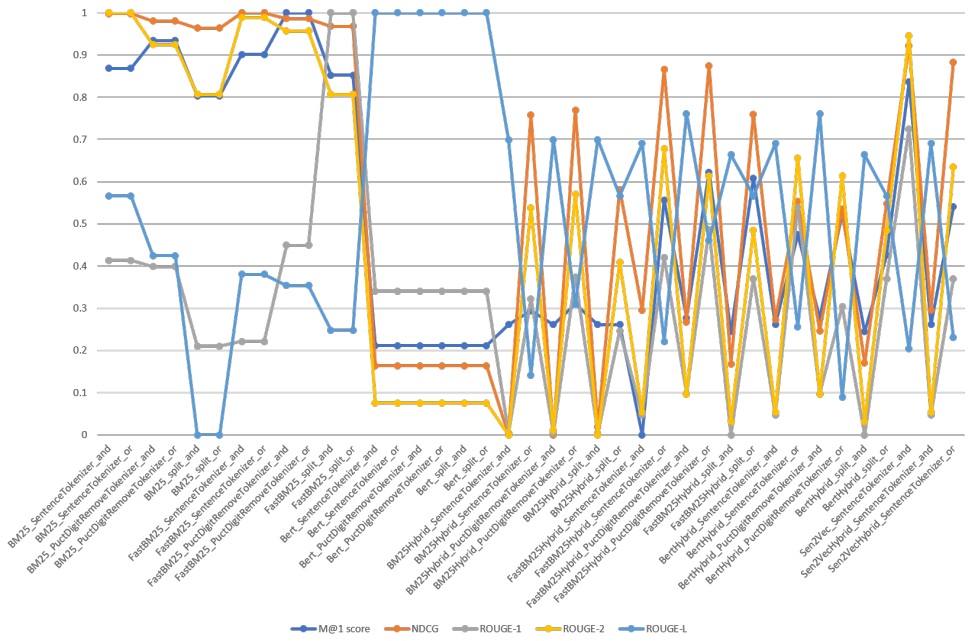
The retrieving performance of all the models using different metrics are normalized to the range of [0,1] and collected together as shown in figure 3.8a.

Evaluation metrics comparison M@L score is not used for retrieving evaluation and is used for pairing task, the M@1 score in the figure is shown as a reference which helps indicate the tendency of model performance.

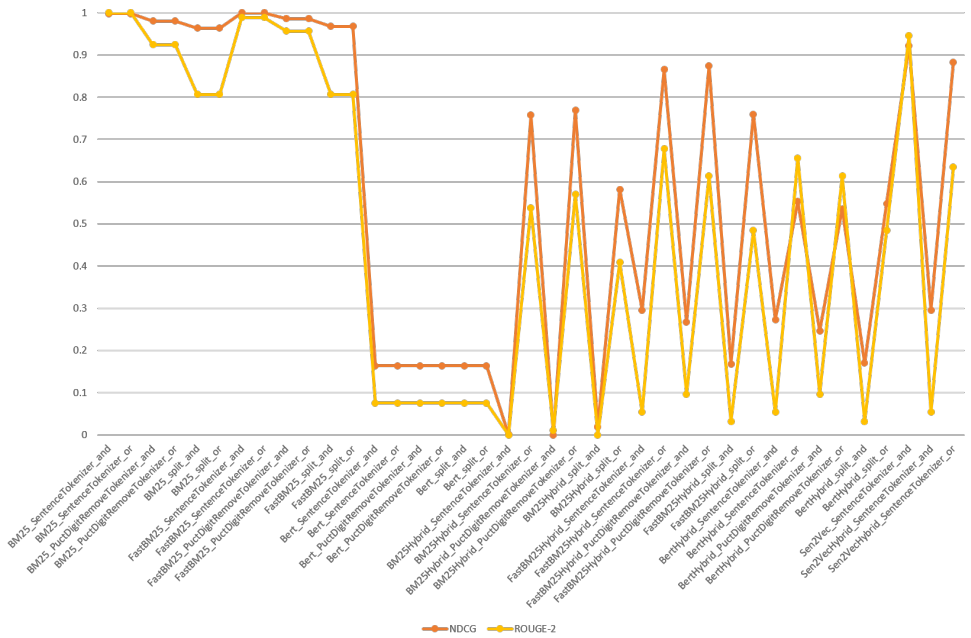
By checking the score tendency of different models in the figure, it seems that ROUGE-2 and NDCG score tend to reveal the performance of different models compared with other evaluation metrics.

As shown in figure 3.8b, the ROUGE-2 score and NDCG is extracted from all the metrics. It can be found that NDCG scores are tend to have higher normalized values compared with ROUGE-2 value, which indicates that NDCG can be a metric which can reveal the model performance better compared with other matrices.

Retrieval methods comparison As shown in left side of the figure 3.8a (or figure 3.8b), the BM25 and FastBM25 model have higher normalized values than other models which indicates BM25 and FastBM25 model have the best retrieving performance in ”Retrieval Task”. Adding hybrid functions seems



(a)



(b)

Figure 3.8: Normalized scores of different evaluation matrices using different models in retrieving task.(a) all the metrics. (b) NDCG and ROUGE-2 metrics.

to have bad influence to the task performance. In addition BERT model seem to have the worst performance compared with other methods.

Keyword selection comparison **The Boolean logic** have obvious influence on hybrid models. As shown in the right side of figure 3.8b, with an "and" logic, the model performance is much worse than the "or" logic hybrid version.

Tokenizors function can also influence the performance, although the influence is not big, a tendency that models with SentenceTokenizer and PuctDigitRemoveTokenizer seem to have a higher normalized performance scores than models with split function.

3.2.3 Conclusion

Pairing task

With the pairing task M@L results and the evaluation, it can be concluded that:

- BM25 and FastBM25 models have best retrieving performance compared with other models. Considering the computing speed, FastBM25 can be a more preferred model compared with BM25 model which requires shorter computation time and similar retrieving performance.
- Boolean logic filter has no effect on non-hybrid models and hybrid model with "or" logic has better performance compared with hybrid model with "and" filter.
- The tokenizors' effect on performance is not obvious on pairing task.

Retrieval tasks

With the retrieval task results and evaluation, it can be concluded that:

- Both ROUGE-2 and NDCG metrics can well reveal the retrieval methods performance, and NDCG metric is found can indicate model performance better than ROUGE-2 metric.
- BM25 and FastBM25 models have best retrieving performance compared with other methods. Considering the computing speed, FastBM25 can be a more preferred model compared with BM25 model.
- For hybrid models, "or" Boolean logic can have a better retrieving performance compared with "and" Boolean logic.
- Tokenizors like "SentenceTokenizer" and "PuctDigitRemoveTokenizer" have a better performance compare with "split" function.

Chapter 4

Summary

	Task A
Metrics	(M@20, ROUGE-1 and ROUGE-2) $>$ (M@1,ROUGE-L and ROUGE-W.)
Models	BM25 $\approx$ Sent2Vec $>$ hybrid
Tokenizer	SentenceTokenizer
Hybrid models	”and” logic $>$ ”or” logic. A higher number of keywords in ”and” logic A lower number of keywords in ”or” logic help improve the performance of hybrid models.

	Task B - Pairing
Metrics	M@1 $>$ M@20
Models	FastBM25 $\geq$ BM25 $>$ (Sent2Vec / BERT / hybrid)
Tokenizer	Tokenizers’ effect is not obvious
Hybrid models	”or” logic $>$ ”and” logic

	Task B - Retrieving
Metrics	NDCG $>$ ROUGE-2 metric
Models	FastBM25 $\geq$ BM25 $>$ (Sent2Vec / BERT / hybrid)
Tokenizer	(SentenceTokenizer / PuctDigitRemoveTokenizer) $>$ split
Hybrid models	”or” logic $>$ ”and” logic

Table 4.1: Project conclusions summary

The conclusions for different tasks are summarized in the table 4.1.

Metrics For pairing tasks like ”Task A” and ”Task B - Pairing task”, M@L score can be a good reference. With a small database (Task A), M@20 score (also ROUGE-1 and ROUGE-2) can reveal the performance better. With a big database (Task B), M@20 score tend to converge for different models which

all have same M@20 values and under this condition, M@1 score can be a better reference for indicating the performance of different models.

For retrieving tasks like "Task B - Retrieving task" where human rating is involved, ROUGE score and NDCG score are used for model behavior analysis, it is find that NDCG have best performance compared with ROUGE-2 score and other ROUGE scores.

Models In all the tasks, BM25 (or Fast BM25) model seems to have the best retrieving performance. This might be explained by the similarity of keywords tend to be used in the same article because of author's writing habit.

Hybrid models In task with limited dataset, keyword "and" logic seems to have better retrieving performance compared with "or" logic. In comparison, in task with big dataset base, keyword "or" logic would lead to a better retrieving performance. The keyword "and" and "or" logic filtering strategy therefore is supposed to be influenced by the size of the dataset.

Tokenizers In this project, different tokenizers are applied, it seems that Sentence Tokenizer have a better performance compared with PuctDigitRemove-Tokenizer and split function.

In conclusion, BM25 (Fast BM25) in the project have the best retrieving performance. M@20 score can be used as a metric with a small database, with a bigger database where human rating is involved, NDCG can be a better metric. Sen tense Tokenizer seem to lead to best model performance. When applying hybrid function into the model, with a big database, "or" logic can be a better filtering strategy and with a smaller database (1000 dataset) , the "and" logic seems to lead to a better retrieval performance.

Chapter 5

Individual contributions

Francesco:

- For part A, I have implemented ROUGE scoring following the python package guidelines and have introduced example code in the Jupyter notebook.
- For part B, I have analysed and extracted the paragraphs from 20 papers and paired the results and discussions within each paper.
- Then I have produced the scoring for 5 papers, resulting in around 150 scores. I have fixed the retrieval code so that it would correctly produce and display the scores and added the ROUGE metric.
- I have reviewed, streamlined and fixed the code at the various stages of production.

Simone Barbaro: My contribution to the project consists mainly of building the codebase.

- For part A, I reorganized the code given and completed the missing parts in order to solve the task. My changes also allowed a faster process of extension and addition of different models.
- For part B, I also added support for the differences with the previous task, in particular reading the new data format and running the tests on it. In particular I implemented the testing script and the computation of the NDCG metric. I also coded the Bert ranker and added different facilitations for running multiple tests and saving the results.
- Finally, I also gave my contribution to the collection of our dataset. First I choose two topics: “cerebrospinal venous” and ”Buruli ulcer”, for each topic I used the elastic search engine to find a relevant paper with the topic in the title or abstract. Then, to find related papers, I

used the elastic search engine with the title tokenized of each of the two papers as keywords. From that I generated pairs of results and discussions and gave scores as described in the main section of the report. I gave my manual scores to all the pairs for 5 of the papers I extracted.

Qiuhong Lai:

- For part A, I reviewed the code and tested the models at various stages.
- For part B, I tested the models and fixed the code, ran the code on colab for all the models, showed results in Jupyter notebook, collected them and made an Excel sheet, made a flow chart for the system so that it could be easy to understand, made some figures and tables for the final report, wrote part of the report.
- I also have a contribution to our dataset, I chose two topics: "Bluetongue", "Meliodosis". For each topic, I got 10 papers using Elastic search engine, then I extracted the result and discussion paragraphs from 20 papers and paired the results and discussions. Then I gave scores to all the pairs for 5 papers, around 250 scores.

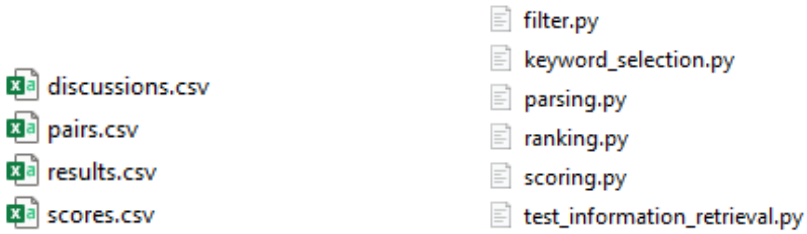
Wenjie He: The contribution from my part is processing and plot model results, build up dataset for part B, analyzing the results, writing the report, preparing presentation and in addition, reviewing and build some code.

- The codes for part A and part B are reviewed and one code used for extracting the results, discussions from the website is built by me.
- Part of the database used in part B (two topics "auditory learning" and "reinforcement learning") is created and more effort is put on scoring the result and discussion pairs. 10 different articles are scored and 1891 score pairs (2547 pairs scored in total) are generated.
- The model results are further organized, normalized, plotted and analyzed.
- Effort on the report is more paid on writing results, evaluation and conclusion of project part A and part B(chapter 3), the summary and abstract, part of chapter 2 about the implementation overall procedure, organizing the structure of the report, visualize some procedure and results (figures and tables), modification and merge of other's content and generate the final report.
- At last, the presentation file is prepared.

Appendix A

Appendix

A.1 Dataset Introduction



(a) (b)

Figure A.1: (a) Prepared dataset structure. (b) Code module structure.

There are four parts of the dataset as shown in figure A.1a :

- **discussions** contains document ids, discussion ids, and text of discussions,
- **results** contains document ids, result ids, and text of results,
- **pairs** contains document ids, discussion ids, and result ids,
- **scores** contains document ids of result paragraphs, result ids, document ids of discussion paragraphs, discussion ids, scores.

An example of the dataset created for scoring the result paragraphs and discussion paragraphs (”scores” part of the dataset) is shown in table A.1.

For each paper, an id to every result paragraph and discussion paragraph is given, and these result and discussion paragraphs are manually scored, i.e. pair result paragraphs to discussion paragraphs. For example, if discussion paragraph 0 discusses result paragraphs 0 and 1, then we have 2 pairs 0,0

doc_id_result	result_id	doc_id_discussion	discussion_id	score
1421389	0	1421389	0	4
1421389	0	1421389	3	2
653044	1	653044	2	4
2442696	1	2442696	1	2
2442696	1	193334	5	0
363993	1	363993	1	4
363993	1	363993	2	2
360628	1	363993	1	1
363993	1	995302	2	2

Table A.1: An example of the result and discussion scores.

0,1 for the paper. Then we manually give a score 0,1,2,3,4 to some random discussion-result pairs from a subset of 100 papers.

A.2 Code Module Introduction

As shown in figure A.1b 5 modules have been created in this search engine system :

- **parsing module** contains some functions to read and pre-process dataset,
- **keyword_selection module** helps to build keyword scorer which score tokens in a query to generate keywords and then to build keyword selector based on scorer,
- **filter module** helps to build a corpus filter which fetches an id list of documents based on specific filter strategy (AND, OR),
- **ranking module** builds a ranker based on models like BM25, BERT, Sen2vec model, etc. and their hybrid versions,
- **test_information_retrieval** module evaluates a prediction based on M-score, ROUGE score, DCG score, etc., it provides an easy way to test the whole system.

Table A.2 to A.6 show details in these modules.

Table A.2: parsing

name	description
parse_exemple_file	Load exemple file.
get_dataset	Create example dataset from loaded dataframe.
get_part2_datasets	Load part2 dataset.
SentenceTokenizer	Tokenizer removing all characters which are not from the set [0-9A-Za-z].
PuctDigitRemoveTokenizer	Tokenizer removing all the punctuations.
get_ngrams	Get n-grams.
add_unigram	Add unigram into inverted index according to new document.
get_doc_id_mapping	Get index-to-id mapper and id-to-index mapper.
get_inverted_index_data	Compute inverted index, return a dictionary.

Table A.3: keyword_selection

name	description
get_unique_token_occurrences	Get occurrences of each token,return a map from tokens to frequencies in the input.
add_doc_unigrams	Add unigram into inverted index.
InvertedIndex	Wrapper to help work with the inverted index.
KeywordScorer	Abstract class to score tokens in a query to generate keywords.
TfidfKeywordScorer	Scorer based on tf-idf metric.
KeywordSelector	Abstract class to select keywords based on scores generated by a scorer.
SelectKKeywordSelector	Simple strategy that select a fixed number of keywords taken by decreasing score.

Table A.4: filter

name	description
FilterStrategy	Interface of the corpus filter based on the results of the keyword selection.
AndFilterStrategy	Implementation of filter strategy that take the intersection of the documents.
OrFilterStrategy	Implementation of filter strategy that take the union of the documents.
CorpusFilter	Corpus filter that uses strategies injected in the init method to personalize how the filtering is done.
CorpusFilterBuilder	Corpus filter builder that build a filter based on filter strategy, keyword scorer, and keyword selector.

Table A.5: ranking

name	description
Ranker	Abstract class that rank documents based on a given query.
Bm25Ranker	Ranker based on the BM25 model.
FastBM25Ranker	Ranker based on the FastBM25 model.
EmbeddingRanker	Ranker based on the Embedding model.
Sent2VecRanker	Ranker based on Sen2vec model.
BertRanker	Rnker based on BERT model.
HybridRanker	Abstract Hybrid ranker.
Bm25HybridRanker	Hybrid version of the ranker based on the BM25 model.
FastBm25HybridRanker	Hybrid version of the ranker based on the FastBM25 model.
EmbeddingHybridRanker	Hybrid version of the ranker based on the Embedding model.

Table A.6: test_information_retrieval

name	description
get_tokenizer_fn	Get tokenizer.
get_corpus_filter	Get corpus filter.
get_ranker	Get ranker.
test_pairing	Evaluation of prediction based on M-score.
test_retrieval	Evaluation of prediction based on ROUGE score and DCG score.

Bibliography

[1] https://en.wikipedia.org/wiki/Tf%E2%80%93idf.

[2] https://en.wikipedia.org/wiki/Okapi_BM25.

[3] https://en.wikipedia.org/wiki/BERT_(language_model.

[4] Martin Jaggi Matteo Pagliardini, Prakhar Gupta. Unsupervised learning of sentence embeddings using compositional n-gram features. *NAACL 2018 - Conference of the North American Chapter of the Association for Computational Linguistics, pages 528-540*, 2018.

[5] Matjaz Perc Tobias Kuhn and Dirk Helbing. Inheritance patterns in citation networks reveal scientific memes. *Phys. Rev. X 4 (2014) 041036*, 2014.