Text Retrieval using Natural Language Processing

## Goal of Project

As a motivation, read the paper on ‘Embedding-based Scientific Literature Discovery in a Text Editor Application’ (a copy of the manuscript is in Polybox). The overall goal of the project is to explore natural language processing methods for retrieving scientific content based on a manuscript (or query) and potentially keywords of interest.

You have the opportunity to implement your own search engine! Throughout the project, you will gain insight into the challenges and proposed solutions to providing accurate and relevant search results. Ultimately, you will program a prototype search engine and evaluate its performance.

Along the project, you will need to find answers to questions such as:

1. How are keywords used to get relevant search results? How can the results be ranked according to the search query?
2. How can I search for related content beyond using only using keywords? How can keyword based search methods combined with nearest neighbor searches on text embeddings ?
3. What is the best strategy for suggesting keywords to make search more efficient and less work for the user?
4. What is an "accurate" and "relevant" search result? How can it be quantified and evaluated? How can the retrieval performance of a search engine be evaluated?

During the practical session, you will discuss with your instructors various strategies for programming the building blocks of your search engine and the tools to evaluate the quality of its results. You will be provided with example code. Outside the classroom, you will implement, optimize and evaluate different information retrieval methods, and develop your own solutions.

## Different information retrieval methods to be used

Throughout the project you will learn and experiment with the following text retrieval methods:

1. Traditional keyword-based information retrieval methods, such as Boolean filtering [REF] and TF-IDF (example implementations: “Okapi BM25” [1], Elastic Search engine [2,3]).
2. Nearest neighbor search on text embeddings, such as on text embedding from "Sent2vec" model which can map a sentence or paragraph into a vector [4].

For a starting point, we will provide you with a pre-trained sent2vec model and a text tokenizer (cleaner). You can first tokenize the queries/corpus and compute embeddings, then use a nearest-neighbor search method to find a list of results (e.g. 20 results) which have the closest cosine distance to the query’s embedding. (For nearest neighbor search, the Faiss library IndexFlatIP and the cosine distance metric are recommended. Please refer to the github [5]).

You can try more various sentence embedding methods like transformer networks (BERT[6]), and compare the performance across different embedding models.

1. “Hybrid” search: combining traditional keyword-based methods with text-embedding based methods.

One strategy is keyword boolean filtering + embedding based ranking. More specifically, for a given corpus, an inverted index can be computed. With this inverted index, given a set of keywords, eg {“lung”,”cancer”}, one can fetch an id list of documents in the corpus which must contain the keywords “lung” and “cancer”. This step is called “keyword boolean filtering”. Afterwards, in the fetched subset of documents, nearest neighbor search on text embedding can be applied to get most relevant results to the query.

To use the keyword boolean filtering plus embedding-based ranking in the case where we only have a query text but without pre-defined keywords, we need to consider how to extract keywords from the query text itself. There are several possible ways of doing this:

1) Randomly pickup one word or several words from the query text;

2) After reading the query text, manually select keyword(s) which you think are so important that the to-be-searched results must contain them.

3) Take advantage of TF-IDF score [7]. For example, for each word in the query, compute the term frequency within the query text and the document frequency in the corpus, and then compute their product to get a “TF-IDF” style score, pick up a certain number of keywords with highest tf-idf score as the selected keyword.

4) You can try other ways of keyword extraction, as well as optimizing the methods above. For example, what is the optimal number of keywords to be selected?

Moreover, the operation of keyword boolean filtering can also be optimized. The hybrid search described above has some limitations in some cases. For example, if given a query if you choose an improper keyword, after using this keyword to filter out all the documents that do not contain this keyword, you are very likely to miss some meaningful relevant matches.

To tackle this problem, one can try some variations. For example, one can try to select not only one keyword, but multiple keywords, such as {“lung”, “cancer”, “smoke”, “death”}. In the keyword-filtering stage, keep the document as long as it contains at least one keyword in the keyword set. Or we can use the selected keywords set {“lung”, “caner”, “smoke”, “death”} as a simplified “query” for the BM25 search, to compute the BM25 score of each document in the corpus, and keep documents whose BM25 scores are larger than a certain threshold for sent2vec nearest neighbor search.

## Project Description

The project is organized in 2 parts.

In **Part A**, each group will implement and experiment with the methods mentioned above. You will optimize and evaluate the retrieval performance of various methods. The goal is to have hands-on experience on basic concepts in information retrieval and getting proficient in implementing search engines.

In **Part B**, you will use your best “hybrid” search engine and keyword-based search engine from Part A in more advanced information retrieval tasks. You will evaluate and compare the retrieval performance of your search engines on these specific tasks, and further optimize the engines if necessary. There are 2 sub-projects in Part B. Half of the class will work on one sub-project, and the other half on the other.

Individual tasks in this project will search and retrieve search results from a dataset which you will be provided with. The dataset contains the metadata and full-body of 2.7 million research papers from biomedical and biology related fields (Pubmed Central Open-Access Subset, PMC-OA). It is a large text file, where each line of the file contains the data from one research paper. The data are structured in columns separated by "<-SEPARATOR->" string, and contains information on the DOI, title, authors, journal, publication year, abstract, and the full-body.

## **PART A (~6 Weeks)**

**Tasks**

Use the reduced dataset initially to work on these tasks. The full dataset is around 60 GB and running a search service requires around 30 GB of RAM. Initially use a reduced dataset to develop your code. Refer to the example code when implementing your solutions to the tasks below. Once your code is ready we'll discuss how to scale the dataset up.

1. Use each title from the random papers as a query to search from the all abstracts, and find 20 most relevant abstracts. Ideally the abstract which are likely to be the “real” abstract of the query title will appear in these 20 results. You will also run the task in reverse and try to retrieve the title from the abstract.
2. Use each abstract as a query and retrieve 20 most relevant full bodies from the database (and vice versa).
3. Use each abstract as a query and retrieve 20 most relevant paragraphs from the database (and vice versa).

**Results and evaluation**

Try to answer the following questions for different retrieval methods:

1. M-score: Performing a certain amount of querying operations, what is the frequency that the “real” content (e.g. the abstract if title is the query) appears in the top 1,2, ... 20 searched results?
2. M-distribution: What is the distribution of the rank of “real” content in the searched results? (histogram)
3. For each searched result, you can evaluate its relevance to the query abstract using some automatic metrics such as Recall-Oriented Understudy for Gisting Evaluation (ROUGE) [8]. Try to analyze:

* Is there any relationship between the relevance metric score (like ROUGE) and the rank order in the searched results? And how to used this metric to quantitatively evaluate the performance of the retrieval results?
* Based on the relevance metric scores, which retrieval method achieve the best performance?
* In which way we can fine-tune the hybrid search method and its variants to improve the performance and achieve a comparable result.

4. You are also welcome to try human rating score on a manageable subset. For example, given a query, for each searched result, rating it using {0,1,2,3,4} according to its relevance to the query. Then quantitatively evaluate the performance of different retrieval methods using metrics like Normalized Discounted Cumulative Gain (NDCG) [9,10,11]. Try to answer:

* Based on the human rating scores, which retrieval method achieve the best performance? Is the result consistent with what you get using automatic metrics?

Compare the results over different retrieval methods and analyze the performance.

## **PART B (~10 Weeks)**

For this part you will use 3 search engines you used/implemented in Part A. Select those which gave best retrieval performance. One search engine should be a “hybrid” engine.

## Project 1 (3 Groups)

**Tasks**

1. Choose 10 papers which were given to you in the Neural Systems course. Extract the full-body content from the pdf files.
2. For each paper, use your “hybrid” search engine to retrieve 10 similar papers which also interest you from the PMC-OA dataset. In total you will have 100 papers to work on.
3. Skim through each paper, and identify its results and discussion paragraphs. Then pair results paragraphs to discussion paragraphs. For example, if discussion paragraph d1 discusses results paragraphs r1 and r2, then we pair { r1, r2 } with discussion d1.
4. After step 3, you will have a list of paired results-discussions. Then you need to evaluate different retrieval methods on the task of searching for discussion paragraph given results paragraphs:
   1. Given results paragraphs, search for discussion paragraph from current paper;
   2. Given results paragraphs, search for discussion paragraph from all 100 papers;
5. Give a rating score {0,1,2,3,4} to each retrieved search result according to its relevance to the query. Using quantitative metric such as ROUGE, NDCG to evaluate the retrieval performance of different nlp methods.

When you use keywords with hybrid search, record the keywords you used and analyze which keyword selection favors a better search accuracy.

## Project 2 (3 Groups)

The purpose of this project is to explore if we can use different retrieval methods to extract a corpus for eventually generating a discussion section.

The scenario of discussion generation is described as follows. Suppose that an user has written a manuscript which mainly contains a few results paragraphs. Now he want to use an automatic model to generate a discussion on a certain topic, for example, “Zebra finch”, “climate change in Europe”. This model should first use this topic keyword to find related paragraphs from the current manuscript, then use the found paragraphs together with the topic keywords to find related papers/paragraphs from the database. Afterwards, based on the paragraphs searched, a summarizing model is used to generate a discussion.

**Task Description**

1. Choose 10 papers which were given to you in the Neural Systems course. Extract the full-body content from the pdf files.
2. For each paper, use your “hybrid” search engine to retrieve 10 similar papers which interest you from the PMC-OA dataset. In total you will have 100 papers to work on.
3. Skim through each paper, and identify its paragraphs from the results section. For each paragraph, your task is to use the "results" paragraphs as query and a few keywords, to search from the whole database for related papers which a discussion of such results would potentially cite. From the retrieved papers, pinpoint the paragraphs which are most relevant, which means they can be potentially used together with the query results paragraphs to produce a discussion paragraph close to the ground-truth discussion.
4. For the selection of keywords, you can try different variations:
   1. Identify the corresponding discussion paragraph of the query results paragraphs, select keywords from the discussion paragraph using TF-IDF score;
   2. Manually select keywords;

**Results and evaluation**

Evaluation of your retrieval results in the following aspects:

1. Given a query, retrieve 20 relevant paragraphs. For each paragraph, manually give a rating score {0,1,2,3,4} according to its usefulness for writing a discussion paragraph. “0” means completely irrelevant, “4” means a good match.

If a searched result with higher human rating score appears with higher rank order, the search engine should be good. To quantitatively evaluate this, one option is NDCG [9, 10,11]. Read the documentation, compute and compare the NDCG score of different information retrieval methods.

1. Using automatic text similarity metrics like ROUGE-2 [8] to compute the similarity between query and each searched result. ROUGE score is normalized between [0,1]. We can first quantize the ROUGE score into a discrete value {0,1,2,..9} to make it more compatible with NDCG. Compute and compare the NDCG score of different retrieval methods, and check if ROUGE score is fair enough to be substitute of human rating score.

Bib:

[1] <https://pypi.org/project/rank-bm25/>

[2] <https://medium.com/naukri-engineering/elasticsearch-tutorial-for-beginners-using-python-b9cb48edcedc>

[3] <https://www.elastic.co/downloads/elasticsearch>

[4] Pagliardini M, Gupta P, Jaggi M. Unsupervised learning of sentence embeddings using compositional n-gram features[J]. arXiv preprint arXiv:1703.02507, 2017.

[5] <https://github.com/facebookresearch/faiss/wiki/Getting-started>

[6] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. ARXIV:1810.04805v2

[7] [https://en.wikipedia.org/wiki/Tf%E2%80%93idf](https://en.wikipedia.org/wiki/Tf–idf)

[8] <https://en.wikipedia.org/wiki/ROUGE_(metric)>

[9] <https://en.wikipedia.org/wiki/Discounted_cumulative_gain>

[10] <https://dl.acm.org/doi/10.1145/582415.582418>

[11] <https://www.microsoft.com/en-us/research/wp-content/uploads/2005/08/icml_ranking.pdf>

[12] Nearest neighbours reveal fast and slow components of motor learning [<https://www.nature.com/articles/s41586-019-1892-x?proof=true19>]