ECOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

CS-423, Distributed Information System

Project 1:

Combining Non-Semantic Approaches for Multilingual Document Retrieval

---

TEAM_EIJ

Eeshaan Jain - 359473

Imane Benkamoun - 315906

Jeanne Allocio - 311844

Notebook link: `https://www.kaggle.com/code/eeshaanjain1/bestofall`

November 4, 2024

1 Introduction

The objective of this project was to create a document retrieval system for a multilingual corpus in seven languages ($\mathcal{L}$): English (`en`), French (`fr`), German (`de`), Spanish (`es`), Italian (`it`), Arabic (`ar`) and Korean (`ko`), and to rank documents according to their relevance to a query, with performance evaluation based on the `recall@10` criterion. We noticed that the retrieval task was not cross-lingual, and hence we divided the task into seven language-specific document retrieval sub-tasks. We surveyed the literature in semantic and non-semantic retrieval systems, and combined several non-semantic retrieval methods for the sub-tasks. Our code is available at GitHub.

2 Methods

Let $\mathcal{C}^\ell = \{D_1^\ell, D_2^\ell, \dots\}$ denote the corpus for a specific language, where D_i^ℓ is the i^{th} document in the corpus, and $\mathcal{C} = \bigcup_{\ell \in \mathcal{L}} \mathcal{C}^\ell$ denote the entire corpus. Similarly, we denote $Q^\ell = \{q_1^\ell, q_2^\ell, \dots\}$ as the set of queries for a language, and $\mathcal{Q} = \bigcup_{\ell \in \mathcal{L}} \mathcal{Q}^\ell$. After processing of a corpus, let $\mathcal{V}^\ell$ denote the vocabulary for corpus $\mathcal{C}^\ell$. We denote the number of relevant words (post processing) in document D_i^ℓ as $|D_i^\ell|$, and a term t^ℓ exists in D_i^ℓ is denoted as $t^\ell \in D_i^\ell$. We denote $\theta(t^\ell, D_i^\ell) = \sum_{t \in D_i^\ell} \mathbb{I}\{t = t^\ell\}$ as the count of t^ℓ in D_i^ℓ, and $\eta(t^\ell) = |\{d \in \mathcal{C}^\ell : t^\ell \in d\}|$ as the number of documents in which t^ℓ is present.

2.1 Efficient Data Processing and Caching

For language ℓ, we have two types of texts: (1) Document D_i^ℓ from the corpus, and (2) Query q_i^ℓ from an unknown set. We preprocess $\mathcal{C}^\ell$ beforehand, with the following steps for each document D_i^ℓ:

1. Let Ω^ℓ denote the stopwords, and S^ℓ denote the Stemmer, which converts a word to its root form. We first use the regular expression: `(?u)\b\w\w+\b` to break the document into words. We describe its working in Appendix A.2.1.

2. Our aim is to convert each word to a numerical index, so that we can look it up in a $|\mathcal{V}^\ell| \times |\mathcal{C}^\ell|$ matrix. To do so, we first skip all words in Ω^ℓ, and convert all words in D_i^ℓ to their root form using S^ℓ, and map each unique root form to an index $j \in \mathbb{N}$. Looping over all documents in $\mathcal{C}^\ell$, we can map all relevant words in the document to their indices, and construct $\mathcal{V}^\ell$: Root word $\to \mathbb{N}$.

However, when we encounter a query q_i^ℓ, we cannot proceed in the same way, as the query might contain words that are not in the cached $\mathcal{V}^\ell$. Hence, for the queries, we first find all unique words $\notin \Omega^\ell$, convert them to their root forms, and discard those words whose root words are not in $\mathcal{V}^\ell$. Then we map the tokens to the corresponding index in $\mathcal{V}^\ell$. We provide the algorithm in Appendix A.2.1.

2.2 Non-Semantic Approaches

We explore non-semantic retrieval approaches, which are based around the TF-IDF algorithm. We list the best approaches here, and describe these, and others in Appendix A.2.2. The TF-IDF algorithm (Term Frequency-Inverse Document Frequency) assigns importance to term t^ℓ in document D_i^ℓ by computing $\text{TF-IDF}(t^\ell, D_i^\ell) = \text{TF}(t^\ell, D_i^\ell) \times \text{IDF}(t^\ell)$, where $\text{TF}(t^\ell, D^\ell)$ is the term's frequency in D_i^ℓ, and $\text{IDF}(t^\ell)$ balances the term's uniqueness across documents. We implemented improvements over the TF-IDF algorithm, and modified some of them to gauge their performance on each language ℓ, and describe the 4 best-performing approaches below (where corpus is $\mathcal{C}^\ell$):

- BM25 (Original) [5]: BM25 is a probabilistic ranking function that incorporates term saturation via parameter k_1 and document length normalization via parameter b. Specifically:

$$\text{TF}(t^\ell, D_i^\ell) = \frac{\theta(t^\ell, D_i^\ell)}{\theta(t^\ell, D_i^\ell) + k_1 \cdot \left(1 - b\left(1 - \frac{|D_i^\ell|}{\mathbb{E}_{d \sim \mathcal{C}^\ell}[|d|]}\right)\right)} \quad \text{IDF}(t^\ell) = \ln\left(\frac{|\mathcal{C}^\ell| - \eta(t^\ell) + 0.5}{\eta(t^\ell) + 0.5}\right)$$

- BM25 (Modified): We modify the IDF term in BM25 (Original), keeping the TF term the same, to $\text{IDF}(t^\ell) = 1 + \ln\left(\frac{1+|\mathcal{C}^\ell|}{1+\eta(t^\ell)}\right)$

- $\text{TF}_{\ell\circ\delta\circ p}$ [6]: $\text{TF}_{\ell\circ\delta\circ p}$ aims to improve ad hoc retrieval via successive normalizations to the TF term. Specifically,

$$\text{TF}(t^\ell, D_i^\ell) = 1 + \ln\left(1 + \ln\left(\delta + \frac{\theta(t^\ell, D_i^\ell)}{1 - b\left(1 - \frac{|D_i^\ell|}{\mathbb{E}_{d\sim\mathcal{C}^\ell}[|d|]}\right)}\right)\right) \quad \text{IDF}(t^\ell) = \ln\left(\frac{1 + |\mathcal{C}^\ell|}{\eta(t^\ell)}\right)$$

- DLITE [2]: DLITE proposes an alternative to the standard IDF based on an information theoretic metric. Setting the TF term same as BM25, and $p_{t^\ell} = \frac{\eta(t^\ell)}{|\mathcal{C}^\ell|}$, $p'_{t^\ell} = 1 - p_{t^\ell}$, we write $\text{IDF}(t^\ell) = p'_{t^\ell}/2 + (1 - p_{t^\ell}(1 - \ln p_{t^\ell})) - (1 - p_{t^\ell}^2(1 - 2\ln p_{t^\ell}))/(2 + 2p_{t^\ell})$.

After computing and caching the TF-IDF matrix, given a processed query $q_j^\ell \in \mathcal{Q}^\ell$, we can compute $\textsc{Score}(q_j^\ell, D_i^\ell) = \sum_{t^\ell \in q_j^\ell} \text{TF}(t^\ell, D_i^\ell)\text{IDF}(t^\ell)$. Since we remove unseen words from q_j^ℓ, this computation is just indexing the TF-IDF matrix and calculating the sum, which is extremely efficient.

2.3 Optimizing Approaches

We discussed our optimizations and caching of text processing in Section 2.1. With regards to the approaches, the TF-IDF matrix has dimension $|\mathcal{V}^\ell| \times |\mathcal{C}^\ell|$, which is extremely large to store in memory. For example, after processing $\mathcal{C}^{\texttt{en}}$, we had $|\mathcal{V}^{\texttt{en}}| = 2741227$ and $|\mathcal{C}^{\texttt{en}}| = 207363$, and considering FP32, this would take 2.27 TB of storage! However, observing that terms are sparse, and $\theta(t^\ell, D_i^\ell) = 0$ for most $i \in [|\mathcal{C}^{\texttt{en}}|]$, we can efficiently store them using a sparse matrix format[1]. Moreover, we can parallelize the computation of the TF-IDF matrix over all tokens of a document, and also parallelize across chunks of documents. Further, given $\mathcal{Q}^\ell$, we can parallelize the computation of the scores for words in each $q_j^\ell \in \mathcal{Q}^\ell$, and further parallelize across q_j^ℓ, increasing efficiency.

2.4 Results

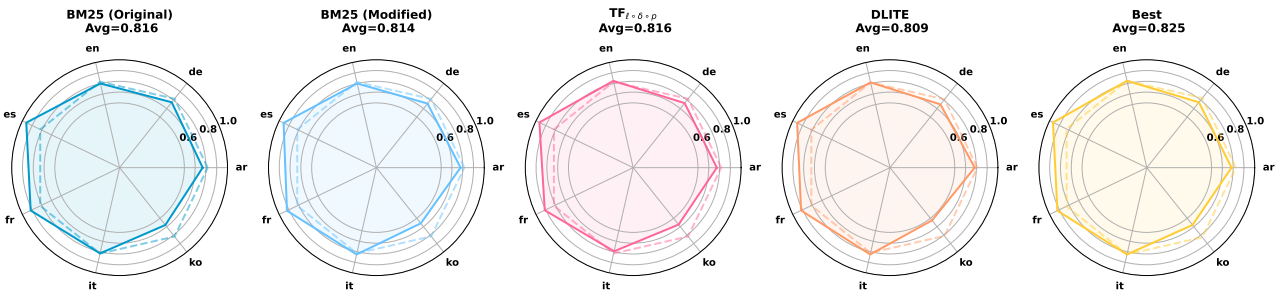

Figure 1: `Recall@10` for all languages for BM25 (Original), BM25 (Modified), $\text{TF}_{\ell\circ\delta\circ p}$, DLITE, and the Best of all on the validation set. **Avg** indicates the average `recall@10`, represented by `--`.

In Figure 1, we present the results in terms of `recall@10` for all languages and the average over all languages on the validation set for the approaches described above. We describe other approaches tried, and their results in Appendix A.2.2. We performed a grid search over a small range of hyperparameters to find the setting with the best performance on $\mathcal{Q}_{\text{dev}}$.

3 Conclusion

In our approach for the project, we split the multilinguagal document retrieval task into a language-specific document retrieval pipeline, consisting of implementations of state-of-the-art non-semantic methods and their modifications. We discuss limitations and future directions in Appendix A.4.

[1] `https://docs.scipy.org/doc/scipy/reference/generated/scipy.sparse.csc_matrix.html`

References

[1] V. Karpukhin, B. Oğuz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, and W. tau Yih. Dense Passage Retrieval for Open-Domain Question Answering, 2020.

[2] W. Ke. Alternatives to Classic BM25-IDF based on a New Information Theoretical Framework. In *2022 IEEE International Conference on Big Data (Big Data)*, pages 36–44. IEEE, 2022.

[3] O. Khattab and M. Zaharia. ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT, 2020.

[4] Y. Lv and C. Zhai. Lower-bounding term frequency normalization. In *International Conference on Information and Knowledge Management*, 2011.

[5] S. E. Robertson, S. Walker, S. Jones, M. Hancock-Beaulieu, and M. Gatford. Okapi at TREC-3. In *Text Retrieval Conference*, 1994.

[6] F. Rousseau and M. Vazirgiannis. Composition of TF normalizations: new insights on scoring functions for ad hoc IR. *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, 2013.

A Appendix

A.1 Dataset statistics

Language	$\lvert\mathcal{C}\rvert^{\ell}$	No. of tokens before PROCESSOR	$\lvert\mathcal{V}^{\ell}\rvert$ (after PROCESSOR)	time (in s) taken by PROCESSOR	time (in s) for TF-IDF matrix
English	207363	9385570	2741227	120	27.62
Deutsch	10992	3931898	1623086	19	4.21
Arabic	8829	2224752	631666	13	3.47
Korean	7893	3061396	2212304	7	3.99
French	10676	1905234	539069	15	3.11
Italian	11250	2269311	610344	20	3.47
Espagnol	11019	1821167	392307	14	2.98

Table 1: Dataset statistics indicating the number of corpus entries, vocabulary size before and after our processing pipeline We also present the time taken by PROCESSOR to process the corpus text, and the best method to calculate the TF-IDF matrix in seconds. All the times are calculated on an M2 Air, averaged over 3 runs. Due to the optimizations done, the time to calculate the top-k documents is instant, and hence we don't show them here.

A.2 Further discussion on methods implemented

A.2.1 Text Preprocessing

Regex Splitting: It first enables the unicode flag for Regex in python, and then matches boundaries (`\b`), followed by atleast 2 alphanumeric characters (`\w\w+`) and the final boundary match (`\b`). Hence, we inherently remove any punctuations present, and further can split based on the tokens matched to get words in the document.
Note about Stemming: We could not find a stemmer for the Korean language. Hence, we implemented $S^{\texttt{ko}}$ as an identity map.
Algorithm: We present the text processing algorithm below. We treat $\mathcal{C}^{\ell}$ as an array of strings, D_i^{ℓ} as a string, and W as an array of strings.

Algorithm 1 PROCESSOR($\mathcal{C}^{\ell}, \ell$)

$\Omega^{\ell} \leftarrow$ STOPWORDS(ℓ)
$S^{\ell} \leftarrow$ STEMMER(ℓ)
$\mathcal{V}^{\ell}, \mathcal{B}^{\ell} \leftarrow$ HashMap(), HashMap() *// $\mathcal{V}^{\ell}$ is the base word to base index mapping, and $\mathcal{B}^{\ell}$ is word to base index mapping*
$\mathcal{O}^{\ell} \leftarrow$ Empty Array(Array)
for $i \in [1, \ldots, |\mathcal{C}^{\ell}|]$ **do**
  $D_i^{\ell} \leftarrow \mathcal{C}^{\ell}[i]$
  $W \leftarrow$ PUNCTSPLIT(D_i^{ℓ}) *// Split document into array of words and remove punctuations*
  $T \leftarrow$ Empty Array *// T will contain the tokenized indices*
  for $j \in [1, \ldots, |W|]$ **do**
    $w \leftarrow W[j]$
    if seen w and $w \notin \Omega^{\ell}$ **then**
      $T[j] \leftarrow \mathcal{B}^{\ell}[w]$
    end if
    if not (seen w or $w \in \Omega^{\ell}$) **then**
      $b \leftarrow S^{\ell}(w)$ *// Base word*
      if seen b **then**
        $\mathcal{B}^{\ell}[w] \leftarrow \mathcal{V}^{\ell}[b]$
        $T[j] \leftarrow \mathcal{V}^{\ell}[b]$

```
        else
          V^ℓ[b] ← |V^ℓ| + 1
          B^ℓ[w] ← V^ℓ[b]
          T[j] ← V^ℓ[b]
        end if
      end if
    end for
    O^ℓ[i] ← T
end for
return O^ℓ, V^ℓ
```

A.2.2 Approaches

Drawbacks of TF-IDF Naively using the TF-IDF algorithm, with $\mathrm{TF}(t^\ell, D_i^\ell) = \theta(t^\ell, D_i^\ell)$, with optional weighting by $|D_i^\ell|$, and $\mathrm{IDF}(t^\ell) = \log\left(\frac{|\mathcal{C}|^\ell}{\eta(t^\ell)}\right)$ is suboptimal since with a large corpus, the term frequencies can have a high variance, and rare terms can get assigned high IDF values, but not all rare terms are useful.

Alongside BM25 (Original), BM25 (Modified), $\mathrm{TF}_{\ell\circ\delta\circ p}$, and DLITE, we also tried the following:

- TF-IDF: Here, we naively use

$$\mathrm{TF}(t^\ell, D_i^\ell) = \theta(t^\ell, D_i^\ell) \quad \mathrm{IDF}(t^\ell) = 1 + \log\left(\frac{1+|\mathcal{C}|^\ell}{1+\eta(t^\ell)}\right)$$

- BM25+ (Original)[4]: BM25+ aims to improve the normalized TF term's core issue – over-penalization of long documents. It uses a parameter δ, as a pseudo TF value to control the lower bound of TF. Specifically,

$$\mathrm{TF}(t^\ell, D_i^\ell) = \delta + \frac{(k_1+1)\theta(t^\ell, D_i^\ell)}{\theta(t^\ell, D_i^\ell) + k_1 \cdot \left(1 - b\left(1 - \frac{|D_i^\ell|}{\mathbb{E}_{d\sim\mathcal{C}^\ell}[|d|]}\right)\right)} \quad \mathrm{IDF}(t^\ell) = \ln\left(\frac{|\mathcal{C}^\ell|+1}{\eta(t^\ell)}\right)$$

- BM25+ (Modified): We modify the BM25+ algorithm to have the same IDF term as BM25 (Original), *i.e.,* $\mathrm{IDF}(t^\ell) = \ln\left(\frac{|\mathcal{C}^\ell| - \eta(t^\ell) + 0.5}{\eta(t^\ell)+0.5}\right)$.

- $\sqrt[3]{\mathrm{DLITE}}$: It is shown in [2] that a cuberoot of the IDF term proposed is also a metric, and hence can be an alternative to the IDF term. Hence, it proposes, with $p_{t^\ell} = \frac{\eta(t^\ell)}{|\mathcal{C}^\ell|}$, $p'_{t^\ell} = 1 - p_{t^\ell}$,

$$\mathrm{IDF}(t^\ell) = \sqrt[3]{\frac{p'_{t^\ell}}{2} + (1 - p_{t^\ell}(1 - \ln p_{t^\ell})) - \frac{(1 - p_{t^\ell}^2(1 - 2\ln p_{t^\ell}))}{(2 + 2p_{t^\ell})}}$$

A.2.3 Results

Name	Arabic	Deutsch	English	Espagnol	French	Italian	Korean	Mean
TF-IDF	0.315	0.325	0.285	0.680	0.650	0.480	0.415	0.450
BM25 (Original)	0.770	**0.775**	0.800	0.960	0.915	0.815	**0.680**	0.816
BM25 (Modified)	0.780	0.760	0.800	0.955	**0.920**	**0.825**	0.660	0.814
$\mathrm{TF}_{\ell\circ\delta\circ p}$	0.775	0.765	**0.825**	**0.965**	0.910	0.795	0.675	0.816
DLITE	**0.785**	0.750	0.810	0.955	0.915	**0.825**	0.625	0.809
$\sqrt[3]{\mathrm{DLITE}}$	0.755	0.720	0.710	0.920	0.900	0.805	0.570	0.769
BM25+ (Original)	0.750	0.740	0.785	0.930	0.895	0.765	0.665	0.790
BM25+ (Modified)	0.750	0.740	0.785	0.930	0.895	0.765	0.660	0.789
Best	**0.785**	**0.775**	**0.825**	**0.965**	**0.920**	**0.825**	**0.680**	**0.825**

Table 2: Best results after hyperparameter search for all methods implemented on $\mathcal{Q}_{\mathrm{dev}}$

A.3 Details about submission

We submit the notebook at `https://www.kaggle.com/code/eeshaanjain1/bestofall` (version 5). We store the tf-idf scores for the best methods as a dataset (`https://www.kaggle.com/datasets/eeshaanjain1/tfidf-dump/data`), which are then loaded to the notebook and run.

A.4 Limitations

We currently focused on enhancing and working with non-semantic approaches for document retrieval. Methods surrounding BM25 have also been popular in industry, with recently the CEO of Perplexity favoring BM25 for Retrieval Augmented Generation[2]. But, these methods don't capture semantic information, for *e.g.,* we would like *good* and *great* to be clubbed together, but in our case, they get assigned different TF-IDF scores. Moreover, recent methods like DPR [1] use Language Models (LMs) to embed documents and queries into single embeddings, and performing a semantic search using similarity metrics such as cosine similarity have become popular. Contrasting to dense retrieval, sparse retrieval breaks the document and queries into tokens (words), and performs a late-interaction-based MaxSim search, such as in ColBERT [3], which has been considered state-of-the-art in sparse retrieval. Future directions could be towards hybrid retrieval, where we select the top-nk documents using BM25-like algorithms, and then select the top-k from these using sparse/dense retrieval methods.

[2] `https://www.youtube.com/watch?v=I3T805paHvE`