DIS 17.1 Search Engine Technology

Constantin Krah, Andreas Kruff, Joshua Thos, and Anh Huy Matthias Tran

Technische Hochschule Köln, Ubierring 48, 50678 Cologne, Germany

Abstract. Improving the search results in a COVID-19 corpus with the help of boosting, synonym expansion and re-ranking via clusters.

Keywords: Elasticsearch · Information Retrieval · Re-ranking

1 Introduction & Problem Statement

In this semester assignment, we were tasked to specifically build our own search engine with the help of either Solr or Elasticsearch with either the predefined function or our own approaches in order to retrieve accurate search results on a predefined, fixed data set called CORD-19 based on various articles about the coronavirus. On the basis of 50 provided topics/questions, the main goal was to improve and to refine our search results retrieved by the search engine so that important measures such as the precision or the recall continually improve. To achieve this, we have learned multiple methods in modules such as Information Retrieval and Search Engine Technology. Part of the project's agenda lies in determining which of Elasticsearch's default settings to keep, which of them to alter or altogether remove and which methods to add to our search engine in order to complement it. For every run, the 1000 best results for each topic/question that we used for the querying were retrieved. The resulting file then was compared to the provided qrels file and consequently evaluated with the help of TREC_EVAL, which calculates important measures to evaluate the performance of our search engine. Our focus will lie on optimizing our search engine in Elasticsearch for the precision@5 and reciprocal rank metrics as we think these are the most important measures for a search engine in a real environment.

2 Approach & Implementation

2.1 Index management

We decided to split the data within one single shard, since the benefit of parallel search requests by splitting it into multiple shards is not relevant for our environment and the relevant metrics such as term frequency are calculated on shard-per-shard basis. By using only one shard, our metrics represent the whole index corpus more accurately(Connelly, 2019).
The mapping of our index is specifically tailored towards our query and as such, focuses mostly on the fields "title" and "abstract". As the title of a document

is generally the most important and most comprehensive description of a document and its contents, we decided to map the title into multiple multi-fields in order to be able to offer this particular field for more in-depth queries:

- **Title:** Raw, unconverted, tokenized title text
- **Title.analysis:** Analyzed, stemmed and tokenized title text
- **Title.keyword** The whole title text in one unconverted token

By mapping these different title fields into the index, we allow more in-depth queries when it comes to finding matches from the title alone.
The abstract field is indexed as a regular text-field without multi-fields in order to avoid excessively bloating the size of our index.

2.2 Analyzers

In order to prepare the content of our Index properly for queries, it is necessary to analyze and convert the content from the metadata into proper tokens that can be used for an accurate search.

- **Covid Analyzer:** Analyzed, stemmed and tokenized title text and added synonyms
- **Query Analyzer:** Analyzed, stemmed and tokenized title text

The following subsections are describing the filters that we applied for the "Covid Analyzer". Besides the synonyms and the acronyms filters the "Query Analyzer" contains the same filters as the "Covid Analyzer".

Synonyms For the synonyms we thought about multiple approaches to reach a query expansion that improves our TREC_EVAL results: In order to obtain relevant words, we extended the queries with the given topics, narratives and the questions. The first step for this task was to remove the stopwords and punctuation via Python. After that we tokenized the words and used the NLTK Wordnet thesaurus(Princeton University Department of Psychology, n.d.) to add more synonyms to our tokens. While it works well for non-specific terms, it offered no additional synonyms for the COVID-19 context. For the medical terms we found a specifically thesaurus for COVID-19 terms (CAS, n.d.) to extend our synonyms file. In the resulting file We discovered synonyms, which did not fit within our context. So we had to clean the file manually. As we did that, we also merged some of the synonyms, because words like COVID-19 and Coronavirus were not linked with each other. The resulting file was integrated into our index.

Language, Stemmers, Keywords & Tokenization Due to the fact that the underlying queries from the topics file are all in English and that consciously disregarding documents of different languages did not worsen our results significantly, we aimed to optimize our Covid Analyzer just for English under the assumption that only English documents are relevant. In addition to the synonyms being applied in index time within the Covid Analyzer, an acronyms filter is used to complement our bag of words concerning various abbreviations. Additionally, stop words are also filtered through the analyzer via a predefined

function within Elasticsearch, a lower-case filter is employed to avoid making capitalization matter for query matches and an ASCII-folding token filter is used to transform special characters such as "à" to "a" in order to make querying the index for matches more relevant. In order to avoid tokenizing important keywords such as "United States" for the sake of preserving their true meaning, we implemented a keyword_marker filter based on the content of the provided topics-file. For stemming we used and compared two built-in stemmers within Elasticsearch: The Standard Porter Stemmer (Porter Stemming Algorithm, 2006) Algorithm and the Snowball Stemmer Algorithm (Snowball, n.d.) which cuts words into tokens more aggressively than the former.

2.3 Querying

The title multi-fields and the abstract fields stand central for our querying purposes. In order to make the query builder more specific, multiple specifications, weights and expansions have been made beyond just inserting the query into Elasticsearch:
For the title multi-field we used the multi-match query with the best-fields specification in order to not score the same match multiple times for the final score.For a different run in the re-ranking period, we also executed an exact-match search on the cluster field with the weight of five for the most reoccurring clusters of the previous ranking, which were created from the SPECTER embeddings of the COVID-19 corpus(more on that in re-ranking). Additionally, we assumed that any document before the end of 2019 are to be scored as less relevant, as the COVID-19 virus was not known before that time. This condition is only optional and will only have an effect on the score, if a match either on title or abstract has been found. To effectively apply this boosting on the whole corpus it was necessary to create consistency in the "publish_time" column. For that every column that just contained the year was set to the first of January while the 18 documents without any entries in this column had to be added manually by searching external sources.

2.4 Ranking Models

Concerning the Ranking Models, multiple similarity modules(Elasticsearch B.V., n.d.-c) and scoring methods have been implemented:

- **TF-IDF:** As a baseline scoring method.
- **BM25:** As the standardized scoring method for Elasticsearch and many other lucene-powered search engines.
- **DFR:** Divergence from randomness.
- **LMJelinekMercer:** An algorithm that captures important patterns in a text.

These ranking models were quickly tested and evaluated for their search performance in the title and abstract fields respectively, in order to use the most optimal one for the COVID-19 corpus. Ultimately, we used different similarity modules for the fields; LMJ for the title multi-fields and the DFR algorithm for the longer texts found in the abstracts as this combination performed the best.

Divergence from randomness (DFR) We implemented the DFR framework introduced by Gianni Amati and Cornelis Joost Van Rijsbergen.(Amati Van Rijsbergen, 2002, p. 371) They introduced probabilistic models of information retrieval based on measuring the DFR. The model was tested on both the title and the abstract fields, but it turned out to be most successful in longer sections of pure texts that can be found in the abstracts. For our approach the basic model "geometric approximation of Bose-Einstein"(University of Glasgow, School of Computing Science, n.d.) delivered the best results. It turns out that for the first normalization, also called the "After Effect", the ratio of 2 for Bernoulli processes works better than Laplace's law of succession. For the second normalization the term frequency normalization provided by the Zipfian Relation (TREC_EVAL: Calculating Scores for Evaluation of Information Retrieval, 2016) turns out to be the best.

LMJelinekMercer (LMJ) The LMJ similarity module within Elasticsearch is a language model based on the J. Mercer smoothing method (Apache Software Foundation, 2015). This model only has a single parameter called λ. Per default definition the value of λ is optimal at 0.1 for short queries and 0.7 for long queries. Different parameters were tested out in preliminary runs, but the optimal ones provided the best results. The language model calculates values for the queries that are between zero and one. If the number is near 0 the conjunction between text and query is pretty good, but the closer the number gets to one the more unimportant the document is. We used this algorithm on the title, because in our tests it provides the best results when used on short texts found regularly in the title fields.

2.5 Re-ranking

In order to implement a proper re-ranking procedure we aimed to use the contents of an embeddings file to re-rank our search results. For this, we referenced the approaches mentioned in scientific papers that compared various methods to assign documents according to their similarities based on citation graphs .(Wang Lo, 2020, p. 13) Out of the mentioned approaches, SPECTER seemed to be the most powerful for our specific task for creating embeddings and finding certain similarities based on citations between the documents in the corpus. With the SPECTER approach, for every cord_uid, a vector of 768 dimensions has been assigned, describing the similarity based on citations and content between different documents. In order to use the embeddings for our use case, we found an approach that clusters the documents according to their similarity vectors with the k-means elbow method (Eychaner, 2020). For our data set we received six different clusters with this approach. For this, we created a new metadata.csv that contains the cord_uid and their associated clusters. Afterwards, we added the identification of the cluster as a column in the existing metadata file and added it to our index and included it within our query as mentioned as in the previous querying section. To implement the re-ranking, we wrote a function that counts how often a cluster occurs in the first 50 retrieved results for every query. Consequently, we implemented a step-wise boosting, which boosts documents of

a certain cluster higher if a specific cluster is represented significantly more often than others(see re-ranking 1 2 in table 1). Based on the clusters that occurred more than 40 times in a query result we applied an additional re-ranking for the next run when a document belongs to one of these clusters for the specific query in order to increase the number of retrieved related documents even further(see re-ranking 3 in table 1). The re-ranking 1 and 2 were, in our opinion, the most impactful approaches in improving our search engine in the aspects of precision@5 and the reciprocal rank, as it managed to achieve incremental increases in both aspects while the percentages were already high, under the assumption that improvement becomes harder the better the initial results already are. It is also important to note that, since our re-ranking procedure depends on the best 50 search results of the previous ranking, its success is highly correlated with all the other approaches that have been done before the re-ranking. Nearer details about each methods and its specific effect on the search results will be explained in the next section.

3 Experiments & Evaluation

3.1 Introduction and Evaluation Plan

In order to compare and document the success and failures of our previously explained approaches, we decided to measure the success of our elastic back end and any recent additional implementations incrementally, keeping what works and tossing out what leads to worse result as individually testing singular parameters for their impact on the search results seems inefficient, since all the components within the search engine invariably rely and influence each other. For this comparison, we focus on comparing our various prototypes with four metrics:
Precision@5, **Precision@10**, **Reciprocal Rank**, **Recall**

Between these attributes, we mainly focused on optimizing our **Precision@5** and **Precision@10** as in a real search environment, they are of utmost importance to achieve accurate and precise results for any user on their search query. While recall is important as well, achieving a good recall usually comes at a cost with precision. As long as the search engine manages to retrieve relevant search results for all 50 queries, the recall should serve fine enough, which is why the recall is just a secondary metric for our evaluation plan in comparison to the precision and the reciprocal rank. The different prototypes were tested in an incremental order from left to right and resulted in these following runs (see Table 1):

Table 1: Comparing the different runs

	MVP	**Index**	**Boosting**	**Improv. Synonyms**	**Re-ranking 1**	**Re-ranking 2**	**Re-ranking 3**
num_q	50	50	50	50	50	50	50
num_ret	49416	49330	49329	49347	49347	49450	49348
num_rel	26664	26664	26664	26664	26664	26664	26664
num_rel_ret	**9679**	**9401**	**9788**	**9930**	**9929**	**9666**	**9935**
map	0.1847	0.1892	0.2035	0.2059	0.2057	0.1927	0.2057
Rprec	0.2826	0.2807	0.2978	0.3027	0.2994	0.2898	0.2995
recip_rank	**0.7690**	**0.8627**	**0.8627**	**0.8686**	**0.8935**	**0.8474**	**0.8935**
P_5	**0.6480**	**0.6820**	**0.7320**	**0.7600**	**0.7880**	**0.7040**	**0.7880**
P_10	0.6380	0.6720	0.6820	0.6980	0.7000	0.6660	0.7000
P_30	0.5633	0.6233	0.6300	0.6327	0.6373	0.6007	0.6373
P_1000	0.1936	0.1880	0.1958	0.1986	0.1986	0.1933	0.1987
Recall	0.3630	0.3526	0.3671	0.3724	0.3724	0.3625	0.3726

3.2 MVP

The MVP portrays the very basis of our search engine. The raw metadata have been inserted via the Kibana GUI (Elasticsearch B.V., n.d.-a) without any kinds of configurations on either the index or the mapping. The query builder was kept simple and simply inputs the given query from the topics-file into our Elasticsearch cluster to search both the title and abstract fields. In order to avoid duplicates in our results we just accepted the best scoring cord_uid and ignored following identical cord_uid´s.

3.3 Configured Index + Synonyms + Query + Similarity Module

This first milestone contains multiple configurations and implementations: The index has been fully configured and expanded with a synonyms list, acronyms and its tokens get produced by our custom analyzer "Covid Analyzer". The index mapping has been diversified to multiple multi-field mappings in the title field(including a raw field, a keyword field and an analyzed field) in order to allow for more precise querying.

The query itself uses the provided query from the topics-file and applies it to the various multi-fields in the title-field and the unchanged abstract-field. Matches from analyzed title-fields are valued regularly, while matches from the keyword field from title are valued five times higher than normal. Additionally, documents with a publish_date before the year 2020 were scored as less relevant than any other documents with a modifier of 0.8, as they did not seem topically relevant to the given topics-files, while more relevant documents from 2020 are scored 1.2 times higher. In addition, the similarity modules BM25, DFR or LMJ have been compared between each other for their performance. The final run of this milestone uses a combination of DFR for abstract fields and LMJ for title multi-fields.

As it is evident in this exported run, this step actually did not overly improve our search engine while making our recall even worse. This can be explained

by the untuned nature of the various settings on the index and the fields that made the requirements for matches much more stricter than the MVP. The more strict and exact a query gets evaluated alongside the search pipeline, the higher the precision and the lower the achieved recall. This explains why this run has achieved a higher P@5 of **0.6820** and a higher reciprocal rank of **0.8627** than the MVP, but overall a worse recall with **278** less relevant documents retrieved.

3.4 Boosting

After adjusting the boosting in our query to 1.5 for matches in the title.analysis field, one in the title-field, five in title.keyword and eight in the abstract, the search results have been significantly improved in both precision and recall.
The testing here was done incrementally until the optimal distribution of weights to each field has been found. It is clear that boosting can achieve both good and bad results. Maladjusted boosting can bring down the quality of the search results immensely as it can be seen in the previous run. Fine-tuning or nearly optimizing the distribution of weights for boosting can elevate both the recall and precision of the search results even further. In our case, it retrieved **387** more relevant documents with a higher P@5 of **0.6720**.

3.5 Improved Synonyms

Afterwards, the synonyms list has been summarized in a more concise manner in order to assure a more standardized and correct usage of the various synonyms while creating the index.
A more comprehensive and concise list of important synonyms enable the query to capture matches with documents, that were previously marked as false negatives, with more accuracy. This increases the precision slightly, and the recall heavily and was actually quite effective in improving our search performance with **142** more relevant documents retrieved and a higher P@5 of **0.7600**.

3.6 Re-ranking 1 & 2: Clusters

With the help of re-ranking by assigning each document to a specific cluster according to their embedding file, the following run has been achieved. As we achieved these runs, we retroactively tried out BM25 (re-ranking 2) as a chosen similarity module for both title and abstract fields and compared it with our previous set up of LMJ + DFR(re-ranking 1) in order to ensure that after all the previous procedures, we still used the best initial ranking model for our search engine.
While the recall stays mainly the same, the precision have been improved greatly in re-ranking 1 with **9929** retrieved documents and a P@5 of **0.7880**. This could be explained by the process of the implemented re-ranking, where the top result's clusters where evaluated and all documents in the same clusters were boosted in score. In effect, previously low-ranking documents for a specific query started to get rated higher in the new ranking merely because they were in the same cluster as a top-ranked documents. This resulted in a trade-off, where the precision in the higher rankings managed to improve, while the precision at lower rankings(@500) got slightly worse. The downside of this trade-off is of no consequence for most

users, as most users do not even look past the first page of a search result, so we can categorize this implementation of a re-ranking. Concerning the initial ranking module, it turns out that BM25 performs considerably worse than our previous set up with LMJ for the title fields and DFR for the abstract field. This milestone showed us that that using a document's inter-document similarity to the rest of the corpus is a very useful approach to extend your search results with relevant documents which do not directly fit for the initial assumed information need of the user.

3.7 Re-ranking 3: Query Specific Cluster

In a last bid to improve the search results, we used query-specific clusters which were defined for the re-ranking in order to boost documents in a top-occurring cluster even further and potentially retrieve more relevant documents that were previously outside the top 1000 search of each query. The intent behind this implementation lied in improving overall precision and recall by retrieving more relevant documents that were previously marked as false negatives. While this re-ranking procedure did not improve our search results massively, it did increase our precision at lower ranks at P@1000. This can be explained by the six additional relevant documents that have been retrieved through this additional approach.

4 Context within our Study

In the module "DIS12 Information Retrieval", we were taught the theoretical knowledge of how to retrieve documents of an unstructured nature in a large collection based on a specific information need. We learned that the starting point for an information need is always a query and a corpus of documents from which documents relevant to the user are to be identified. Solving this problem in an automated manner is a difficult task as programs can not easily infer a user's contextual information need just from a simple query. Coming from this setting, we learned many applicable concepts to tackle issues such as relevance problem or the "Feast-Or-Famine" problem with methods such as the TF-IDF model to calculate relevance ranking, other language models to match specific query questions and the basics behind tokenization, stop words or lemmatization. Additionally, we learned basic knowledge in understanding various metrics such as precision@k or recall, that ultimately helped us conduct and properly evaluate our experiments with our search results. This knowledge has then been extended with new approaches we learned in the current module "DIS17 Suchmaschinentechnologie", where we utilized our newly acquired knowledge as described in this paper. Another module that helped elevate this project was "DIS06 Programmierung Softwareentwicklung" in conjunction with "DIS08 Datenmodellierung", where we were taught the basics of the python programming language and the common procedure behind effectively manipulating and cleaning data in an automated manner. These modules were essential for us in order to build a proper python pipeline that cleans the COVID-19 corpus, creates the index, builds the

query, searches the index and outputs the results in a specific text-format while collaboratively working within one shared GitHub setting.

5 Conclusion & Future Research

Although the implemented search engine was just optimized for the COVID-19 topic it allowed us to apply learned approaches and methods in an actual programming environment and see how these concepts actually play out in either improving or worsening our search results, leading to a deeper understanding of the Information Retrieval concepts. Focusing on just one topic allowed us to achieve quick, presentable results even without using own expensive machine learning algorithms or balancing a broad vocabulary of wildly different topics and contexts. While many concepts like fuzzy matches or using the title as a keyword had been worthless for our project, they would be essential in a real use case. In future research we could consider the usability of our search engine in a dynamic environment without predefined queries. Additionally, further concepts could be introduced in order to improve our search engine; many machine learning concepts that we learned about during this semester, could be trained on the massive COVID-19 dataset in order to optimize the search results in order to gain context from the synonyms in both query and index time or generating embeddings in order to find more similarities within the document corpus. There is also the possibility for introducing the concept of knowledge graphs into the system, which could properly present the context between all documents within the data set and lead to a deeper understanding between the relations in the document corpus. In order to employ a more robust and usable testing procedure, it might be optimal to create an automated testing pipeline to handle creating the index, testing different variables and document the results. Implementing a system for this purpose might save more time in the long run and enables even more optimization. In the end, we are ultimately rather satisfied with our results, especially since we were able to implement external functions not native to the Elasticsearch environment to further improve our search engine results. This was especially gratifying as success nor improvement was not guaranteed, but the effort paid out in the end for both our results and our personal learning experience.

References

1. Amati, G., Van Rijsbergen, C. J. (2002). Probabilistic models of information retrieval based on measuring the divergence from randomness. ACM Transactions on Information Systems, 20(4), 357–389. https://doi.org/10.1145/582415.582416
2. Apache Software Foundation. (2015, September 17). LMJelinekMercerSimilarity (Lucene 5.3.1 API). Lucene.Apache.Org. https://lucene.apache.org/core/5_3_1/core/org/apache/lucene/search/similarities/LMJelinekMercerSimilarity.html
3. CAS. (n.d.). Your CAS COVID-19 Thesaurus Request. Retrieved 12 December 2020, from https://www.cas.org/covid-19-thesaurus-completed
4. Cohan, A., Feldman, S., Beltagy, I., Downey, D., Weld, D. (2020). SPECTER: Document-level Representation Learning using Citation-informed Transformers. Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, 2270–2282. https://doi.org/10.18653/v1/2020.acl-main.207
5. Connelly, S. (2019, April 1). Practical BM25 - Part 1: How Shards Affect Relevance Scoring in Elasticsearch. Elastic Blog. https://www.elastic.co/de/blog/practical-bm25-part-1-how-shards-affect-relevance-scoring-in-elasticsearch
6. COVID-19 Open Research Dataset Challenge (CORD-19). (2021, January 19). Kaggle. https://www.kaggle.com/allen-institute-for-ai/CORD-19-research-challenge
7. Elasticsearch B.V. (n.d.-a). Multi-match query — search Reference [7.10]. Elastic. Retrieved 22 January 2020, from https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-multi-match-query.html#_literal_fields_literal_and_per_field_boosting
8. Elasticsearch B.V. (n.d.-b). Similarity module — search Reference [7.10]. Elastic. Retrieved 18 January 2021, from https://www.elastic.co/guide/en/elasticsearch/reference/current/index-modules-similarity.html
9. Eychaner, B. (2020, May 14). Exploration of Document Clustering with SPECTER Embeddings. Medium. https://medium.com/@beychaner/exploration-of-document-clustering-with-specter-embeddings-7d255f0f7392
10. Porter Stemming Algorithm. (2006). Tartarus. https://tartarus.org/martin/PorterStemmer/
11. Princeton University Department of Psychology. (n.d.). WordNet — A Lexical Database for English. Wordnet.Princeton. Retrieved 22 January 2021, from https://wordnet.princeton.edu/
12. Snowball. (n.d.). Snowballstem. Retrieved 22 January 2021, from https://snowballstem.org/The Allen Institute for Artificial Intelligence. (n.d.). CORD-19 Historical Releases. Ai2-Semanticscholar-Cord-19. Retrieved 15 January 2021, from https://ai2-semanticscholar-cord-19.s3-us-west-2.amazonaws.com/historical_releases.html
13. trec_eval: calculating scores for evaluation of information retrieval. (2016, February 19). Zipf's Law. https://zipfslaw.org/2016/02/19/trec_eval-calculating-scores-for-evaluation-of-information-retrieval/
14. Turnbull, D. (2020, September 9). BM25 The Next Generation of Lucene Relevance. OpenSource Connections. ://opensourceconnections.com/blog/2015/10/16/bm25-the-next-generation-of-lucene-relevation/
15. Turnbull, D., Berryman, J. (2016). Relevant Search: With applications for Solr and Elasticsearch. O'Reilly.

16. University of Glasgow, School of Computing Science. (n.d.). Divergence From Randomness (DFR) Framework. Terrierteam. Retrieved 20 January 2020, from http://terrier.org/docs/v3.5/dfr_description.html
17. Wang, L. L., Lo, K. (2020). Text mining approaches for dealing with the rapidly expanding literature on COVID-19. Briefings in Bioinformatics, 1–15. https://doi.org/10.1093/bib/bbaa296