VOH.CoLAB at TREC 2020 Precision Medicine Track*

Miguel D. Cardoso[1,2] and Flávio Martins[1,2,3]

[1] VOH.CoLAB, Universidade NOVA de Lisboa, Lisboa, Portugal
[2] NOVA LINCS, Universidade NOVA de Lisboa, Caparica, Portugal
[3] Comprehensive Health Research Centre (CHRC), Lisboa, Portugal
msd.cardoso@campus.fct.unl.pt flavio.martins@vohcolab.org

Abstract. This paper describes our participation in the Scientific Abstracts task of the TREC 2020 Precision Medicine Track. We present our approach and the methods implemented, including both submitted runs and several post-mortem experiments using different methods.
We performed experiments with Drugbank-based synonym expansion, Rocchio-based pseudo-relevance feedback, and neural re-ranking using the BioBERT biomedical pre-trained language models. In our evaluation, the Rocchio-based pseudo-relevance feedback method was the best performing method. Finally, we found that metadata and other textual fields in the document (e.g., journal name), are useful for retrieval and, when indexed, can improve recall-oriented metrics considerably leading to improvements in retrieval performance across the board.

Keywords: Precision Medicine, Information Retrieval, TREC

1 Introduction

Precision Medicine is a medical model that proposes the customization of healthcare, with medical decisions, treatments, practices, or products being tailored to a subgroup of patients, instead of a one-drug-fits-all model. This medical model leverages research on diseases and treatments to promote such customized healthcare to a patient. However, poor accessibility to the growing amount of research prevents a quick and easy use of such information creating obstacles for a successful application of Precision Medicine.

The TREC Precision Medicine Track (TREC PM), was created to try to close the gap between the conceptualization and the practice of the model. It challenges participants to explore methods that leverage existing document collections and patient information and build search engines to aid clinical staff in the tailoring of patient treatment. TREC has hosted several different challenges in related clinical tracks throughout the years, such as TREC Clinical Decision Support

* This work is supported by DSAIPA project FrailCare.AI (DSAIPA/0106/2019/02) and by NOVA LINCS (UIDB/04516/2020) with the financial support of FCT – Fundação para a Ciência e a Tecnologia, through national funds.

Table 1. TREC PM 2020 Scientific Abstracts task example topics.

	Patient 1	Patient 2	Patient 3
Disease	colorectal cancer	non-small cell carcinoma	acute myeloid leukemia
Gene	ABL1	ALK	ALK
Treatment	Regorafenib	Alectinib	Gilteritinib

Track (TREC CDS), which ran for three years between 2014–2016, which was the precursor of the TREC Precision Medicine Track (TREC PM) introduced in 2017. In the 2020 edition of TREC PM, the challenge is to rank scientific abstracts from PubMed according to their relevance for supporting a given treatment for a given patient's disease and a known gene variant.

In this paper we describe our two submitted runs to the TREC 2020 PM track. In addition, we present a number of post-mortem experiments and describe our findings with a brief discussion of possible future work.

2 Track Description

In contrast with the 2019 edition, which promoted two tasks, one on Clinical Trials and another on Scientific Abstracts, this years' TREC Precision Medicine track focused only on a Scientific Abstracts task. The overall task formulation is similar to the 2019 edition of the task. However, this year, the challenge was to rank scientific abstracts from PubMed according to their relevance for supporting a given 1) treatment, for a 2) patient's disease, and a 3) known gene variant.

In order to evaluate the performance of our experimental methods, we dropped the field "treatment" from the 2019 topics. This was necessary for a better fit between the 2019 topics and the new task formulation in 2020 allowing it to be used as a training set and tuning any hyper-parameters.

2.1 Topics

For the 2020 edition, assessors initially developed a total of 40 topics. Each topic, represented using a XML document, describes a patient using the following structure with three fields: **Disease** name; **Gene** affected; **Treatment** proposed.

2.2 Scientific Abstracts Task

The document collection used for the 2020 edition is the same 2018 mid-december snapshot of PubMed abstracts used for the 2019 edition. It contains more than 30M (31,677,119) abstracts. These PubMed abstracts hold the information about its authors, its MeSH terms, the abstract itself and other fields of information.

2.3 Drugbank

Drugbank is a website where anyone can retrieve information regarding treatments, genes and other multitude of health related concepts. We used the data available from this website and automatically retrieved synonyms, descriptions and indications of the treatments presented in the topics. Ultimately we only used synonyms information, with an average of 1.27 synonyms per treatment.

3 Evaluation and Methods

With the goal of creating a Precision Medicine search engine, we explored several strategies described in this section. We used Elasticsearch as our database to store the documents, and also as our primary search engine, since it has built-in information retrieval methods.

Elasticsearch was our go-to decision since it is widely used for Information Retrieval and it is *open-source*. With Elasticsearch as our primary search engine, we then defined strategies to expand the queries but also explored re-rank methods. We used stop words used in the PubMed search engine, `pubmed list`[4], and the Galago `rmstop list`[5] when computing relevance models.

3.1 Hardware

All of the development occurred in JupyterHub hosted on a server. Furthermore, the dataset and Elasticsearch were also hosted on the same machine with the following hardware specifications:

Motherboard ASUS P9X79 PRO
CPU Intel Core i7-3930K Processor (6 cores, 12 threads, 12M)
RAM 64 GB DDR3 (1333Mhz)

3.2 Collection Pre-processing

The dataset contains several compressed XML files (.gz), that were parsed and indexed in two different ways into Elasticsearch. We named these indexes, *filtered corpus* and *unfiltered corpus*. However, the latter was only created post-mortem. We note that *unfiltered corpus* index was later implemented due to the poor recall of the methods applied on the *filtered corpus* index. Thus we speculated that we were removing relevant information.

[4] `https://github.com/igorbrigadir/stopwords/blob/master/en/pubmed.txt`
[5] `https://github.com/igorbrigadir/stopwords/blob/master/en/galago_rmstop.txt`

The filtered corpus Here we only indexed the abstracts that did not had animal or animals on its mesh terms. This filtering reduced the number of documents to 22,826,528 from 31,677,119, which means that 8,850,591 were documents that related to animals. Besides filtering out, we only used id, title, abstract text, mesh terms and substances fields from the original documents.

The Elasticsearch documents had the following schema:

id text
title text
body text
mesh text

The unfiltered corpus Here we did not filter out any documents. We indexed all the text in the PubMed Abstract concatenated into the body field. The total documents present in this index are 31,677,119.

The Elasticsearch documents had the following schema:

id text
title text
body text

3.3 Runs

We used Okapi BM25 as the retrieval function to get the documents from Elasticsearch. BM25 is a bag-of-words function that ranks a set of documents according to the given query, ignoring the proximity of its terms in the document itself. BM25 Equation (1) makes use of well-known notion of inverse document frequency (IDF) Equation (2) to score word importance in the collection (rarity).

We did not perform any type of tuning on its parameters and used the recommended default parameters of $k1 = 1.2$ and $b = 0.75$

$$\mathrm{BM25}(D,Q) = \sum_{i=1}^{n} \mathrm{IDF}(q_i) \cdot \frac{f(q_i,D)\cdot(k_1+1)}{f(q_i,D)+k_1\cdot\left(1-b+b\frac{|D|}{avgdl}\right)} \tag{1}$$

$$\mathrm{IDF}(q_i) = \ln\left(\frac{N-n(q_i)+0.5}{n(q_i)+0.5}+1\right) \tag{2}$$

Run run_bm25 For our baseline, we queried Elasticsearch by all the elements in the presented topic, (treatment, gene, disease). Giving no specific weight to any element of the topic, besides stating that the documents found must have both the gene and disease but also should (not must) have the treatment in it.

Run bm25_synonyms This run expands the information of the query with all the synonyms found for given treatment in each topic using Drugbank. The remaining execution of this run is similar to run_bm25.

3.4 Not Submitted Experiments

BioBERT We experimented with BioBERT [6] [3] which is a fine-tuned version of the well-known language model Bert [2]. BioBERT was fined-tuned using PubMed abstracts, therefore, the use of this language model apriori appeared to be quite useful for this task. First, we used this pre-trained model to compute both the embeddings of the retrieved documents and the query so that can we later use them. These initial documents were retrieved using the same method as in *run_bm25*. Here we experimented with up to 10000 initial retrieved documents, starting at 1000 with a step of 250. The embeddings were used to re-rank the retrieved documents according to their cosine similarity to the query.

Afterwards, we decided to linearly combine the bm25 scores that elastic search produces, and the cosine similarity scores. The BM25 scores had to be normalized between $[0, 1]$ since the cosine similarity also ranges from $[0, 1]$. We used this final combined score to re-rank the retrieved documents.

In order to get the weights for the linear combination, we performed a linear search starting with $\alpha = 0$, increasing it by 0.1 in each step, totaling in 10 steps. At each step we validated the results against the 2019 relevance judgments. The parameter α defines the weight given to the cosine similarity score. Henceforth, the higher the α, the higher the weight given to BioBERT cosine similarity between the query and the given document.

We optimized the weights and number of documents retrieved based on Rprec, map and num_rel_ret, in this exact order. We found that $\alpha = 0.4$ and a number of initial retrieved documents over 2000, those metrics were optimal.

In this experiment, we also tested different ways of building the query inspired by Dai et al. [1]. However, we did not notice any improvement. We also attempted to re-rank the documents based on the top-k initially retrieved documents and not only on the query, which also did not yield any improvement.

Rocchio Expansion We leverage the Rocchio formula as seen in Eq. (3)

$$\vec{Q_m} = (\alpha \cdot \vec{Q_o}) + \left(b \cdot \frac{1}{|D_r|} \cdot \sum_{\vec{D_j} \in D_r} \vec{D_j} \right) - \left(c \cdot \frac{1}{|D_{nr}|} \cdot \sum_{\vec{D_k} \in D_{nr}} \vec{D_k} \right), \quad (3)$$

[6] `https://github.com/dmis-lab/biobert`

where $Q_o, D_r, D_n r$ are the original query, top documents and bottom documents, respectively. In order to get D_j and D_k, we retrieved two sets of 1000 documents using the methods described in Section 3.3. We then selected the top 100 and bottom 5 documents, defined the parameters *a*,*b* and *c*, to compute Q_m.

Having $\vec{Q_m}$ calculated, we retrieved the top 75 terms of the query with the highest score, joined all of those terms into a query boosting each term with its score and queried Elasticsearch once again, retrieving another 1000 documents. We fined tuned Equation (3) parameters *a*,*b* and *c* using the 2019 results. We used a = 6, b = 5 and c = 5 as the final parameters.

Rocchio Rerank Alternatively to Section 3.4, after the initial retrieval and consequent $\vec{Q_m}$ computation, here we decided to use rocchio to re-rank the initially retrieved documents. In this experiment we used Q_m as the ground truth vector and computed the similarity scores of all documents against this vector.

RM3 Expansion For this experiment we created a feedback relevance model using the top 20 documents and then selected the top 20 terms. Having these 20 terms, we normalized their weights between [0,1], and queried Elasticsearch once again, with all of these 20 terms boosting them with their respective weights. This feedback relevance model, $P(w \mid \theta_F)$, is computed using the following equation:

$$\begin{aligned} P(w \mid \theta_F) &\propto \sum_{d \in \mathcal{R}} P(w \mid d) \cdot \underbrace{P(q \mid d) \cdot P(d)}_{\text{document score}} \\ &\propto \frac{1}{|\mathcal{R}|} \sum_{d \in \mathcal{R}} P(w \mid d) \cdot \mathrm{BM25}(q, d), \end{aligned} \tag{4}$$

where $\mathcal{R}$ is the top N relevant documents and BM25(q, d) is the BM25 score returned by Elasticsearch for document d matching query q. The relevance model $P(w \mid \mathcal{R}) \approx P(w \mid \theta_F)$ for query q is a weighted average of the terms in the top documents retrieved, where the weights are the BM25 scores for the query q. After Eq. (4) is computed, we interpolate the original query model weights, θ_q, with the estimated relevance model, θ_F, using parameter $\lambda = 0.5$ as follows,

$$P(w \mid \theta_{q'}) = \lambda \cdot P(w \mid \theta_q) + (1 - \lambda) \cdot P(w \mid \theta_F). \tag{5}$$

4 Results

In this section we display the benchmarks of all of our methods against the judgments of the 2020 edition. On this edition, the evaluation was divided into two phases. The phase 1 judgments evaluate if the documents retrieved are on the precision medicine topic and how much they match towards the query tokens individually. While the phase 2 judgments evaluate if the documents retrieved are relevant towards the relevant query as an whole. The phase 2 judgments were constructed by manually labeling the documents up to 100 documents by topic.

Table 2. TREC PM 2020 Scientific Abstracts task Phase 1 results.

	filtered corpus			unfiltered corpus		
	infNDCG	P10	Rprec	infNDCG	P10	Rprec
median	**0.4316**	**0.4645**	**0.3259**	0.4316	0.4645	0.3259
run_bm25*	0.3587	0.4452	0.2521	0.4538	0.5000	0.3486
run_bm25_syn*	0.2357	0.2839	0.1706	0.3587	0.3677	0.2862
rocchio_cosine	0.3567	0.4452	0.2478	0.4538	0.5000	0.3486
rocchio_exp	**0.3941**	0.4710	0.2892	**0.4843**	**0.5032**	**0.3698**
BioBERT_40%_2000	0.3567	0.4452	0.2477	0.4518	0.5000	0.3486
RM3	0.3024	0.3742	0.2148	0.4048	0.4226	0.2803

* *official runs*

Table 3. TREC PM 2020 Scientific Abstracts task Phase 2 results.

	filtered corpus		unfiltered corpus	
	ndcg@30	ndcg@5	ndcg@30	ndcg@5
median	0.2857	0.2529	0.2857	0.2529
run_bm25*	0.3009	0.2706	0.3042	0.2612
run_bm25_syn*	0.2476	0.2242	0.2701	0.2249
rocchio_cosine	0.3039	0.2706	0.3042	0.2612
rocchio_exp	**0.3332**	**0.3045**	**0.3228**	**0.2714**
BioBERT_40%_2000	0.3019	0.2706	0.3042	0.2612
RM3	0.2731	0.2597	0.2602	0.2391

* *official runs*

As seen in Table 2 none of our runs and post-mortem experiences were above the median using the filtered corpus index. However, when we used the unfiltered corpus index, the rocchio based method showed better results. However, in phase 2 as shown in Table 3, the better results were on the filtered corpus index. In addition we can also note that the Rocchio methods outperforms RM3 relevance model in all collections and displayed metrics.

The synonym-based query expansion methods always yields worse results due to the fact that some of the synonyms presents on Drugbank are their chemical names. For example, *Alectinib* had the synonym *6-dimethyl-8-[4-(morpholin-4-yl)piperidin-1-yl]-11-oxo-6*, which after tokenization creates unnecessary tokens that promote worse results. In the previous results tables we decided to not show the results of these methods since run_bm25_syn already performs worse than run_bm25. Thus, applying rocchio cosine or expansion, RM3 or BioBERT methods would consequently yield poorer results than the same methods that used the run_bm25 as the first step.

It is a common pattern in Information Retrieval to expand the information presented in the query or even in the text to retrieve before storing it. However, in our early experiments with this approach we did not see any improvement on the results. Henceforth, we did not explore further besides expanding the query with synonyms of the treatments. In this experiments we also concluded that big queries slow down the system significantly. Expanding the document itself with related data from *SNOMED CT*[7] or other healthcare sources at indexing time, instead of expanding the query, may yield a more efficient system.

Initially, we hypothesized that pre-filtering the corpus could help and make the search more effective. However, this was not the case, as shown in Table 2, where we were merely interested on how well the system matched the topics as individual tokens. We believe that in phase 1, the filtering removed metadata that could have been helpful. In phase 2 results, shown in Table 3, the behaviour was mostly similar, except on the non-submitted experiments rocchio_exp and RM3, where the filtered corpus showed a slight improvement. The reason for these results deserves further investigation, nonetheless, we suspect that it is mostly due to the relevance judgments two-phase annotation model of the task. For phase 2, accessors labeled the documents manually by reading them, thus the documents were labeled solely by their content, which is the same information that the methods applied for the filtered corpus had access to.

In addition, we experimented with BioBERT but it did not yield good results when compared to the Rocchio Expansion method. BioBERT has a maximum limit of context it can work with, which is a limiting factor when working with long documents. We did not employ any method to compensate this limitation and just cropped the document so that it could fit in BioBERT's context window. Furthermore, in our experiments, BioBERT promotes BM25 low-scoring documents, which ultimately yields poorer results if we increase the number of re-ranked documents.

5 Conclusions and Future Work

Finally, we conclude that in this multi-stage retrieval pipeline, the initial method of retrieval is of the utmost importance. Most of the methods that we employed a the second stage of retrieval, especially Rocchio and other pseudo-relevant feedback methods, require a good first stage retrieval to get an initial set of useful and relevant documents. Therefore, to improve the chances of the second stage method, the first retrieval should be tuned for P5, P10, and P20. In addition, we also conclude that no pre-filtering should be done since there is meta-data and other fields that are relevant as they improve the results.

As future work we look towards document expansions methods at indexing time and adapting BioBERT methods to longer documents. We will explore methods making use of medical knowledge-bases as well as semantic search using neural-based retrieval methods to improve the second stage retrieval.

[7] http://www.snomed.org/

References

1. Dai, Z., Callan, J.: Deeper text understanding for ir with contextual neural language modeling. Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (Jul 2019), http://dx.doi.org/10.1145/3331184.3331303
2. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding (2019)
3. Lee, J., Yoon, W., Kim, S., Kim, D., Kim, S., So, C.H., Kang, J.: BioBERT: a pre-trained biomedical language representation model for biomedical text mining. Bioinformatics (09 2019), https://doi.org/10.1093/bioinformatics/btz682