University of Glasgow Terrier Team and UFMG at the TREC 2020 Precision Medicine Track

Alberto Ueda
UFMG, Brazil
ueda@dcc.ufmg.br

Rodrygo L. T. Santos
UFMG, Brazil
rodrygo@dcc.ufmg.br

Craig Macdonald, Iadh Ounis
University of Glasgow, UK
firstname.lastname@glasgow.ac.uk

ABSTRACT

For TREC 2020, we explore different re-ranking strategies by integrating PyTerrier — a framework which allows us to build advanced retrieval pipelines — with state-of-the-art BERT-based contextual language models. To produce the initial ranking, we use traditional retrieval models, such as Divergence From Randomness (DFR) weighting models. Then, we assign new scores for each document with BERT-based models, such as SciBERT and ColBERT. Finally, we re-rank the documents using combinations of those scores, via linear combination or learning-to-rank. We also conduct experiments with models able to leverage the structure information of the documents, i.e., by assigning different scores for their individual sections (e.g., Background, Results, Conclusions). We submitted five official runs, `uog_ufmg_DFRee`, `uog_ufmg_s_dfr0`, `uog_ufmg_s_dfr5`, `uog_ufmg_sb_df5`, `uog_ufmg_secL2R`. Our results in TREC 2020 confirm our main observations in our tests using the TREC 2019 test collection, showing that re-ranking strategies such as simple linear combinations of DFR and SciBERT scores (`uog_ufmg_sb_df5`) are competitive and outperform the TREC median performance.

1 INTRODUCTION

In our first participation in the TREC Precision Medicine track, we opt to evaluate the effectiveness of traditional retrieval models integrated with state-of-the-art contextual language models, in the domain of medical literature. Inspired by our participation in the TREC Deep Learning Track [15, 16], we leverage our existing team infrastructure, while enhancing it with the integration of new deep learning techniques — some of them specific to scientific literature, such as the SciBERT pre-trained model. In particular, we use PyTerrier [11],[1] a Python framework which allows advanced retrieval pipelines to be expressed, and evaluated, in a declarative manner similar to their conceptual design. We integrate PyTerrier with a toolkit which allows us to straightforwardly handle BERT-based models (CEDR [9], described in Section 4) and then conduct several experiments on the application of different standard IR models (e.g., BM25, DPH, DFRee) and BERT-based models (e.g., BERT$_{CLS}$ [12], SciBERT [3], CEDR_pacrr [9], and the more recently proposed ColBERT [5]). To evaluate the performance of those models and choose which of them would be submitted, we use the datasets provided by previous editions of the track, namely, TREC-PM 2017-2019[13]. We combine the different scores given by each model using linear combinations and the LightGBM implementation of LambdaMART.[2] Moreover, we conduct initial experiments using models that leverage the inherited structure of biomedical abstracts, i.e., the presence of named sections such as Background, Results, and Conclusions.

[1] https://github.com/terrier-org/pyterrier
[2] https://lightgbm.readthedocs.io

The remainder of this paper is structured as follows: Section 2 describes our settings for indexing the TREC-PM 2020 data; Section 3 discusses the models used to produce the initial candidate set of documents, i.e., the first-phase retrieval, while in Section 4 we show detailed information about the contextual language models used in the second-phase retrieval; Section 5 provides the full list of the official submissions we made to TREC and the strategies used in each submission; Section 6 and Section 7 discuss our results and provide concluding remarks, respectively.

2 INDEXING

To index the 29,138,919 documents (i.e., biomedical abstracts) of the Pubmed/MEDLINE 2019 Baseline, we use Terrier [10] v5.3. For each document, we index only its identifier (tag `PMID`), title (tag `articletitle`) and text (tag `abstracttext`). We initially also indexed its MeSH Terms (found in tags such as `descriptorname` inside a (`meshheadinglist`)), however, in our experiments they did not improve the first-phase nor the second-phase retrieval, hence they were discarded. For all documents, we identify named sections such as Background and Results (e.g., `NlmCategory="BACKGROUND"` inside an `abstracttext`), and we index them as Terrier fields. We discard all the other information from the original corpus.

We use the following indexing configurations:

- Positions: to record the positions that terms occur in;
- Fields: to record the frequencies of terms occurring in different parts of the document: abstract title, text, and named sections (as described previously);
- Stemming & Stopwords: to improve retrieval recall and precision, we applied Porter's stemmer and Terrier standard stopwords removal at indexing time.

We apply the classical two-pass indexing of Terrier[3] to create both a direct and an inverted index of the corpus, to support query expansion and other retrieval techniques. We also record the raw text of the document titles and content as metadata, to allow deep learning upon these textual representations, as discussed further in Section 4. In the next section, we describe how we use that index to create the initial candidate set of documents in our first-phase retrieval.

3 FIRST-PHASE RETRIEVAL

To avoid computing expensive ranking models for large collections such as Pubmed/MEDLINE, with millions of documents, we take a common approach in most modern search engines, namely the *cascading architecture*: where an initial ranking of k documents with respect to a query q is thereafter re-ranked by the application of more advanced and complex ranking techniques, such as neural

[3] https://github.com/terrier-org/terrier-core/blob/5.x/doc/

rankers. The initial ranking is the result of the referred first-phase retrieval. Using PyTerrier, we employ three standard models of Information Retrieval (IR) — BM25, DPH, and DFRee — in two different versions of each of them, with/without query expansion (QE):

- DFRee is a hyper-parameter-free implementation of the Divergence From Randomness hyper-geometric weighting model [2];
- DPH [14] is the DFR-based weighting model with Popper's normalization (the 'P' in the DPH name). DPH is the default weighting model of PyTerrier;
- BM25 (i.e. Okapi BM25) is a popular weighting model to address various retrieval tasks. In contrast to DPH and DFRee, we note that the BM25 hyper-parameters can be fine-tuned, providing corpus-fitted models. However, we use the PyTerrier default parameter values in our experiments.

For each topic, the query consists of the concatenation of the contents of the `disease`, `gene`, and `treatment` tags provided in the TREC topics file. To expand the original queries, we used the default parameters of query expansion in PyTerrier, in which, for each query q, the top ten most informative terms of the top three documents retrieved were added to q with the weighting scores given by the Bo1 model [1]. Although query expansion (QE) improved the recall of BM25 and DPH, as expected, the DFRee ranking model without QE, surprisingly, outperformed all of the models considered, including those with QE. Thus, we decided to use DFRee without QE as our first-phase retrieval model in all further experiments. We also experimented with different values for the size k of the initial candidate set, from 200 to 5,000 documents per query. In these experiments, we found that using $k = 1000$ provided similar performances to higher values of k in the final ranking, not only with respect to precision but also in terms of recall. Therefore, we use this setting in all experiments described in this paper. We easily deploy all the aforementioned ranking models using PyTerrier.

Next, we describe the second-phase retrieval, where we employ more tailored ranking models to assign new scores to each document.

4 SECOND-PHASE RETRIEVAL

Since it has been proposed, the BERT contextual language model [4] (LM) and many of its extensions such as ALBERT [6] and RoBERTa [8] have showed significant improvements in several natural language processing (NLP) tasks, such as question answering, named entity recognition, and passage retrieval [9, 12]. In particular, in the BERT-based text re-ranking approach of [12], the query and document are concatenated with a `[SEP]` in between, and a `[CLS]` token at the end. The final relevance estimate of the document for a query is obtained from the embedding of the `[CLS]` token. We also use this approach in all BERT-based experiments for TREC-PM.

More recently, some variants of the BERT language model have been proposed for the scientific literature domain, such as SciBERT [3] and BioBERT [7]. In practice, each LM variant is focused on capturing the specific context of texts in the domain to better represent the semantics of words in a vocabulary. In this work, we apply the SciBERT LM, which we have found to be more effective in biomedical search tasks compared to BERT_{CLS} and BioBERT [3].

Listing 1: Example how to apply a BERT-based model using PyTerrier and CEDR. In particular, in this example we create and test the SciBERT-pm, a SciBERT model fine-tuned in previous TREC-PM track editions.

```
1  # 1st-phase retrieval: DFRee
2  dfree = pt.BatchRetrieve(index, wmodel="DFRee",
3    metadata=["docno","articletitle","abstracttext"])
4
5  # 2nd-phase retrieval: SciBERT re-ranking DFRee
6  scibert_pm = dfree >> CEDRPipe(modelname="SciBERT")
7
8  # fine-tune SciBERT on training & validation topics
9  scibert_pm.fit(topicsTrain, qrelsTrain,
10                topicsValid, qrelsTrain)
11
12 # evaluate performance
13 pt.Experiment([dfree, scibert_pm],
14               topicsTest, qrelsTest,
15               ['map', 'ndcg'],
16               names=["DFRee", "SciBERT-pm"])
```

To use SciBERT in our experiments, we add a simple integration between PyTerrier and the CEDR [9] toolkit, described next.

CEDR: Contextualized Embeddings for Document Ranking. The CEDR toolkit[4] allows us to use state-of-the-art BERT language models, including the original BERT_{CLS} model (with no additional fine-tuning, and known as "vanilla BERT" in CEDR) and the PACRR model [9]. Moreover, it provides us with the possibility of fine-tuning the BERT-based models for different corpora. We made experiments with fine-tuned versions of the BERT_{CLS}, CEDR-PACRR, and SciBERT models, but the former two were discontinued as they were observed to exhibit less effectiveness than using a fine-tuned SciBERT model, described next.

SciBERT. We implement an integration class for SciBERT in CEDR, based on the BERT_{CLS} model, differing only by the pre-trained models for the vocabulary and text tokenizer which are loaded. Then, we fine-tune the SciBERT and other BERT-based models in our corpus. To fine-tune the pretrained SciBERT model, we use the 80 queries from TREC-PM 2017 and 2018 (72 for training and 8 for validation) and the top 1,000 articles retrieved by DFRee for each query, over a maximum number of epochs equal to 100 with an early-stopping of the fine-tuning process after 20 iterations without validation improvements (i.e., parameter $patience = 20$). For all BERT-based text re-rankings in this work, we use the simple concatenation of the article's title and abstract as the document representation. To change the resulting fine-tuned model from the original one, we name it SciBERT-pm, in contrast to the original SciBERT model. Listing 1 illustrates an example of how we create and apply the SciBERT-pm model using PyTerrier. Note that in Line 6 we use PyTerrier's operator >> – known as "Then" – which indicates that we are chaining two pipelines, i.e., applying one PyTerrier transformation to re-rank the results of the proceeding transformer. For instance, in this example, the `scibert_pm` pipeline is deployed as the SciBERT re-ranking of the initial documents retrieved by the DFRee model. However, it is noteworthy to mention that both the first-phase and second-phase retrieval will be executed only when

[4] https://github.com/Georgetown-IR-Lab/cedr

it is necessary: for instance, in our case, in Line 9, when PyTerrier needs the results to start the fine-tuning process.

After fine-tuning the model (with `scibert_pm.fit()`, we may store it for reuse in other experiments later. The output of `pt.Experiment()` will be a table comparing the results of the DFRee and SciBERT-pm retrieval pipelines, according to the given TREC qrels file, in terms of MAP (Mean Average Precision) and nDCG (normalized Discounted Cumulative Gain) .

ColBERT. We also employ ColBERT [5], a recently proposed contextual embedding with promising results for IR ad hoc tasks. ColBERT has been shown to be faster and at least as effective as BERT_{CLS}, which was confirmed in our experiments. However, in our tests for TREC-PM, ColBERT was not as effective as our SciBERT-pm model, so the former was not used in further experiments. The reason for this disappointing performance might be related to the fine-tuning process applied to SciBERT-pm. At the time of writing this paper, we did not fine-tune ColBERT in our TREC-PM data, leaving this as future work.

Using Document Structure. This approach builds on the observation that different biomedical abstract sections — e.g., Background and Conclusions — may convey different semantics. We hypothesize that the linguistic nuances of each section can be modelled individually with SciBERT and combined to improve our ranking models. To test this hypothesis, we once again use the top 1,000 abstracts retrieved by DFRee for each of the TREC-PM 2017 and 2018 queries. However, in contrast to the SciBERT-pm model, which is fine-tuned only once using the entire abstract, we produce one fined-tuned SciBERT model per each identified abstract section. We denote these novel models as SciBERT-s (i.e., section-based SciBERT). Thus, besides the original SciBERT-pm score, in this approach, we also have additional SciBERT-s scores, one score for each abstract section. In our initial experiments for TREC, we assume that all biomedical abstracts inherited have four sections (namely, Background, Methods, Results, and Conclusions). For the documents without the named sections, we use logistic regression classifier[5] to determine the section of each abstract sentence, trained on 30k sentences from the corpus that were separated into section by their respective authors.

Re-Ranking. In order to leverage both standard retrieval (first-phase) and semantic (second-phase) scores, we use as simpler score aggregation alternatives, for each document d and query q, linear combinations between the DFRee and SciBERT-pm scores, varying the weighting parameter *alpha* (α) from 0 to 1, as follows:

$$score(d,q) = \alpha \times score_{m_1}(d,q) + (1-\alpha) \times score_{m_2}(d,q)$$

where m_1 are statistic-based models – such as DFRee – and m_2 are semantic-based models, such as SciBERT-pm. In particular, the combination of DFRee and SciBERT-pm was the best we found in our experiments based on the past editions of TREC-PM, hence we focus on this combination in our TREC submissions this year. In general, our experiments showed that these combinations are consistently better than the models evaluated separately, also considering other statistic-based models, such as BM25 and DPH, and other semantic-based models, such as PACRR and ColBERT. We aim to investigate and explain the reasons for such a behavior in future studies.

For our structure-based experiments, in which we have more than two features as input — given by the section-based models — we use a listwise learning-to-rank (LTR) method to combine the features. Specifically, we adopt the LightGBM 3.0[6] implementation of LambdaMART. To train the LTR model, we use 80 queries (TREC-PM 2017 and 2018), of which the first 72 queries we use as training set and the last 8 queries as our validation set. For the configuration settings, we use the default parameter values of LightGBM ranker.

5 SUBMITTED RUNS

We submitted the following five runs to the TREC 2020 Precision Medicine Track:

- `uog_ufmg_DFRee`: this run is the result of the first-phase retrieval with PyTerrier's batch retrieval with DFRee, as detailed in Section 3;
- `uog_ufmg_s_dfr0`: this run is the result of the second-phase retrieval with the SciBERT-pm re-ranking, using DFRee as the initial ranking model (c.f. `uog_ufmg_DFRee`), as detailed in Section 4;
- `uog_ufmg_s_dfr5`: this run is the linear combination ($\alpha = 0.5$) of the scores of DFRee and SciBERT-pm (`uog_ufmg_DFRee` and `uog_ufmg_s_dfr0`, respectively);
- `uog_ufmg_sb_df5`: this run is the result of the same methodology used in the previous `uog_ufmg_s_dfr5` run, but with a different query tokenization;
- `uog_ufmg_secL2R`: this run is the result of the second-phase retrieval with the structured-based SciBERT-s re-ranking, as detailed in Section 4.

6 RESULTS & ANALYSIS

Table 1 summarizes the results of our submitted runs in TREC-PM 2020 for the evaluation metrics Inferred Discounted Cumulative Gain (infNDCG), Precision@10 (P@10), and R-precision (R-prec), the official evaluation metrics of the track.

Firstly, we note that `uog_ufmg_DFRee` is better than TREC median for P@10 and their performances are close to each other for infNDCG and R-prec. This shows that our first-phase retrieval with DFRee is a reasonably good baseline and a reliable starting point to subsequent re-rank approaches. As expected by our experiments in TREC-PM 2019, the pure re-ranking with SciBERT-pm (`uog_ufmg_s_dfr0`) is less effective than the DFRee model. The goal with this approach is to allow us to combine DFRee scores with semantic-based models scores, in particular, SciBERT-pm.

However, our linear combination `uog_ufmg_s_dfr5` in TREC-PM 2020 did not reproduce the same improvements over its baselines as observed in TREC-PM 2019, although it slightly outperformed the DFRee run for P@10. We will investigate the differences between the TREC-PM 2019 and 2020 results as future work. Nonetheless, our best run `uog_ufmg_sb_df5` followed the same experimental methodology of `uog_ufmg_s_df5`, but with a different query tokenization. The run `uog_ufmg_sb_df5` outperformed not only all our submitted runs, but also the TREC median on all measures.

[5]Namely, SKLearn's LogisticClassifier: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

[6]https://github.com/microsoft/LightGBM

Table 1: Results of the five submitted runs for TREC-PM 2020. The last two rows show the median and best performances over all submissions, aggregated by average.

Run	infNDCG	P@10	R-prec
uog_ufmg_DFRee	0.418	0.500	0.315
uog_ufmg_s_dfr0	0.280	0.332	0.197
uog_ufmg_s_dfr5	0.408	0.513	0.309
uog_ufmg_sb_df5	**0.498**	**0.548**	**0.391**
uog_ufmg_secL2R	0.401	0.455	0.311
Median	0.432	0.456	0.326
Best	0.696	0.722	0.568

Finally, the results of run uog_ufmg_secL2R reflect our first experiments with the structured-based framework, i.e., when we use multiple SciBERT-s models to re-rank the different abstract sections. This approach outperformed the SciBERT-pm model (i.e., uog_ufmg_s_dfr0) on all metrics, which suggests that this is a promising research direction. However, the structure-based run was not consistently better than the other runs or the TREC median. To overcome this, we aim to further study the exact benefits we can achieve when we use structure-based information.

It is noteworthy to mention that the reported results above concern Phase 1 of the TREC-PM 2020 evaluation, i.e., for the relevance assessment. At the time of writing this paper, Phase 2, i.e., the evidence assessment, was not yet available to participants. The goal of this second phase of evaluation is as follows: given a pair of a disease-treatment as a query and a biomedical article retrieved, the aim is to determine how strong is the evidence brought by the article with respect to the pair disease-treatment. Articles with strong evidence should be ranked over articles with weaker evidence (whether positive or negative). In fact, we did not implement any specific methods to model the evidence strength of the articles. Instead, our approaches were purely relevance-oriented. We will present our obtained results in Phase 2 in the final version of the notebook paper.

7 CONCLUSIONS

In our participation in the TREC 2020 Precision Medicine Track we investigated the integration of domain-specific state-of-the-art contextual language models, such as SciBERT, with PyTerrier retrieval pipelines, in a two-phase retrieval process. In terms of effectiveness, we observed promising improvements using simple linear combinations such as uog_ufmg_sb_df5, outperforming the median performance of the TREC participants. Moreover, part of the experimental methodology built for this task was used in the critical and urgent task provided by TREC-Covid, within the context of the ongoing pandemic. Finally, as discussed in this paper, several interesting research directions were raised during the experiments that we conducted this year for this task.

ACKNOWLEDGMENTS

This study was financed in part by the *Coordenação de Aperfeiçoamento de Pessoal de Nível Superior*, Brazil (Capes), Finance Code 001.

REFERENCES

[1] Giambattista Amati. 2003. *Probability models for information retrieval based on divergence from randomness.* Ph.D. Dissertation. Department of Computing Science, University of Glasgow.

[2] Gianni Amati and Cornelis Joost Van Rijsbergen. 2002. Probabilistic Models of Information Retrieval Based on Measuring the Divergence from Randomness. *ACM Trans. Inf. Syst.* 20, 4 (2002), 357–389.

[3] Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. SciBERT: A pretrained language model for scientific text. *arXiv preprint arXiv:1903.10676* (2019).

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of NAACL*. 4171–4186.

[5] Omar Khattab and Matei Zaharia. 2020. ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT. In *Proceedings of SIGIR*. 39–48.

[6] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. *arXiv preprint 1909.11942* (2020).

[7] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2020. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics* 36, 4 (2020), 1234–1240.

[8] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv preprint 1907.11692* (2019).

[9] Sean MacAvaney, Andrew Yates, Arman Cohan, and Nazli Goharian. 2019. CEDR: Contextualized embeddings for document ranking. In *Proceedings of SIGIR*. 1101–1104.

[10] Craig Macdonald, Richard McCreadie, Rodrygo LT Santos, and Iadh Ounis. [n.d.]. From Puppy to Maturity: Experiences in Developing Terrier.. In *Proceedings of OSIR workshop at SIGIR*.

[11] Craig Macdonald and Nicola Tonellotto. 2020. Declarative Experimentation in Information Retrieval Using PyTerrier. In *Proceedings of ICTIR*. 161–168.

[12] Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage Re-ranking with BERT. *arXiv preprint arXiv:1901.04085* (2019).

[13] Kirk Roberts, Dina Demner-Fushman, Ellen M. Voorhees, William R. Hersh, Steven Bedrick, Alexander J. Lazar, and Shubham Pant. 2019. Overview of the TREC 2019 Precision Medicine Track. In *Proceedings of TREC*.

[14] Stephen E Robertson and Steve Walker. 1994. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *Proceedings of SIGIR*. 232–241.

[15] Ting Su, Xi Wang, Craig Macdonald, and Iadh Ounis. 2019. University of Glasgow Terrier Team at the TREC 2019 Deep Learning Track. In *Proceedings of TREC*.

[16] Xiao Wang, Yaxiong Wu, Craig Macdonald, and Iadh Ounis. 2020. University of Glasgow Terrier Team at the TREC 2020 Deep Learning Track. In *Proceedings of TREC*.