Poznań Contribution to TREC-PM 2020*

Jakub Dutkiewicz[1] and Czesław Jędrzejek[1]

Faculty of Computing, Poznań University of Technology, Plac Marii Skłodowskiej-Curie 5, 60-965 Poznań, Poland

Abstract. This paper describes our contribution to TREC PM 2020. We discuss the retrieval architecture used for our contribution. We split our system into four layers - preprocessing, representation, baseline and neural layer. We go over goals and specification of each layer. We conclude the section with description of our hardware setup. Then, we describe experiments conducted within this contribution - we discuss used data and retrieval models.The reranking gives little but noticeable improvement. Our results are significantly better than the median. Paper is concluded with a discussion over weaknesses and strengths of our approach, we briefly formulate what has to be done in the future.

1 Introduction

TREC PM 2020 track continues on lines of PM (2017, 2018, 2019) tracks with topics asking for precision medicine-related evidence to clinicians treating cancer patients. Documents in the form of The MEDLINE 2019 baseline (roughly mid-December 2018) of PubMed abtracts. This is the same corpus as last year. Topics are in a form of XML statements tagger with: Disease, Gene and Treatment. We have quite an experience in the medical search. We participated in TREC CDS 2016 [6], TREC PM 2017, TREC PM 2018 track [2], TREC PM 2019 [3];and in bioCADDIE 2016 [1]. There are two basic approaches to Information Retrieval:

- Classical:
- Neural networks (NN). Straight-forward implementation faces difficulty with only keywords. Lack of sentences prevents to use sequence approach as is dowe with query answering. Lack of or a small number of annotated cases (15-400) makes learning inefficient.

In this paper we expand our unsupervised approaches, which up to this point was using two steps:

- Baseline step: Divergence from Randomness, language models, within-document term-frequency (tf-idf),
- Query expansion (with word embedding – obtained by neural network based autoencoders),

* Supported by PUT statutory funds. One of the authors (CJ) acknowledges the nVIDIA GPU Grant of Quadro P6000 card.

The expansion comprises of the recently popular machine learning based extension. We use data from the previous competition - TREC PM 2019 in order to train a neural network, which is capable of enhancing our results. For the first time, we use the attention type network to rerank the result.

2 Retrieval Architecture

We propose a dedicated Retrieval Architecture, which consists of four major layers. These layers compute the output for the following tasks:

1. Preprocessing Layer: document parsing, preprocessing and initial extraction of information.
2. Baseline Layer: indexing and retrieval with classical methods.
3. Word Space Layer: generating a word vector space.
4. Neural layer: training and reranking.

Layers are executed separately, and each of them generates data specified by an interface. The general retrieval architecture is illustrated in Fig. 2. This section comprises of specification of each layer.

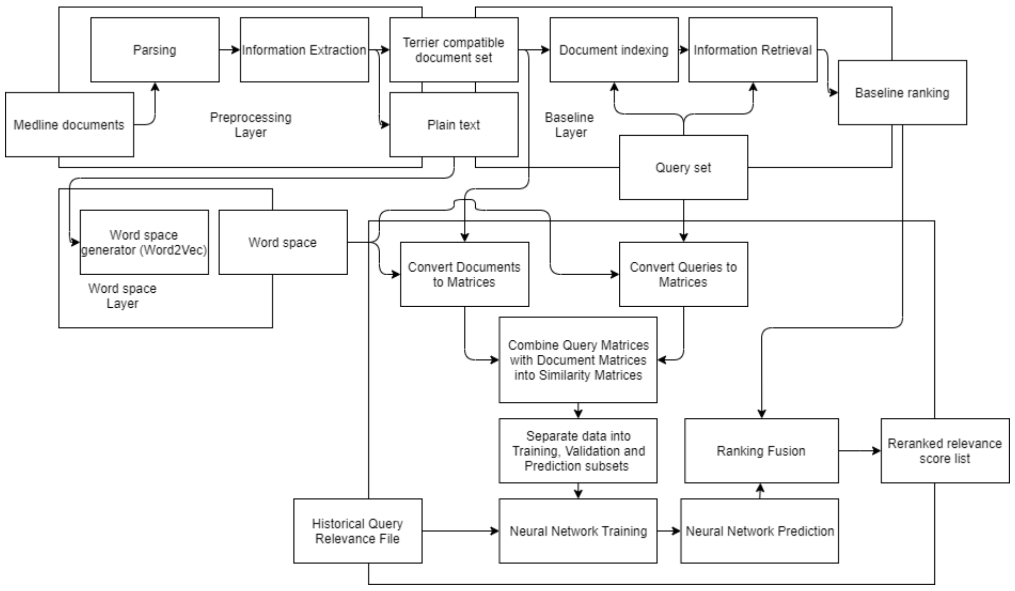

Parsing
Information Extraction
Terrier compatible document set
Document indexing
Information Retrieval
Baseline ranking
Medline documents
Preprocessing Layer
Plain text
Baseline Layer
Query set
Word space generator (Word2Vec)
Word space
Word space Layer
Convert Documents to Matrices
Convert Queries to Matrices
Combine Query Matrices with Document Matrices into Similarity Matrices
Separate data into Training, Validation and Prediction subsets
Ranking Fusion
Reranked relevance score list
Historical Query Relevance File
Neural Network Training
Neural Network Prediction

Fig. 1. General retrieval architecture used in this contribution.

2.1 Preprocessing Layer

Goal of this layer is to prepare the raw XML data for further processing. In this layer we extract essential data and use it to create corpora with a suitable structure. We use a dedicated XML processor. This software was implemented by our team for TREC-PM 2019.Source code for this layer can be obtained via a public repository[1]. It is a fast processor, implemented in C++ language. Processor is CPU based, engages all available system resources. The parser executes three tasks:

1. it reads the XML document and recognizes names of mark-up key,
2. it compares values of mark-up key names within a document with a list of informative mark-up names; it processes values of the fields within the list,
3. it extracts and tokenizes values of the processed fields.

The tokenizator is an integral part of the processor. The tokenizator executes the following rules:

1. special characters, punctuation, space and tabulator are treated as a separator,
2. new line is saved as a token,
3. all digits and numbers are converted into the same token.

A text field, after tokenization is ready for further processing. In this particular setup, we use the following mark-up keys:

– PMID,
– ArticleTitle,
– AbstractText,
– Keyword,
– NameOfSubstance,
– DescriptorName.

Based on the value of these fields, two types of documents are created. Flat-structured document, which is an input to the Baseline layer and Neural layer, and a plain text document, which is an input to the Word Space Layer.

2.2 Baseline Layer

Goal of this layer is to create a baseline ranking list. It uses structured data in order to create an index and performs the classical Information Retrieval process. We employ Terrier [10] tool in order to create baseline ranking. We have conducted several intermediate experiments and picked the DFR_BM25 model. We believe the experiments should be abstract in relation to the task, so we have chosen the model, which has been performing best on the Trec-Covid (CORD-19) dataset. Conducted experiments were in the Pseudo Relevance Feedback

[1] https://github.com/dudenzz/myindex

setting, with the parameters value of $d \in \{10, 30\}$ and $t \in \{100, 300, 1000\}$, where d is a number of documents for the pseudo relevance feedback, and t is a number of terms extending the query. Results of the experiment are presented in table 1. It should be noted, that classical model performance has relatively high variance related to task characteristics. In our previous works, LGD model in some cases performed better [1]. Model performance function, based on the task characteristics is a very interesting venue of research and requires further investigation.

PRF parameters: d = documents; t = terms						
	d=30			d=10		
model	t=1000	t=300	t=100	t=1000	t=300	t=100
DPH	0.1565	0.1602	0.1636	0.1521	0.1592	0.1580
BM25	0.1745	0.1743	0.1722	0.1724	0.1711	0.1734
PL2	0.1717	0.1716	0.1707	0.1592	0.1654	0.1624
DFR_BM25	**0.1815**	**0.1815**	**0.1819**	0.1802	0.1798	0.1782
BB2	0.1710	0.1719	0.1721	0.1750	1.752	0.1764
DLH	0.1642	0.1640	0.1616	0.1652	0.1640	0.1638
DLH13	0.1744	0.1770	0.1787	0.1704	0.1724	0.1731
DFRee	0.1441	0.1507	0.1464	0.1452	0.1462	0.1471
IFB2	0.1751	0.1756	0.1772	0.1747	0.1772	0.1791
In_expB2	0.1760	0.1762	0.1760	0.1758	0.1748	0.1732
In_expC2	0.1764	0.1766	0.1766	0.1762	0.1750	0.1740
InL2	0.1784	0.1787	0.1796	0.1774	0.1722	0.1712
LemurTF_IDF	0.1556	0.1558	0.1556	0.1556	0.1556	0.1558
LGD	0.1515	0.1570	0.1532	0.1535	0.1542	0.1552

Table 1. Terrier results for the Trec Covid benchmark.

2.3 Word Space Layer

This layer is used as a model for language artifacts representation. Thanks to this layer we can take documents, written in natural language and transform them into points or shapes in multidimensional spaces. Such representation of language artifacts works very well with machine learning techniques, which otherwise we wouldn't be able to use. We employ a classical Word2Vec[9] approach in order to create a word space model. This model is used as a dictionary in a process of creating document matrices. Word vector v_w is a multidimensional representation of a word w. Document matrix M is an organized list of word vectors.

$$M_D = \{v_w : \forall w \in D\}$$

We use L2 normalized version of the vector space, length of each vector within the space is one. In order to calculate similarity matrix S between two documents

D_1 and D_2, one just needs to multiply the document matrices.

$$S(D_1, D_2) = M_{D_1} M_{D_2}^T$$

We use the similarity matrices as an input to the Neural Layer.

Implementation of the Word Space Layer is done in Python3 language with use of the Gensim library [12].

2.4 Neural Layer

This is the final layer. Here we take processed documents, train the neural model and generate the final result. We follow recent works in the field of machine learning enhanced Information Retrieval to manage this layer. We look forward to implement the currently best performing Attention based Networks. Before use of attention networks [11] there have been few positive results of deep models on IR tasks, especially ad-hoc retrieval tasks. Generally, there are two ways for solving a matching problem between a query and a document or a passage. One is the representation-focused model, which tries to build a good representation for a single text with a deep neural network, and then conducts matching between the compositional text representations built on word embeddings. The other is the interaction-focused model, appropriate for IR as argued by [7], [8].which first builds local interactions (i.e., local matching correlations) between two pieces of text, and then uses deep neural networks to learn hierarchical interaction patterns for matching. It has been established that or IR the attention should be in the lowest layer. We roughly follow [5] and [4] for the system architecture.

Here we propose a simplified version of the Neural Layer. This layer uses Word Space component in order to create document matrices for both documents and queries. In order to simplify the computation, we limit the vocabulary V. With use of the word space, we find 150 most similar terms for each query word. Words, which are not within the vocabulary are skipped. Then we create the similarity matrices. Similarity matrices are limited to a size of 100 highest values for each similarity vector. Then each vector is fed into three layer neural network. First layer in the network performs Max Pooling. Second and Third layer comprise of Dense layers, first one uses ReLU as an activation function; second one employs softmax. Network objective is set as a categorical cross entropy between the network prediction and query-document relevance vector (relevance is expressed as a one-hot vector).

2.5 System Specification

We run the system on a machine with information retrieval and machine learning dedicated specification. For physical memory, we employ a disk array, which uses five disks directly connected to peripheral component interface with Non-Volatile Memory Express technology. We use this technology, to create a fast link between graphical processing unit, operating memory and physical memory. It is vital in information retrieval, as the textual corpora tend to get very

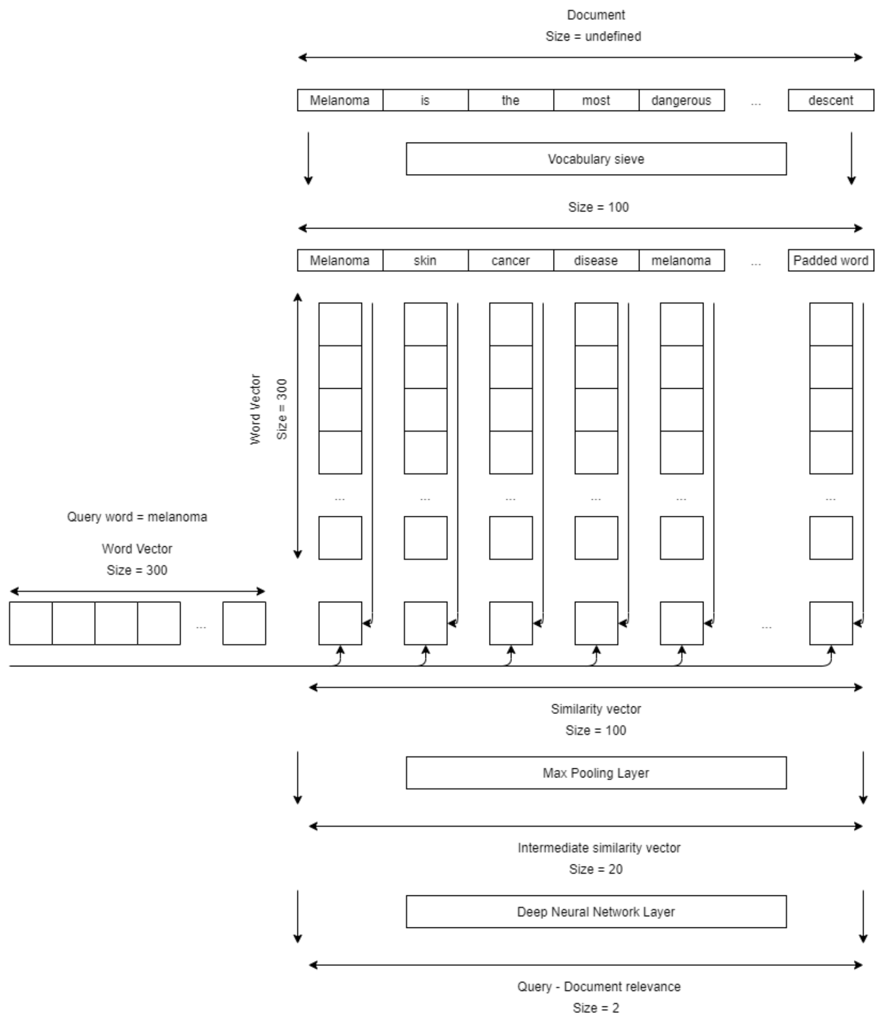


Fig. 2. Neural layer architecture used in this contribution.

large (e.g. the corpus used in Trec PM 2020 comprises of 220 GB of unprocessed data). We employ 126 GB of operational memory - it is convenient to store corpus indices directly in the operational memory for research purposes. Here, we use Elasticsearch tool for instant access to the documents within the corpus. Medium sized operational memory allows us to work on relatively large document and query matrices. We use nVIDIA P6000 graphics card for calculations - distributed, GPU based machine learning allows us to process 20000 samples per minute. We use classical driver-CUDA-CudNN-tensorflow-keras software stack. The intermediate layer, which multiplies Document matrices by Query matrices is still implemented in Python3 with numpy library and at this moment is our bottleneck. Currently we can process roughly 400 documents per minute in this layer. Considering the size of Medline, and other IR corpora, this part requires a redo. Our system is maintained on a Clear Linux machine with remote access.

3 Conducted Experiments

In this section, we describe the conducted experiments and provided results. We go over the evaluation of our approach and compare ourselves with the Median Trec Result. In this paper we discuss our two main runs - the baseline run and the reranked run. The baseline run uses our preprocessing architecture combined with the Terrier tool in order to create a ranking. We choose DFR_BM25 model in a Pseudo Relevance Feedback setting with 30 documents and 100 terms used for feedback. We use a simple concatenation of every field within the topic to create a query. We obtain a solid baseline, it outperforms median result in every available evaluation measure.

We use the remainder of our retrieval procedure to create a reranked list. Here, to create a query, we use only gene and disease fields from the topic. These fields are consistent for training data and data used in TREC PM 2020. We use concatenation of all document fields in the document matrix. Final score s_f is calculated as a sum of neural network prediction s_p and the baseline score s_b.

$$s_f = s_b - w \cdot s_p^{neg} + w \cdot s_p^{pos}$$

The weight parameter w is arbitrarily set to 3. It should be noted, that documents are heavily shortened in the process and the queries are fairly short. Thanks to this approach, we obtain relatively cheap method of training and using network predictions. We observe, that our reranking approach outperforms original ranking in general measures - infAP, infNDCG, bpref and R-prec. The reranked list however, performs worse in a head of the list, as it is shown by P@5 and P@10 metrics. A detailed list of evaluation values is presented in Tab. 2.

4 Conclusions

We describe experiments conducted within this contribution - we discuss used data and retrieval models. For the first time, we use the attention type network to

	infAP		infNDCG			P@5		P@10			bpref		R-prec		
topic	r	b	r	b	m	r	b	r	b	m	r	b	r	b	m
2	0,19	0,19	0,88	0,88	**0,89**	1,00	1,00	1,00	1,00	1,00	**0,80**	0,75	**0,66**	0,63	0,61
4	0,56	**0,60**	0,65	**0,67**	0,51	0,80	**1,00**	0,90	0,90	0,90	0,66	**0,70**	0,67	**0,73**	0,52
5	**0,01**	0,00	0,04	0,03	**0,08**	0,00	0,00	**0,10**	0,00	0,00	**0,02**	0,00	**0,09**	0,00	0,00
6	0,12	0,12	0,34	**0,35**	0,24	0,60	0,60	**0,40**	**0,40**	0,20	0,17	**0,18**	**0,20**	**0,20**	0,16
7	0,41	0,41	0,65	**0,67**	0,63	1,00	1,00	**1,00**	**1,00**	0,90	**0,74**	0,71	0,60	**0,66**	0,60
8	**0,39**	0,37	**0,67**	0,65	0,58	0,80	**1,00**	0,70	0,70	0,70	**0,48**	0,46	**0,51**	**0,51**	0,46
9	0,03	0,03	**0,12**	0,11	**0,12**	0,20	**0,60**	0,30	0,30	0,20	0,08	0,08	**0,11**	0,10	0,13
10	0,17	**0,18**	0,45	**0,47**	0,46	0,60	**0,80**	0,50	**0,70**	0,70	0,39	0,39	0,42	**0,44**	0,42
11	**0,64**	0,59	**0,80**	0,77	0,73	1,00	1,00	0,90	0,90	0,90	**0,78**	0,75	0,72	**0,73**	0,67
12	**0,61**	0,47	**0,72**	0,67	0,70	0,80	0,80	**0,90**	0,80	0,80	**0,80**	0,62	**0,74**	0,61	0,62
13	**0,42**	0,22	**0,60**	0,44	0,44	0,60	0,60	**0,70**	0,50	0,50	**0,62**	0,36	**0,61**	0,42	0,42
14	**0,52**	0,40	**0,73**	0,62	0,60	1,00	1,00	**1,00**	**1,00**	0,90	**0,82**	0,66	**0,72**	0,61	0,59
15	0,18	0,19	**0,87**	0,85	0,82	1,00	1,00	1,00	1,00	1,00	**0,81**	0,77	**0,66**	0,64	0,61
16	**0,33**	0,26	**0,87**	0,79	0,80	1,00	1,00	**1,00**	0,90	0,90	**0,80**	0,74	**0,66**	0,65	0,64
17	**0,21**	0,14	**0,82**	0,73	0,65	1,00	1,00	**1,00**	0,80	0,80	**0,78**	0,69	**0,59**	0,53	0,50
20	0,07	**0,11**	0,27	**0,40**	0,26	0,20	0,20	0,20	**0,40**	0,20	0,08	**0,16**	0,10	**0,20**	0,15
22	**0,05**	0,04	**0,23**	**0,21**	0,18	0,00	0,00	0,10	0,00	0,00	0,02	0,00	**0,14**	0,00	0,00
24	0,22	**0,29**	0,43	**0,53**	0,48	0,80	**1,00**	0,60	**0,70**	0,50	0,21	**0,29**	0,21	**0,32**	0,21
25	0,13	**0,13**	**0,40**	**0,40**	0,33	**0,40**	0,20	0,30	0,20	0,20	0,13	**0,14**	0,17	**0,20**	0,17
26	**0,38**	0,31	**0,56**	0,49	0,51	**0,80**	0,60	**0,70**	**0,70**	0,50	**0,46**	0,43	**0,50**	0,44	0,38
27	0,25	**0,36**	0,54	**0,60**	0,47	1,00	1,00	0,60	**0,90**	0,40	0,23	**0,35**	0,25	**0,36**	0,29
28	0,01	**0,07**	0,10	**0,23**	0,16	0,00	0,20	0,00	0,20	0,00	0,00	**0,11**	0,00	**0,17**	0,00
29	0,06	**0,17**	0,22	**0,35**	0,29	0,40	**0,80**	0,40	**0,60**	0,50	0,16	**0,27**	0,19	**0,30**	0,25
32	**0,14**	0,13	**0,35**	0,27	0,34	**0,20**	0,00	0,20	0,20	0,20	0,06	0,06	0,17	0,17	0,17
33	0,01	0,01	0,05	**0,06**	**0,06**	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
35	**0,16**	0,15	**0,43**	0,37	0,42	**0,60**	0,40	0,30	**0,40**	0,30	0,15	**0,19**	0,16	**0,21**	**0,21**
36	**0,16**	0,06	**0,46**	0,27	0,24	0,20	0,20	0,10	0,10	0,10	**0,25**	0,13	0,25	0,25	0,25
37	0,12	**0,16**	0,37	**0,46**	0,35	**0,40**	0,20	0,20	0,20	0,30	**0,14**	0,12	0,20	0,20	**0,30**
39	0,68	**0,63**	**0,68**	0,67	0,56	1,00	1,00	**0,60**	0,50	0,40	**0,64**	0,57	**0,60**	0,50	0,40
40	**0,33**	0,32	**0,66**	0,64	0,47	0,80	**1,00**	**0,50**	**0,50**	0,40	0,49	**0,50**	**0,50**	**0,50**	0,40
all	**0,24**	0,23	**0,48**	0,47	0,44	0,59	**0,62**	0,52	**0,53**	0,48	**0,38**	0,36	**0,37**	0,36	0,34

Table 2. Comparison of evaluation measures for our runs (r - reranked; b - baseline) with median run.

rerank the result. Overall, the reranking gives little but noticeable improvement. TREC PM 2020 contains a Treatment tag in queries, whereas this feature was missing in the TREC PM 2019 track.

Specifically, that our reranking approach outperforms original ranking in general measures - infAP, infNDCG, bpref and R-prec. The reranked list however, performs worse in a head of the list, as it is shown by P@5 and P@10 metrics.Our results are significantly better than the median. Due to a computing power limitations we adopted serious decrease of a size of a system.

Upon overcoming these limitations we expect some improvement of results.

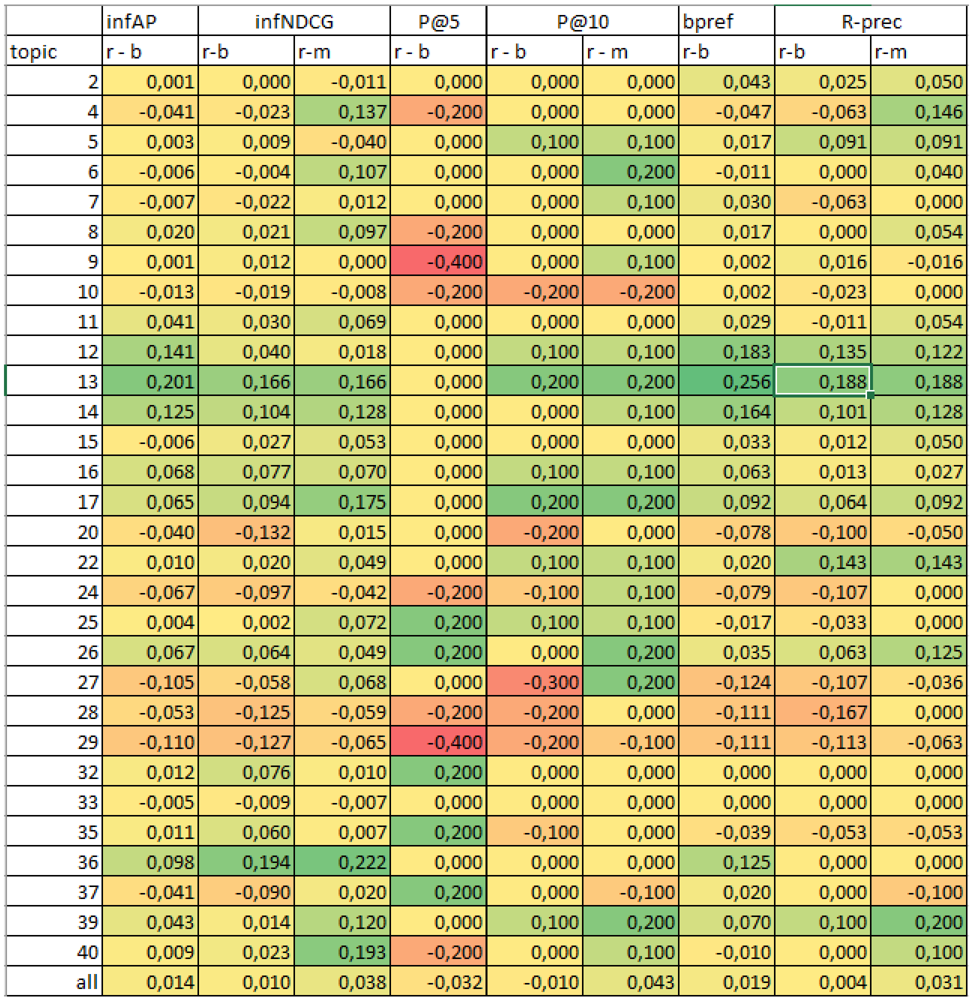
References

1. Cieslewicz, A., Dutkiewicz, J., Jedrzejek, C.: Baseline and extensions approach to information retrieval of complex medical data: Poznan's approach to the biocaddie 2016. Database **2018**, bax103 (2018). https://doi.org/10.1093/database/bax103, https://doi.org/10.1093/database/bax103
2. Cieslewicz, A., Dutkiewicz, J., Jedrzejek, C.: POZNAN contribution to TREC PM 2018 (notebook paper). In: Voorhees, E.M., Ellis, A. (eds.) Proceedings of the Twenty-Seventh Text REtrieval Conference, TREC 2018, Gaithersburg, Maryland, USA, November 14-16, 2018. NIST Special Publication, vol. 500-331. National Institute of Standards and Technology (NIST) (2018), https://trec.nist.gov/pubs/trec27/papers/Poznan-PM.pdf
3. Cieslewicz, A., Dutkiewicz, J., Jedrzejek, C.: Poznan contribution to TREC-PM 2019. In: Voorhees, E.M., Ellis, A. (eds.) Proceedings of the Twenty-Eighth Text REtrieval Conference, TREC 2019, Gaithersburg, Maryland, USA, November 13-15, 2019. NIST Special Publication, vol. 1250. National Institute of Standards and Technology (NIST) (2019), https://trec.nist.gov/pubs/trec28/papers/Poznan.PM.pdf
4. Dai, Z., Callan, J.: Deeper text understanding for IR with contextual neural language modeling. CoRR **abs/1905.09217** (2019), http://arxiv.org/abs/1905.09217
5. Dai, Z., Xiong, C., Callan, J., Liu, Z.: Convolutional neural networks for soft-matching n-grams in ad-hoc search. In: Chang, Y., Zhai, C., Liu, Y., Maarek, Y. (eds.) Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, WSDM 2018, Marina Del Rey, CA, USA, February 5-9, 2018. pp. 126–134. ACM (2018). https://doi.org/10.1145/3159652.3159659, https://doi.org/10.1145/3159652.3159659
6. Dutkiewicz, J., Jedrzejek, C., Frackowiak, M., Werda, P.: PUT contribution to TREC CDS 2016. In: Voorhees, E.M., Ellis, A. (eds.) Proceedings of The Twenty-Fifth Text REtrieval Conference, TREC 2016, Gaithersburg, Maryland, USA, November 15-18, 2016. NIST Special Publication, vol. 500-321. National Institute of Standards and Technology (NIST) (2016), http://trec.nist.gov/pubs/trec25/papers/IAII_PUT-CL.pdf
7. Guo, J., Fan, Y., Ai, Q., Croft, W.B.: A deep relevance matching model for ad-hoc retrieval. CoRR **abs/1711.08611** (2017), http://arxiv.org/abs/1711.08611
8. Guo, J., Fan, Y., Pang, L., Yang, L., Ai, Q., Zamani, H., Wu, C., Croft, W.B., Cheng, X.: A deep look into neural ranking models for information retrieval. Inf. Process. Manag. **57**(6), 102067 (2020). https://doi.org/10.1016/j.ipm.2019.102067, https://doi.org/10.1016/j.ipm.2019.102067

9. Le, Q.V., Mikolov, T.: Distributed representations of sentences and documents. In: Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014. JMLR Workshop and Conference Proceedings, vol. 32, pp. 1188–1196. JMLR.org (2014), http://proceedings.mlr.press/v32/le14.html
10. Macdonald, C., McCreadie, R., Santos, R.L., Ounis, I.: From puppy to maturity: Experiences in developing terrier. Proc. of OSIR at SIGIR pp. 60–63 (2012)
11. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: Guyon, I., von Luxburg, U., Bengio, S., Wallach, H.M., Fergus, R., Vishwanathan, S.V.N., Garnett, R. (eds.) Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA. pp. 5998–6008 (2017), http://papers.nips.cc/paper/7181-attention-is-all-you-need
12. Řehůřek, R., Sojka, P.: Software framework for topic modelling with large corpora. pp. 45–50 (05 2010). https://doi.org/10.13140/2.1.2393.1847

A Differences between Runs

In this appendix we highlight differences between our runs and Trec median with colours - in general green means that we achieved an improvement, red means that results are worse and yellow means that there is no significant difference.



	infAP	infNDCG		P@5	P@10		bpref	R-prec	
topic	r - b	r-b	r-m	r - b	r - b	r - m	r-b	r-b	r-m
2	0,001	0,000	-0,011	0,000	0,000	0,000	0,043	0,025	0,050
4	-0,041	-0,023	0,137	-0,200	0,000	0,000	-0,047	-0,063	0,146
5	0,003	0,009	-0,040	0,000	0,100	0,100	0,017	0,091	0,091
6	-0,006	-0,004	0,107	0,000	0,000	0,200	-0,011	0,000	0,040
7	-0,007	-0,022	0,012	0,000	0,000	0,100	0,030	-0,063	0,000
8	0,020	0,021	0,097	-0,200	0,000	0,000	0,017	0,000	0,054
9	0,001	0,012	0,000	-0,400	0,000	0,100	0,002	0,016	-0,016
10	-0,013	-0,019	-0,008	-0,200	-0,200	-0,200	0,002	-0,023	0,000
11	0,041	0,030	0,069	0,000	0,000	0,000	0,029	-0,011	0,054
12	0,141	0,040	0,018	0,000	0,100	0,100	0,183	0,135	0,122
13	0,201	0,166	0,166	0,000	0,200	0,200	0,256	0,188	0,188
14	0,125	0,104	0,128	0,000	0,000	0,100	0,164	0,101	0,128
15	-0,006	0,027	0,053	0,000	0,000	0,000	0,033	0,012	0,050
16	0,068	0,077	0,070	0,000	0,100	0,100	0,063	0,013	0,027
17	0,065	0,094	0,175	0,000	0,200	0,200	0,092	0,064	0,092
20	-0,040	-0,132	0,015	0,000	-0,200	0,000	-0,078	-0,100	-0,050
22	0,010	0,020	0,049	0,000	0,100	0,100	0,020	0,143	0,143
24	-0,067	-0,097	-0,042	-0,200	-0,100	0,100	-0,079	-0,107	0,000
25	0,004	0,002	0,072	0,200	0,100	0,100	-0,017	-0,033	0,000
26	0,067	0,064	0,049	0,200	0,000	0,200	0,035	0,063	0,125
27	-0,105	-0,058	0,068	0,000	-0,300	0,200	-0,124	-0,107	-0,036
28	-0,053	-0,125	-0,059	-0,200	-0,200	0,000	-0,111	-0,167	0,000
29	-0,110	-0,127	-0,065	-0,400	-0,200	-0,100	-0,111	-0,113	-0,063
32	0,012	0,076	0,010	0,200	0,000	0,000	0,000	0,000	0,000
33	-0,005	-0,009	-0,007	0,000	0,000	0,000	0,000	0,000	0,000
35	0,011	0,060	0,007	0,200	-0,100	0,000	-0,039	-0,053	-0,053
36	0,098	0,194	0,222	0,000	0,000	0,000	0,125	0,000	0,000
37	-0,041	-0,090	0,020	0,200	0,000	-0,100	0,020	0,000	-0,100
39	0,043	0,014	0,120	0,000	0,100	0,200	0,070	0,100	0,200
40	0,009	0,023	0,193	-0,200	0,000	0,100	-0,010	0,000	0,100
all	0,014	0,010	0,038	-0,032	-0,010	0,043	0,019	0,004	0,031

Fig. 3. This table shows the differences between runs and Trec median. r - reranked run; b - baseline run; m - Trec median.