Aschern at CheckThat! 2021: Lambda-Calculus of Fact-Checked Claims

Anton Chernyavskiy[1], Dmitry Ilvovsky[1] and Preslav Nakov[2]

[1]*HSE University, Moscow, Russia*
[2]*Qatar Computing Research Institute, HBKU, Doha, Qatar*

Abstract
We describe our system for the CLEF 2021 CheckThat! Lab Task 2 Subtask A on detecting previously fact-checked claims. We developed a pipeline using TF.IDF, sentence-BERT fine-tuned on the training data, and reranking using LambdaMART and the predicted similarity scores and positions in the ranked list as features. We examined the quality of each model on the validation set and analyzed its contribution to the final result using the trained LambdaMART. The official evaluation ranked our system 1^{st} by a wide margin over other participants and the organizers' baseline.

Keywords
fact-checking, lexical similarity, semantic similarity, sentence-BERT, TF.IDF, LambdaMART

1. Introduction

Social media provide an easy way to share information online. However, this also causes problems since some users may share false claims. Such claims are often sensational, which further contributes to their fast spread. One possible solution is to fact-check suspicious claims, but this is a difficult and time-consuming task when done manually. Even if the process is automated, it is impossible to fact-check every claim on the web. One could also ask: is it really necessary to fact-check everything? For example, if we aim to limit the spread of some false claim, then it is enough to fact-check only one post where it is present. Then, we can try to find posts that repeat that claim.

The CLEF 2021 CheckThat! Lab Task 2 [1] aims at solving that problem: given a tweet it asks to match it against a database of previously fact-checked claims. The participating systems are asked to rank the list of previously fact-checked claims according to their relevance, so that more useful ones are ranked higher. The task features two datasets for claims collected from tweets and from political debates, and it is offered in English and in Arabic. Below, we describe the system that we built for the English version of the dataset collected from tweets (Subtask 2A). At the core of our system is the sentence-BERT model [2], which was originally pre-trained on the Semantic Textual Similarity benchmark (STSb) data. We further fine-tuned it on the task data and then applied LambdaMART [3] to rerank the top-20 results. As features, LambdaMART uses the relevance scores and ranks predicted by sentence-BERT and TF.IDF.

CLEF 2021 – Conference and Labs of the Evaluation Forum, September 21–24, 2021, Bucharest, Romania
aschernyavskiy_1@edu.hse.ru (A. Chernyavskiy); dilvovsky@hse.ru (D. Ilvovsky); pnakov@hbku.edu.qa (P. Nakov)

2. Related Work

There are many studies that have addressed disinformation and misinformation [4, 5, 6, 7, 8, 9, 10]. However, there are only a few directly related to our task. In the ClaimBuster system [11], the problem was mentioned as part of the general fact-checking pipeline, but no evaluation of its solution was provided. The ClaimKG dataset was presented in [12], where claims from different fact-checking websites can be retrieved by some keywords using a knowledge graph. The original task formulation together with a dataset aimed to address the problem of detecting previously fact-checked claims were presented in [13], where the authors used data from Snopes and PolitiFact. They proposed a solution, which combined Elasticsearch, sentence-BERT, and reranking using RankSVM. Their dataset was then used within the framework of the CLEF 2020 CheckThat! Lab Task 2 [14]. Then, an expanded and cleaned-up dataset consisting of tweets was reused at the CLEF CheckThat! Lab 2021 Task 2A [1].

The winning team of the CLEF 2020 CheckThat! Lab Task 2 was Buster.AI [15], who proposed a solution based on RoBERTa, adversarial hard negative examples, and additional training on external data from FEVER, SciFact, and the Liar datasets. Team UNIPI-NLE [16] performed a cascade training of sentence-BERT models with the preliminary use of Elasticsearch to prune the list of possible candidates. Team UB_ET [17] applied DPH and LambdaMART over query-dependent features. Other participants also used Elasticsearch and sentence-BERT as well as Terrier, KD search, Universal Sentence Encoder (USE), TF.IDF, and BM25 to perform retrieval, and to compute similarity scores [14].

3. Dataset

We use the data presented within the CLEF CheckThat! Lab 2020 Task 2 Subtask A for English. The verified claims (`VerClaims`) database contains 13,825 claims. There are 1,000 positively labeled `<Claim, VerClaim>` pairs in the training set, and 200 input claims in the validation and the test sets. For each `VerClaim`, there is some additional information coming from the article that fact-checkers wrte about the claim: title, subtitle, author, and date of verification.

4. Evaluation Measures

The official evaluation measure is MAP@k for $k = 5$ (Mean Average Precision for the top-k `VerClaims` in the ranked list). Additional evaluation measures computed by the scoring script include MAP, MRR (Mean Reciprocal Rank), and P@k (precision for the top-k in the same range) for $k \in \{1, 3, 10, all\}$.

5. Method

Our pipeline is similar to that in [13], but we changed and improved its components. It is presented schematically in Figure 1. First, we independently calculate lexical and semantic similarity scores between the input `Claim` and each `VerClaim` using TF.IDF and sentence-BERT [2], respectively.

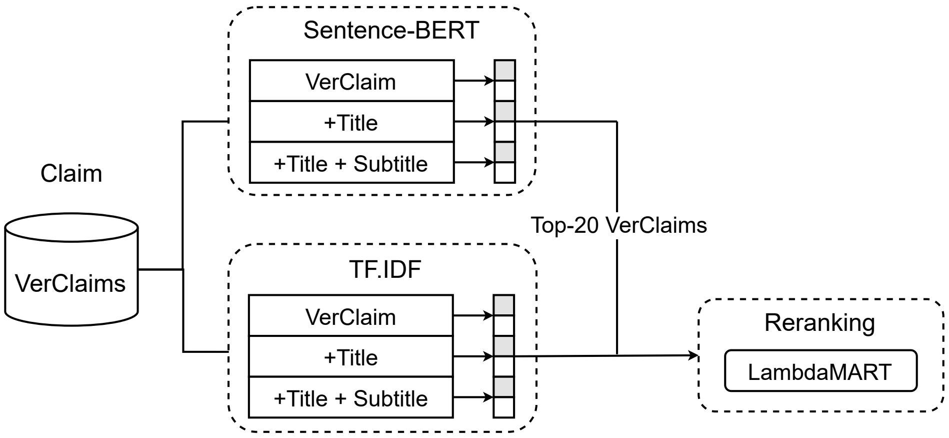

Figure 1: For the input `Claim`, TF.IDF and sentence-BERT independently evaluate the relevance of each `VerClaim` from the database, returning a similarity score and a position in a fully ranked list. The LambdaMART model then reranks the top-20 results from the sentence-BERT model using all predicted scores and positions as features.

We calculate each score for three possible input options: (*i*) `<Claim, VerClaim>` (*ii*) `<Claim + Title, VerClaim>`, (*iii*) `<Claim + Title + Subtitle, VerClaim>`. Here "+" denotes concatenation using `[SEP]` as a separator. Thus, we obtain six independent models. After that, we use LambdaMART [3] to re-rank the top-20 results selected by sentence-BERT trained on the input `<Claim + Title, VerClaim>`. Here, the features are predicted relevance scores and reciprocal ranks for each of the six models.

5.1. Lexical Similarity

To estimate the lexical similarity, we use TF.IDF, as a base model. Since TF.IDF depends on the number of words in the document/corpus, we tried to apply some data-specific pre-processing, e.g., clean up the input text by removing URLs, but this did not improve the results. Thus, our final lexical similarity approach converts the input to lowercase and computes embeddings accounting for the frequency of terms on a logarithmic scale $\mathrm{tf} = 1 + \log(\mathrm{tf})$. Then, we calculate the similarities of the input `Claim` and `VerClaims` as the cosine similarity between the corresponding embeddings. Finally, we use these scores in the re-ranker.

5.2. Semantic Similarity

Our TF.IDF approach relies on word matching. However, there are positive examples in the dataset where such word matching score would be very low, e.g., when comparing the `Claim` "*More Fake News. This was photoshopped, obviously, but the wind was strong and the hair looks good? Anything to demean!*" to the `VerClaim` "*The White House posted and then deleted an unflattering photograph of President Trump that displayed marked facial coloration.*" Thus, we also use sentence-BERT as an additional semantic similarity. This model is based on Siamese networks, where each component (BERT) independently computes embeddings for the `Claim` and for the `VerClaim`, and then the similarity between them is calculated using a cosine.

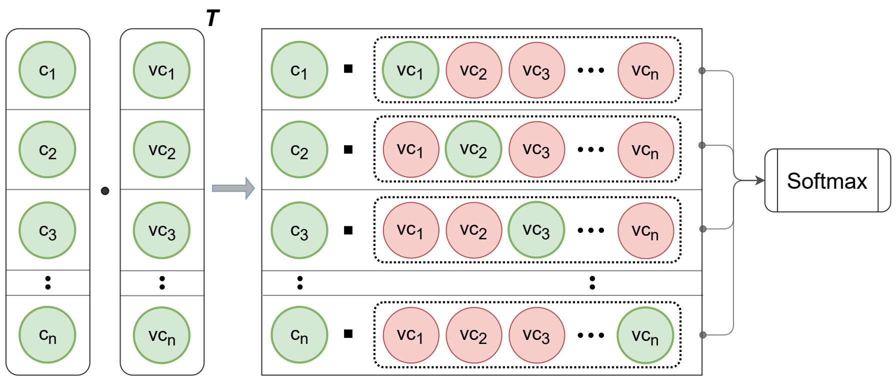

Figure 2: For the batch of positive pairs `<Claim, VerClaim>`, Mutiple Negatives Ranking loss contrasts the similarities between the input claim c_i and the relevant verified claim vc_i vs. between c_i and all other vc_j in the batch using softmax. ■ denotes the dot-product.

Since our task is an instance of the general task of determining the semantic similarity of two pieces of text, we fine-tune the model from the checkpoint that was trained on the STSb (Semantic Sentence Similarity benchmark).

Note, that using the sentence-BERT model to obtain sentence embeddings without any task-specific fine-tuning leads to the bad results for this task [13]. However, training with the MSE loss function is difficult due to the large class imbalance. Here, $MSE = \sum(y_i - \cos(f(\mathrm{c}), f(\mathrm{vc})))^2$, where f is the sentence-BERT encoder, y_i is the relevance score, and it equals 1 for positive `<Claim` (c), `VerClaim` (vc)> pairs, and 0 for negative ones. Note that there are many more negative pairs than positive ones. At the same time, if the triplets are composed of these pairs, then the problem of hard negative mining arises (the search for complex negative examples). Therefore, we apply Multiple Negatives Ranking (MNR) loss [18], which uses only positively marked pairs during training. To this end, it contrasts the similarities between the input `Claim` and the relevant `VerClaim` vs. `Claim` and all other `VerClaims` in the batch using softmax (Figure 2). This allows to simultaneously maximize the relevance score for the input positive pair and to minimize the scores for all other possible pairs in the batch.

It was proved, that the MNR loss function selects hard negatives by itself by using a temperature parameter in the softmax [19]. However, the model requires large batch sizes, since in order to find such an example in a batch, it must be present there. To overcome this limitation, we manually form the input training sequence at each epoch using the current model as follows. We choose an arbitrary `<Claim, VerClaim>` anchor pair from the training set (which contains only positive pairs). Then we select the top-k (k is a hyperparameter) of the closest `Claims` from the unused ones and we add them paired with the relevant `VerClaims` to the result sequence along with the anchor pair. The process ends when there are no unused `Claims` left.

We additionally make the MNR loss symmetric to be able to contrast to the positive pair all possible negative pairs: (`Claim`, `VerClaim`$_i$) and (`Claim`$_i$, `VerClaim`).

5.3. Reranking

At the reranking stage, we apply the LambdaMART model, which is based on Gradient Boosted Decision Trees. This is a learning-to-rank approach, which achieved the best results in different tasks, e.g., in the Yahoo! Learning to Rank Challenge (2011) [20]. To train the LambdaMART model, we use a 12-dimensional vector of features = 2 types of models * 3 types of input * 2 features (estimated relevance score and position in the ranked list of `VerClaims`).

To implement such a stacking approach, in order to prevent LabmdaMART from "peeping" into the labels encoded in the features, we use only the part of the training data that was not available when training sentence-BERT. In this part, for each claim, we select the top-50 candidates using a single model that achieved the best results on the validation set (it turned out to be the sentence-BERT model, trained on the input `<Claim + Title, VerClaim>`; see Section 7). Then, we supplement each of the resulting sets with the relevant `VerClaim`, if it was missing. Then, we train the model using all possible triplets that can be constructed in each set using the `Claim` as the anchor. At the inference stage, we only take the top-20 sentence-BERT results to minimize the final error. Note that we used LambdaMART, which can adjust the training procedure to optimize a specific evaluation measure (unlike RankSVM). To this end, the optimizer takes into account how much gain in the measure can be obtained by swapping two candidates from the triplet in the ranked list, while leaving the others untouched. In this case, we tuned the model to the main competition quality metric MAP@5.

6. Experimental Setup

6.1. Data Split

To train sentence-BERT, we took the first 800 claims from the training dataset, and we used the remaining 200 claims for validation. Then, out of those 200, we took 170 to train LambdaMART, and we validated its quality against the remaining 30 claims.

6.2. Parameter Settings

We used the Sentence-transformers framework[1] to train sentence-BERT models. We used the pre-trained `stsb-bert-base` for the input `<Claim, Verclaim>`, and `stsb-bert-large` for two other variants. We used the following hyperparameter values: learning rate of 1e-5, batch size of 6, training for 20 epochs, and the default optimizer with the number of warm up steps equal to 10% of the total number of training steps. For the MNR loss, we set the temperature to 0.05 and k to 7 to form the input sequence. We validated the model for each epoch, and we chose the best checkpoint. We used the LambdaMART implementation from the Python learning-to-rank toolkit,[2] and the following values for the hyperparameters: number of boosting stages of 1,500, maximum tree depth of 3, learning rate of 0.02, maximum leaf nodes of 12, fraction of queries to use for fitting the base learners of 0.3, fraction of features to use for selecting the best split of 0.3. We kept the best checkpoint as evaluated on the validation set.

[1]http://github.com/UKPLab/sentence-transformers
[2]http://github.com/jma127/pyltr

Table 1
Lexical model comparison on the development set.

Method	Input type	MAP@5	MAP@1	P@3	P@5
Elasticsearch	Claim	0.728	0.683	0.260	0.161
	Claim+Title	0.834	0.781	0.295	0.182
	Claim+Title+Subtitle	0.859	**0.822**	0.300	**0.184**
BM25 Okapi	Claim	0.414	0.352	0.159	0.105
	Claim+Title	0.586	0.528	0.214	0.137
	Claim+Title+Subtitle	0.646	0.608	0.230	0.140
TF.IDF	Claim	0.662	0.577	0.250	0.155
	Claim+Title	0.832	0.779	0.298	0.183
	Claim+Title+Subtitle	**0.861**	0.819	**0.305**	**0.184**

Table 2
Semantic model comparison on the development set.

Method	Input type	MAP@5	MAP@1	P@3	P@5
sentence-BERT	Claim	0.826	0.784	0.290	0.177
	Claim+Title	0.872	0.839	0.302	**0.185**
	Claim+Title+Subtitle	**0.882**	**0.849**	**0.307**	**0.185**

7. Experiments and Results

7.1. Lexical Similarity

A comparison of approaches to estimate the lexical similarity for each of the three input types is presented in Table 1. Here, we applied the source BM25 Okapi algorithm [21] in addition to Elasticsearch, where it is used to build the index. We found that our best TF.IDF approach, which used `Title` and `Subtitle` to calculate scores, outperformed BM25 and Elasticsearch on MAP@5. We also evaluated TF.IDF with the standard tf term calculation, but the results were worse. The results also show the importance of using the title as an additional input.

7.2. Semantic Similarity

The results on the official development set for sentence-BERT are presented in Table 2. Note that we used the `base` model for the input `<Claim>`, and the `large` variant in the other cases. The `base` model achieved a MAP@5 of 0.855 on the input `<Claim + Title, VerClaim>`. Therefore, the gain from the use of the `Title` is not as large as in the case of the lexical component. Although the best quality on the development set was achieved by the model trained on the input `<Claim + Title + Subtitle, VerClaim>`, we chose the one trained on `<Claim + Title, VerClaim>` as the core model, as it achieved MAP@5 of 0.772 vs. 0.739 on our validation sample. Moreover, the training data from which we took part for validation turned out to be much more complicated than the development set. Finally, the results for our best semantic model are better than those for our best lexical model.

Table 3
Results on the development set. Here, *shaar* is a baseline submission (Elasticsearch) by the organizers.

Rank	Team	MAP@5	MAP@1	RR	P@3	P@5
1	**aschern**	**0.941**	**0.932**	**0.940**	**0.318**	0.191
2	simihaylova	0.936	0.927	0.935	0.315	0.190
3	gs_chm	0.902	0.857	0.901	**0.318**	**0.192**
4	shaar	0.818	0.776	0.820	0.286	0.177

Table 4
Evaluation of the importance of 12 features produced by the pipeline components. It is estimated by the LambdaMART model. Each model provides two features: RR (Reciprocal Rank, that is the position in the ranked list) and Sim. score (the predicted similarity score).

Method	Input type	RR	Sim. score
TF.IDF	Claim	0.070	0.054
	Claim+Title	0.075	0.084
	Claim+Title+Subtitle	0.057	0.088
sentence-BERT	Claim	0.078	0.066
	Claim+Title	0.081	0.188
	Claim+Title+Subtitle	0.077	0.081

7.3. Reranking

Reranking with LambdaMART improved MAP@5 to 0.941 on the development set. The results for other participants are shown in Table 3.

We further estimated the importance of each of the 12 features using the trained LambdaMART model (Table 4). These results confirm that the most important features come from sentence-BERT (the semantic component), which used the claim with the title as an input. However, TF.IDF approaches (the lexical component) also have relatively high importance. Thus, we can conclude that the importance of the similarity score predicted by the TF.IDF approach on the input `<Claim + Title + Subtitle, VerClaim>` is higher than for the sentence-BERT base estimated on the same input. If we completely exclude the results of the lexical component from the features, MAP@5 on the development set drop to 0.899.

7.4. Official Results on the Test Set

The official evaluation results on the test set are presented in Table 5. We can see that our system outperforms the systems by the other participants and also the organizers' baseline by a large margin. The table also demonstrates the stability of our solution. Thus, the test performance coincides with what we observed on the validation set.

Table 5
Official results on the test set. shaar is a baseline submission (Elasticsearch) of the competition organizers.

Rank	Team	MAP@5	MAP@1	RR	P@3	P@5
1	**aschern**	**0.883**	**0.861**	**0.884**	**0.300**	**0.182**
2	NLytics	0.799	0.738	0.807	0.289	0.179
3	simihaylova	0.787	0.728	0.795	0.282	0.177
4	shaar	0.749	0.703	0.761	0.262	0.164

8. Conclusion and Future Work

We have described our system for the CLEF 2021 CheckThat! Lab Task 2 Subtask A English on detecting previously fact-checked claims. We developed a pipeline using TF.IDF, fine-tuned sentence-BERT, and reranking using LambdaMART, which used similarity scores and ranks as features. We examined the performance of each model on the validation set and analyzed its contribution to the final reranker. The official evaluation ranked our system 1^{st} by a wide margin ahead of other participants and the organizers' baseline.

In future work, we plan to experiment with other Transformer-based sentence encoders such as RoBERTa [22] and MPNet [23]. Another direction we want to explore is to use other potentially relevant data besides STSb for model pre-training.

Acknowledgments

Anton Chernyavskiy and Dmitry Ilvovsky performed this research in the framework of the HSE University Basic Research Program, funded by the Russian Academic Excellence Project *5-100*.

Preslav Nakov contributed as part of the Tanbih mega-project (tanbih.qcri.org), developed at the Qatar Computing Research Institute, HBKU, which aims to limit the impact of "fake news", propaganda, and media bias by making users aware of what they are reading, thus promoting media literacy and critical thinking.

References

[1] P. Nakov, G. Da San Martino, T. Elsayed, A. Barrón-Cedeño, R. Míguez, S. Shaar, F. Alam, F. Haouari, M. Hasanain, N. Babulkov, A. Nikolov, G. K. Shahi, J. M. Struß, T. Mandl, The CLEF-2021 CheckThat! lab on detecting check-worthy claims, previously fact-checked claims, and fake news, in: Proceedings of the 43rd European Conference on Information Retrieval, ECIR '21, Lucca, Italy, 2021, pp. 639–649.

[2] N. Reimers, I. Gurevych, Sentence-BERT: Sentence embeddings using Siamese BERT-networks, in: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP '19, Hong Kong, China, 2019, pp. 3982–3992.

[3] Q. Wu, C. Burges, K. Svore, J. Gao, Adapting boosting for information retrieval measures, Information Retrieval 13 (2009) 254–270.

[4] A. Zubiaga, A. Aker, K. Bontcheva, M. Liakata, R. Procter, Detection and resolution of rumours in social media: A survey, ACM Comput. Surv. 51 (2018).

[5] Y. Li, J. Gao, C. Meng, Q. Li, L. Su, B. Zhao, W. Fan, J. Han, A survey on truth discovery, SIGKDD Explor. Newsl. 17 (2016) 1–16.

[6] S. Vosoughi, D. Roy, S. Aral, The spread of true and false news online, Science 359 (2018) 1146–1151.

[7] D. Küçük, F. Can, Stance detection: A survey, ACM Comput. Surv. 53 (2020).

[8] G. Da San Martino, S. Cresci, A. Barrón-Cedeño, S. Yu, R. D. Pietro, P. Nakov, A survey on computational propaganda detection, in: Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-PRICAI '20, 2020, pp. 4826–4832.

[9] K. Popat, S. Mukherjee, J. Strötgen, G. Weikum, CredEye: A credibility lens for analyzing and explaining misinformation, in: Proceedings of the Web Conference, WWW '18, 2018, pp. 155–158.

[10] M. Hardalov, A. Arora, P. Nakov, I. Augenstein, A survey on stance detection for mis- and disinformation identification, 2021.

[11] N. Hassan, G. Zhang, F. Arslan, J. Caraballo, D. Jimenez, S. Gawsane, S. Hasan, M. Joseph, A. Kulkarni, A. K. Nayak, V. Sable, C. Li, M. Tremayne, ClaimBuster: The first-ever end-to-end fact-checking system, Proc. VLDB Endow. 10 (2017) 1945–1948.

[12] A. Tchechmedjiev, P. Fafalios, K. Boland, M. Gasquet, M. Zloch, B. Zapilko, S. Dietze, K. Todorov, ClaimsKG: A knowledge graph of fact-checked claims, in: Proceedings of the 18th International Semantic Web Conference, ISWC '19, Auckland, New Zealand, 2019, pp. 309–324.

[13] S. Shaar, N. Babulkov, G. Da San Martino, P. Nakov, That is a known lie: Detecting previously fact-checked claims, in: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL '20, 2020, pp. 3607–3618.

[14] A. Barrón-Cedeño, T. Elsayed, P. Nakov, G. D. S. Martino, M. Hasanain, R. Suwaileh, F. Haouari, N. Babulkov, B. Hamdan, A. Nikolov, S. Shaar, Z. S. Ali, Overview of CheckThat 2020: Automatic identification and verification of claims in social media, in: CLEF, 2020.

[15] M. Bouziane, H. Perrin, A. Cluzeau, J. Mardas, A. Sadeq, Team Buster.ai at CheckThat! 2020 insights and recommendations to improve fact-checking, in: CLEF, 2020.

[16] L. C. Passaro, A. Bondielli, A. Lenci, F. Marcelloni, UNIPI-NLE at CheckThat! 2020: Approaching fact checking from a sentence similarity perspective through the lens of transformers, in: CLEF, 2020.

[17] E. Thuma, N. Motlogelwa, T. Leburu-Dingalo, M. Mudongo, UB_ET at CheckThat! 2020: Exploring ad hoc retrieval approaches in verified claims retrieval, in: CLEF, 2020.

[18] M. Henderson, R. Al-Rfou, B. Strope, Y.-H. Sung, L. Lukács, R. Guo, S. Kumar, B. Miklos, R. Kurzweil, Efficient natural language response suggestion for smart reply, ArXiv 1705.00652 (2017).

[19] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, D. Krishnan, Supervised contrastive learning, arXiv 2004.11362 (2020).

[20] O. Chapelle, Y. Chang, Yahoo! learning to rank challenge overview., Journal of Machine Learning Research - Proceedings Track 14 (2011) 1–24.

[21] S. Robertson, H. Zaragoza, The probabilistic relevance framework: BM25 and beyond, Found. Trends Inf. Retr. 3 (2009) 333–389.

[22] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, V. Stoyanov, RoBERTa: A robustly optimized BERT pretraining approach, ArXiv 1907.11692 (2019).

[23] K. Song, X. Tan, T. Qin, J. Lu, T. Liu, MPNet: Masked and permuted pre-training for language understanding, arXiv 2004.09297 (2020).