HSRM-LAVIS at TREC 2020 Deep Learning Track: Neural First-stage Ranking Complementing Term-based Retrieval

Marco Wrzalik
marco.wrzalik@hs-rm.de
RheinMain University of Applied Sciences

Dirk Krechel
dirk.krechel@hs-rm.de
RheinMain University of Applied Sciences

ABSTRACT

This paper describes our submission to the passage ranking task of the TREC 2020 Deep Learning Track. With our work we focus on improving first-stage ranking and investigate its effect on end-to-end retrieval. Our approach aims to complement term-based retrieval methods with rankings from a representation-focused neural ranking model for first-stage ranking. Thus, we compile re-ranking candidates by mixing rankings from our complementary model with BM25 rankings. Our model incorporates ALBERT to exploit local term interactions and language modeling pretraining. For efficient retrieval, our passage representations are pre-computed and can be indexed in an Approximate Nearest Neighbor index. We investigate the characteristics of our approach based on the following three submitted runs: First, isolated rankings from our complementing first-stage ranking model for to reveal its standalone effectiveness. Second, mixed candidates from both BM25 and our model to inspect its complementary behavior. Third, rankings from an ELECTRA-based re-ranker using the candidates from the second run as an example of end-to-end results.

Table 1: Overview of our runs submitted to the passage retrieval task of the TREC 2020 Deep Learning Track.

Run-id	Description
`CoRT-standalone`	Standalone first-stage rankings from our complementary model, which will be refered to as "CoRT".
`CoRT-bm25`	Mixed rankings (50:50) from BM25 and our complementary model, which we refer to as "CoRT+BM25".
`CoRT-electra`	Exemplary end-to-end results using an ELECTRA-based re-ranker and `CoRT-bm25` as candidate passages. We refer to this run as "CoRT+BM25⟶ELECTRA".

1 INTRODUCTION

The current state of the art in passage retrieval is dominated by re-ranking models. They follow a two- or multi-stage ranking approach, where a number of candidate passages is re-ranked by a more sophisticated model. The candidates are typically retrieved by a sparse bag-of-words retrieval model such as BM25. Although BM25 has proven decent performance as a first-stage ranker, it tends to miss relevant passages. Recently, this has been counteracted with neural extensions to the sparse retrieval model such as query- and document expansion [15, 23, 25] or term weighting [6]. With our submission we investigate a first-stage ranking approach based on dense representations from a language model with the goal to complement term-based rankings. This has been pursued in the past: Boytsov et al. [2], for instance, generated dense document representations using static word embeddings to retrieve re-ranking candidates based on *k-nearest neighbor* search. In contrast, our model incorporates ALBERT [9], a language model and text encoder similar to BERT [7], which is used to exploit local term interactions and language modeling pretraining. We optimize our model towards representations that reflect relevance through vector similarity. At the same time, we guide our model towards a complementary behavior to BM25 by sampling negative examples from BM25 rankings. We hypothesize that using a BERT-like encoder is of great use for this task: Both language modeling pretraining and local interactions based on the attention mechanism of the incorporated transformer module [19] contribute to a better understanding of queries and passages. This differs to most previous *representation-focused* IR models based on language modeling approaches [2, 8, 13], since local term interactions were rarely exploited to form first-stage rankings. Many recent re-ranking models implicitly use the attention mechanism for query-passage interactions [11, 12, 14, 18]. In our approach, however, we intentionally dispense with query-passage interactions in favor of low computational effort to make retrieval from the full corpus feasible. At inference time, only the given query needs to be encoded, while the relevance score is computed as the dot product between query and passage representations. This can effectively be done using a GPU or through an *Approximate Nearest Neighbor* (ANN) index to avoid exhaustive scoring.

Preliminary to our submission to TREC 2020 DL, we used the pooled judgments from the passages task of TREC 2019 DL to evaluate our model [21]. There, we found that 26% of the top-10 candidates from our complementary model are not included in the pooled judgments. At the same time, our model performs well according to top-10 precision: Our standalone model achieves 54.2% P@10 while BM25, which is densely labeled among the top-10 rankings, only achieves 40.5% P@10. Therefore we want to contribute our rankings to the assessment pool of the this year's deep learning track. Also, we greatly appreciate the opportunity to obtain dense labels for the top-10 passages of our first-stage ranking experiments, which will be of great use for the analysis of our model. Beneath the first-stage ranking we also submit an exemplary end-to-end ranking pipeline: we conduct a re-ranking experiment using our complementing first-stage ranking approach. With this, we want to benchmark our approach against state-of-the-art models and demonstrate the effect of increasing candidate selection quality on end-to-end ranking performance.

2 TREC 2020 DEEP LEARNING TRACK

There are two tasks associated with the TREC 2020 Deep Learning Track: Document ranking and passage ranking. Both are based on the MS MARCO dataset [1]. For the passage task – on which we focus – about 530k positive connections between queries and passages are provided for training. Participants were asked to either re-rank a given set of passages or perform a full ranking using the MS MARCO corpus containing about 8.8M passages. 200 queries or topics were selected for the submissions, from which 81 entered the NIST assessment process and 54 are now contained by the final evaluation set. The assessments involve a pooling strategy in which first the top-10 passages from all submitted runs are judged by human assessors. Then, based on the positively labeled passages, additional candidates are chosen for assessment. Further information to the TREC Deep Learning Track and the NIST assessments can be found on the official website[1].

3 APPROACH

We describe a neural representation-focused ranking model for first-stage ranking that aims to act as a complementary ranker to existing term-based retrieval models such as BM25. To achieve this, we sample negative training examples from BM25 rankings and make use of local interactions and language modeling pretraining. The latter are introduced due to a BERT-like language model in the core of our model.

3.1 Architecture

The architecture and training strategy of our model, illustrated in Figure 1, follows the idea of a *Siamese Neural Network* [3]. Passages and queries are encoded using the identical model with shared weights except for one detail: The passage encoder ψ_α and the query encoder ψ_β use different segment embeddings [7]. Our model computes relevance scores as angular similarity between query and passage representations while training a *pair-wise* ranking objective

3.2 Encoding

Our model can incorporate any BERT-like encoder as underlying text encoder. Here, we use a pretrained ALBERT [9] encoder for its smaller model size, the tougher sentence coherence pretraining and increased first-stage ranking quality throughout our early-stage experiments compared to BERT. The tokenizer of ALBERT is a WordPiece tokenizer [22] including the special tokens [CLS] and [SEP] known from BERT. From the text encoder we seek a single representation vector for the whole passage or query, which we call *context representation*. From ALBERT we take the [CLS] embedding of the last layer for this purpose. The context representation obtained from the underlying encoder for an arbitrary string s is denoted with $\tau(s) \in \mathbb{R}^h$ where h is the output representation size.

ALBERT's language modeling approach involves sentence coherence prediction for which segment embeddings are used to signal different input segments. Although we only feed single segments to the encoder, i.e. a query or a passage, we use segment embeddings allowing the model to encode queries differently than passages. The segment embeddings E_A and E_B (illustrated in Figure 1) are part of

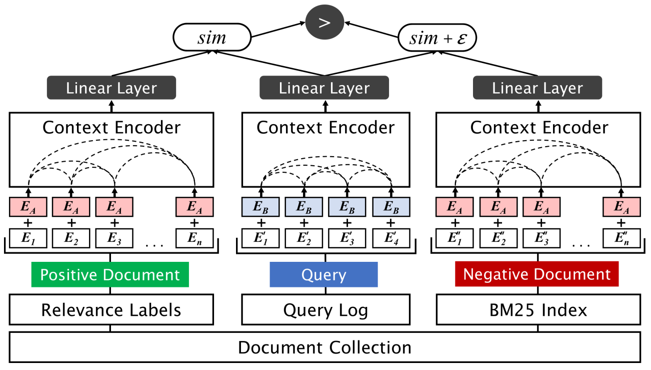


Figure 1: The model architecture of our complementary first-stage ranker and the pair-wise learning objective (simplified).

the context encoder functions τ_α and τ_β for passages and queries respectively. The context representation is further projected to the desired representation size e using a linear layer followed by a tanh activation function. Thus, the complete passage encoder function is $\psi_\alpha(s) := \tanh(W\tau_\alpha(s) + b)$ where $W \in \mathbb{R}^{h\times e}$ and $b \in \mathbb{R}^e$ are parameters of the linear layer. The query encoder ψ_β is defined analogous.

3.3 Training

Training our model corresponds to updating the parameters of the encoder ψ towards representations that reflect relevance between queries and documents through vector similarity. Each training sample is a triple comprising a query q, a positive passage d^+ and a negative passage d^-. While positive passages are taken from relevance assessments, negative passages are sampled from term-based rankings (i.e. BM25) to support the complementary property of our model. The relevance score for a query-passage pair (q, d) is calculated using the angular cosine similarity function.[2]

$$sim(q,d) := 1 - \arccos\left(\frac{\psi_\beta(q)\cdot\psi_\alpha(d)}{||\psi_\beta(q)||\;||\psi_\alpha(d)||}\right)/\pi \qquad (1)$$

As illustrated in Figure 1, the training objective is to score the positive example d^+ by at least the margin ϵ higher than the negative one d^-. We use the triplet margin loss function as part of our batch-wise loss function:

$$l(q,d^+,d^-) := max(0, sim(q,d^-) - sim(q,d^+) + \epsilon) \qquad (2)$$

Inspired by Oh Song et al. [16], we aim to take full advantage of the whole training batch. For each query, each passage in the batch is used as a negative example except for the positive one. Thus, the batch-wise loss function can be defined as follows:

$$\mathcal{L} := \sum_{1\le i\le n}\left(\sum_{1\le j\le n} l(q_i, d_i^+, d_j^-) + \sum_{1\le k\le n,\ k\ne i} l(q_i, d_i^+, d_k^+)\right) \qquad (3)$$

q_i, d_i^+ and d_i^- denote the triple of the i_{th} sample in the batch and n the number of samples per batch. We found this technique makes the training process more robust towards exploding gradients thus

[1]https://microsoft.github.io/TREC-2020-Deep-Learning/

[2]Similar to [4], we found angular similarity performs better than cosine similarity.

Table 2: Evaluation results of our first-stage passage ranking on the Deep Learning Track of TREC 2019 and 2020.

	TREC 2019 DL - Passage Task				TREC 2020 DL - Passage Task			
Method	**nDCG@1k**	**recall@100**	**recall@200**	**recall@1k**	**nDCG@1k**	**recall@100**	**recall@200**	**recall@1k**
BM25	0.6000	0.4976	0.5981	0.7450	0.5879	0.5669	0.6428	0.8031
CoRT	0.5129	0.4471	0.5172	0.6328	0.5413	0.5301	0.5850	0.6973
CoRT+BM25	**0.6636**	**0.5696**	**0.6590**	**0.8308**	**0.6630**	**0.6454**	**0.7230**	**0.8496**

the model can be trained without gradient clipping [26]. Also, it positively affects first-stage ranking results[3].

3.4 Indexing and Retrieval

For retrieval with our model, each passage must be encoded by the passage encoder ψ_α. Subsequent normalization of each vector allows us to use the dot product as a proxy score function, which is sufficient to accurately compile the ranking. Given a query q, we calculate its representation $\psi_\beta(q)$ and the dot product with each normalized passage vector. From those, the k highest scores are selected and sorted to form the ranking. This procedure can be implemented heavily parallelized using a GPU. Alternatively, the passage representations can be indexed in an ANN index to avoid exhaustive similarity search. Finally, we combine the resulting ranking of our model with the respective BM25 ranking by interleaving or *zipping* the positions beginning with our model until a new ranking of the same length has been arranged. During this process, each passage that was already added by the other ranking is omitted. For instance, merging two ranking lists beginning with $[a, b, c, d, \dots]$ and $[e, c, f, a, \dots]$ would result in $[a, e, b, c, \cancel{c}, f, d, \cancel{a}, \dots]$. The zipping procedure stops as soon as the desired ranking size has been reached. The result is a compound ranking of our model and BM25.

4 EXPERIMENTS

Our submitted runs comprise two experiment types: First-stage (full) ranking for candidate retrieval and subsequent re-ranking as an example for an end-to-end ranking pipeline. Since we could only submit three runs, we strategically decided to neither submit a BM25 baseline, nor a re-ranking baseline using BM25 candidates only. We argue, BM25 rankings should already be well covered in the pooled NIST assessments. Next to the evaluation results based on the TREC 2020 DL Passage Task, we also add corresponding results using the qrels from the previous year. Furthermore, we add evaluation results for baseline measures we did not submit. There, the evaluation results were generated using the official `trec_eval` evaluation script and the `"-l 2"` option to prevent the label "1" (for related) being counted as relevant.

4.1 First-stage Ranking

We trained our model as described in Section 3.3 while using a representation size of $e = 768$ to avoid a possible bottleneck. However, our previous experiments indicate that e can be reduced to 256 without significantly hurting the ranking quality, which substantially decreases computational effort and resource cost [21]. Any passage or query is cropped to a sequence length of 512 tokens. The rankings are compiled using exhaustive search on a GPU. Since the initial ordering of a certain set of candidates for re-ranking is not relevant to the final results, we identify *recall* to be the most adequate measure to summarize the quality of candidate selections. Various recall cuts are taken into account to represent situations, where smaller numbers of candidates are favorable. We report *nDCG@1000* as an additional summary of the overall first-stage ranking quality.

Results. As shown in Table 2, mixing candidates from our model with those from BM25 significantly increases recall at all considered cuts, which demonstrates the complementary behavior of our model. It is worth noting, that BM25 achieves much higher *recall@1*k on this year's judgments compared to last year. We hypothesize this translates to the significantly smaller gain we observe on this metric: While the judgments from 2019 show +0.086 *recall@1k* due to candidate mixing, the increase only amounts +0.046 *recall@1k* on this year's judgments. However, the increases on the other considered metrics are quite robust.

4.2 ELECTRA Re-ranking

As an example for an end-to-end ranking pipeline based on our candidates, we trained a neural re-ranker using a point-wise learning objective. Inspired by Nogueira and Cho [14], we fine-tune a pretrained language model on binary relevance classification. Instead of BERT, however, we use a pretrained ELECTRA discriminator [5]. An input query-passage pair (q, p) is concatenated to one token sequence with two segments. This sequence is processed by the ELECTRA discriminator while the embedding for the classification token in the last layer, which we denote with $\phi(q, p)$, is projected to a binary classification logit. We then apply the sigmoid activation function σ to obtain the relevance confidence for query q and passage p. This procedure can be formalized as $\zeta(q, p) = \sigma(W'\phi(q, p) + b')$ where $W' \in \mathbb{R}^{h\times 1}$ and $b' \in \mathbb{R}$ are the parameters of a linear layer with a single output activation. To form a ranking at inference time, we sort the candidates by the model's confidence. During training, we sample query-passage pairs, each associated with a binary relevance label $y \in \{0, 1\}$ and minimize the binary cross-entropy loss:

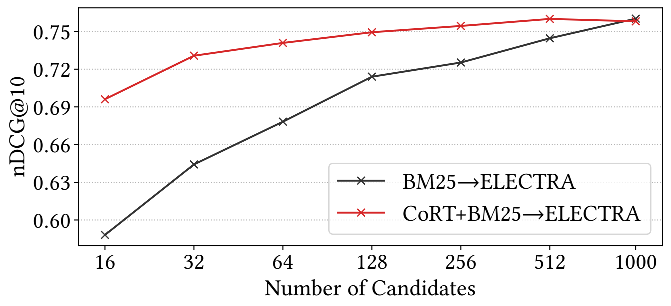
$$l'(q, p, y) = y \cdot log\ \zeta(q, p) + (1 - y) \cdot log\ (1 - \zeta(q, p)) \qquad (4)$$

Results. Table 3 shows the results of our exemplary end-to-end ranking pipeline. For the first stage of our pipeline we use 1000 candidates mixed from BM25 and our complementary model as described in Section 3.4. As a baseline, we conduct the same experiment using 1000 candidates from BM25 only. Decent gains in MAP and nDCG@1000 are achieved due to our complementary first-stage ranking. However, in this setting of 1000 candidates,

[3]We achieve 2.0 p.p. higher MRR@10 compared to the plain triplet margin loss on sparse labels from the MS MARCO *dev* dataset.

Table 3: End-to-end passage ranking results

	TREC 2019 DL - Passage Task				TREC 2020 DL - Passage Task			
Ranking Pipeline	**MRR**	**MAP**	**nDCG@10**	**nDCG@1k**	**MRR**	**MAP**	**nDCG@10**	**nDCG@1k**
BM25⟶ELECTRA	**0.9109**	0.4664	0.7306	0.6962	**0.8721**	0.5210	**0.7587**	0.7107
CoRT+BM25⟶ELECTRA	**0.9128**	**0.5101**	**0.7471**	**0.7336**	**0.8703**	**0.5399**	**0.7566**	**0.7438**



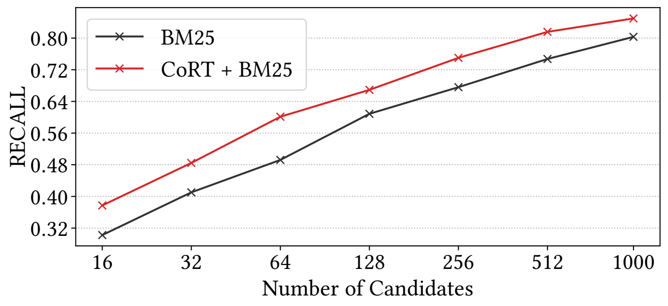


Figure 2: Based on the TREC 2020 DL Passage Task, ranking quality in terms of nDCG@10 vs number of candidates (left) and corresponding candidate recall (right).

where satisfactory numbers of relevant passages are already retrieved by BM25, further increasing recall seems not to translate into improved top-rank-focused metrics, i.e. MRR and nDCG@10.

4.3 Limiting Candidate Numbers

Since most re-ranking methods score each candidate individually, it is reasonable to assume that the corresponding computational cost relates linearly to the number of candidates. Thus, decreasing the candidates is an effective measure to reduce end-to-end ranking time and resource cost. However, this is likely to negatively influence ranking quality. We investigate the effect of limited numbers of candidates on the final ranking quality, while using candidates from a) BM25 and b) CoRT+BM25. As Figure 2 (left) illustrates, the ranking quality in terms of nDCG@10 of the run that uses candidates from CoRT+BM25 suffers much less from the decreasing number of candidates: 128 candidates from CoRT+BM25 result in a higher ranking quality than 512 candidates from BM25 only. This effect becomes heavier for lower candidate numbers: Only 32 candidates from CoRT+BM25 are needed outperform 256 BM25 candidates. However, this behavior is not reflected by the recall on the right side of Figure 2. There, the margin between the approaches is rather constant. Furthermore, we observe that recall is not a good predictor for end-to-end ranking quality with our state-of-the-art re-ranker. For instance, 32 candidates from CoRT+BM25 comprise an approximately equal recall value to 64 candidates from BM25. Contrarily, the resulting ranking quality in terms of nDCG@10 is much higher for the 32 candidates from CoRT+BM25. We hypothesize, it is not only the recall of the candidates that matters, but rather the coverage regarding different types of relevance signals.

5 CONCLUSION

We submitted three passage ranking runs to the TREC 2020 Deep Learning Track. Two of them are dedicated to our first-stage ranking approach, which aims to complement BM25 rankings with candidates from a representation-focused neural ranking model. This model incorporates ALBERT to exploit local interactions and language modeling pretraining. Our third run demonstrates exemplary end-to-end results using our candidate selection. We have shown that mixing BM25 ranking with those from our complementary model results in significantly increased recall. According to MAP and nDCG this translates into small increases to overall ranking quality after re-ranking. However, the nDCG@10 and MRR measures indicate, this does not apply to the top positions of our end-to-end rankings when using 1000 candidates, since the BM25 ranking already contains satisfactory numbers of relevant passages. However, this changes if less candidates are used during re-ranking. There, we find the top ranks are positively impacted by the increased recall our proposed first-stage ranker offers. Furthermore, we find this effect increases as the number of candidates decreases.

REFERENCES

[1] Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, et al. 2016. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268* (2016).
[2] Leonid Boytsov, David Novak, Yury Malkov, and Eric Nyberg. 2016. Off the beaten path: Let's replace term-based retrieval with k-nn search. In *Proceedings of the 25th ACM international on conference on information and knowledge management.* 1099–1108.
[3] Jane Bromley, JW Bentz, L Bottou, I Guyon, Y LeCun, C Moore, E Sackinger, and R Shah. 1993. Signature Verification using a "Siamese" Time Delay Neural Network. *Int.]. Pattern Recognit. Artzf Intell* 7 (1993).
[4] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. 2018. Universal sentence encoder. *arXiv preprint arXiv:1803.11175* (2018).
[5] Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. 2019. ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators. In *International Conference on Learning Representations.*

[6] Zhuyun Dai and Jamie Callan. [n.d.]. Context-Aware Passage Term Weighting For First Stage Retrieval. ([n. d.]).
[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).
[8] Christophe Van Gysel, Maarten De Rijke, and Evangelos Kanoulas. 2018. Neural vector spaces for unsupervised information retrieval. *ACM Transactions on Information Systems (TOIS)* 36, 4 (2018), 1–25.
[9] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942* (2019).
[10] Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101* (2017).
[11] Sean MacAvaney, Andrew Yates, Arman Cohan, and Nazli Goharian. 2019. CEDR: Contextualized embeddings for document ranking. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval.* 1101–1104.
[12] Bhaskar Mitra, Sebastian Hofstatter, Hamed Zamani, and Nick Craswell. 2020. Conformer-Kernel with Query Term Independence for Document Retrieval. *arXiv preprint arXiv:2007.10434* (2020).
[13] Bhaskar Mitra, Eric Nalisnick, Nick Craswell, and Rich Caruana. 2016. A Dual Embedding Space Model for Document Ranking. *arXiv* (2016), arXiv–1602.
[14] Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage Re-ranking with BERT. *arXiv preprint arXiv:1901.04085* (2019).
[15] Rodrigo Nogueira and Jimmy Lin. [n.d.]. From doc2query to docTTTTTquery. ([n. d.]).
[16] Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. 2016. Deep metric learning via lifted structured feature embedding. In *Proceedings of the IEEE conference on computer vision and pattern recognition.* 4004–4012.
[17] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.). Curran Associates, Inc., 8024–8035. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf
[18] Yifan Qiao, Chenyan Xiong, Zhenghao Liu, and Zhiyuan Liu. 2019. Understanding the Behaviors of BERT in Ranking. *arXiv preprint arXiv:1904.07531* (2019).
[19] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems.* 5998–6008.
[20] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. HuggingFace's Transformers: State-of-the-art Natural Language Processing. *ArXiv* abs/1910.03771 (2019).
[21] Marco Wrzalik and Dirk Krechel. 2020. CoRT: Complementary Rankings from Transformers. *arXiv preprint arXiv:2010.10252* (2020).
[22] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144* (2016).
[23] Ming Yan, Chenliang Li, Chen Wu, Bin Bi, Wei Wang, Jiangnan Xia, and Luo Si. 2019. IDST at TREC 2019 Deep Learning Track: Deep Cascade Ranking with Generation-based Document Expansion and Pre-trained Language Modeling. In *Proceedings of the Twenty-Eighth Text REtrieval Conference, TREC 2019, Gaithersburg, Maryland, USA, November 13-15, 2019 (NIST Special Publication, Vol. 1250)*, Ellen M. Voorhees and Angela Ellis (Eds.). National Institute of Standards and Technology (NIST). https://trec.nist.gov/pubs/trec28/papers/IDST.DL.pdf
[24] Peilin Yang, Hui Fang, and Jimmy Lin. 2017. Anserini: Enabling the use of Lucene for information retrieval research. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval.* 1253–1256.
[25] George Zerveas, Ruochen Zhang, Leila Kim, and Carsten Eickhoff. 2019. Brown University at TREC Deep Learning 2019. In *Proceedings of the Twenty-Eighth Text REtrieval Conference, TREC 2019, Gaithersburg, Maryland, USA, November 13-15, 2019 (NIST Special Publication, Vol. 1250)*, Ellen M. Voorhees and Angela Ellis (Eds.). National Institute of Standards and Technology (NIST). https://trec.nist.gov/pubs/trec28/papers/Brown.DL.pdf
[26] Jingzhao Zhang, Tianxing He, Suvrit Sra, and Ali Jadbabaie. 2019. Why Gradient Clipping Accelerates Training: A Theoretical Justification for Adaptivity. In *International Conference on Learning Representations.*

A TRAINING DETAILS

Software. All our implementations are made with *Python 3.7*. We used *PyTorch*[17] and *HuggingFace's Transformers*[20] as deep learning libraries. Any BM25 ranking were generated by the *Anserini* toolkit[24]. Anserini ensures reproducibility by providing optimized parameter sets and ranking scripts based on *Apache Lucene* for several datasets including MS MARCO.

First-stage Ranking. We trained our model based on the pretrained ALBERT model `"albert-base-v2"`, which is the lightest available version in *HuggingFace's* repository[4] [20]. Our model has been trained on MS MARCO for 10 epochs. Each epoch includes all queries that are associated to at least one relevant passage. For each query we randomly sample one positive and one negative passage. Most queries are only associated to one relevant passage, however. Negative examples are sampled from the corresponding top-100 BM25 ranking to support the complementary property of our model. There, we filter any positively labeled passage as well as any passage above rank $n = 8$ for their high probability of actually being relevant. With this we aim to reduce the number of false negatives and thus generated contradictory signals. Due to high computational effort, this parameter was not tuned systematically. However, we achieved 0.7 p.p. higher MRR@10 and 1.2 p.p. higher RECALL@100 on our sparse validation set when training with $n = 8$ compared to $n = 0$. As usual for BERT-based models we use the ADAM optimizer with weight decay fix [10] and the default parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$, $eps = 10^{-6}$. We have empirically chosen a weight decay rate of $\lambda = 0.1$ and a linearly decreasing learning rate schedule starting with $lr = 2\times10^{-6}$ after 20.000 warm-up batches. We train mini-batches of size $b = 6$ samples (triples) while accumulating the gradients of 100 mini-batches before performing one update step. The triplet margin (eq. 2 in Section 3.3) has been set to $\epsilon = 0.1$, which has been coarsely tuned in the range of $[0.01, 0.2]$.

Re-ranker. The pretrained ELECTRA discriminator [5] we used for our re-ranking experiment was accessed through *HuggingFace*'s repository [20], namely `google/electra-large-discriminator`. The optimizer settings have been adopted from our first-stage ranking experiment except for the learning rate, which we empirically set to 5×10^{-5}. We trained the model for 8 epochs on the MS MARCO training set, with batches of $b = 6$ samples and 100 steps of gradient accumulation. The negative examples for the point-wise learning objective has been taken from our top-1000 mixed candidates (`CoRT-bm25`).

[4]https://huggingface.co/transformers/pretrained_models.html