RankT5: Fine-Tuning T5 for Text Ranking with Ranking Losses

Honglei Zhuang
hlz@google.com
Google Research

Zhen Qin
zhenqin@google.com
Google Research

Rolf Jagerman
jagerman@google.com
Google Research

Kai Hui
kaihuibj@google.com
Google Research

Ji Ma
maji@google.com
Google Research

Jing Lu
ljwinnie@google.com
Google Research

Jianmo Ni
jianmon@google.com
Google Research

Xuanhui Wang
xuanhui@google.com
Google Research

Michael Bendersky
bemike@google.com
Google Research

ABSTRACT

Pretrained language models such as BERT have been shown to be exceptionally effective for text ranking. However, there are limited studies on how to leverage more powerful sequence-to-sequence models such as T5. Existing attempts usually formulate text ranking as a classification problem and rely on postprocessing to obtain a ranked list. In this paper, we propose RankT5 and study two T5-based ranking model structures, an encoder-decoder and an encoder-only one, so that they not only can directly output ranking scores for each query-document pair, but also can be fine-tuned with "pairwise" or "listwise" ranking losses to optimize ranking performance. Our experiments show that the proposed models with ranking losses can achieve substantial ranking performance gains on different public text ranking data sets. Moreover, ranking models fine-tuned with listwise ranking losses have better zero-shot ranking performance on out-of-domain data than models fine-tuned with classification losses.

CCS CONCEPTS

• **Information systems → Learning to rank**.

KEYWORDS

T5, text ranking, ranking losses

ACM Reference Format:
Honglei Zhuang, Zhen Qin, Rolf Jagerman, Kai Hui, Ji Ma, Jing Lu, Jianmo Ni, Xuanhui Wang, and Michael Bendersky. 2023. RankT5: Fine-Tuning T5 for Text Ranking with Ranking Losses. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '23), July 23–27, 2023, Taipei, Taiwan.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3539618.3592047

1 INTRODUCTION

Text ranking is a fundamental component of countless real world applications such as search and question answering. Progress on pretrained language models in the past few years [7] and the release of large-scale public data sets [1, 20] enable a series of work [12, 23, 33] on text ranking models which directly encode textual query and document using pretrained language models, noticeably BERT [7]. Recently, large language models such as T5 [37], GPT-3 [2] and InstructGPT [34] have shown superior performance in various NLP tasks including sentiment analysis, coreference resolution, and translation. Such models often have much larger size available than previous models such as BERT [7] to store more hidden knowledge. They also mostly have a sequence-to-sequence interface to unify different NLP tasks from classification to text generation.

While BERT-based models have been well explored for text ranking [12, 23, 33], how to leverage T5 for text ranking is still under-explored and challenging. First, while many classification and text generation tasks fit into the sequence-to-sequence framework, text ranking tasks are more difficult: a text ranking model is often expected to output a numerical ranking score $\hat{y} \in \mathbb{R}$ for each query-document pair. Second, it is important to train a text ranking model with ranking losses [14, 25, 36] to optimize its ranking performance, where the losses take into account the ranking scores from multiple documents for each query. This is different from the typical T5 fine-tuning strategy where the objective is often formulated into a text generation loss for each single input sequence independently.

A typical approach to use T5 for text ranking is to convert the problem into a token generation problem. For example, Nogueira et al. [32] fine-tune the T5 model to predict a "`true`" or "`false`" token for a relevant or irrelevant query-document pair and then use a postprocessing step during inference to derive ranking scores to rank candidate documents. Such an approach can be considered as a "pointwise" classification formulation. How to extend this approach to fine-tune T5 with ranking losses is not well-explored.

In this paper, we propose RankT5 with the goal to support text ranking more natively with T5 by outputting ranking scores, instead of text tokens. We first adapt the encoder-decoder structure for this goal. We also propose an encoder-only structure which omits the T5 decoder. These two structure variants allow us to fine-tune T5 with various ranking losses to directly optimize ranking performance.

Experiments on MS MARCO and Natural Question (NQ) data sets show that our RankT5 models fine-tuned with specialized ranking losses can significantly outperform other T5 ranking models fine-tuned with classification losses and previously proposed T5 adaptations for ranking [32]. We also discover that models fine-tuned with some ranking losses tend to have better zero-shot performance than models fine-tuned with classification losses. Checkpoints of RankT5 fine-tuned with ranking losses are released publicly (Section 5).

2 RELATED WORK

Model structure. A typical ranking model structure design is the cross-attention model structure, where a query and a candidate document is concatenated into a sequence and fed into the model. This model structure has been explored for BERT-like encoder-only

SIGIR '23, July 23–27, 2023, Taipei, Taiwan
© 2023 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-9408-6/23/07.
https://doi.org/10.1145/3539618.3592047

model [10, 12, 33] and T5-like model [17, 32], but the model is not directly fine-tuned with ranking losses for the optimal ranking performance. There are also models taking multiple documents as input [29, 35, 48], but they are usually applied in a *late ranking* stage and are complementary to our work. Some other models rank documents by the likelihood of query generated from language models given the document [8, 17, 40, 49, 50].

Notice that our focus is not on the *retrieval* task [9, 13, 18, 19, 24, 28, 41], where the model needs to score hundreds of millions of documents in the entire corpus almost instantly for each query.

Fine-tuning with ranking losses. Early explorations [32, 33] of applying pretrained language models on the document reranking task mainly use "pointwise" losses, where the loss is calculated for each query-document pair independently. There are some recent works that use a "listwise" loss [10, 12, 27, 38], but they only fine-tune BERT, RoBERTa, ERNIE, etc. There is no existing work fine-tuning sequence-to-sequence models like T5 with *ranking losses*. Others fine-tune retrieval models with pairwise or softmax loss [18, 26, 47], while we focus on reranking models in this work.

There are also several studies of pretraining methods tailored to enhance text ranking [11, 30, 44–46]. While we only focus on fine-tuning, our proposed method can be seamlessly applied.

3 PRELIMINARIES

Problem definition. We provide the formalized definition of a ranking task. For each query q_i, a list of m candidate documents $D_i = (d_{i1}, \ldots, d_{im})$ are provided, which are usually the output from a retriever. The relevance labels of candidate documents with regard to the query are represented as $\mathbf{y}_i = (y_{i1}, \ldots, y_{im})$ where $y_{ij} \geq 0$.

The objective is to train a ranking model f which takes a query-document pair as input and outputs a ranking score $\hat{y}_{ij} = f(q_i, d_{ij}) \in \mathbb{R}$. We aim to optimize the ranking metrics after we sort the documents in D_i for each query q_i based on their ranking scores.

T5. T5 [37] is a text-to-text pretrained generative language model with an encoder-decoder structure. It takes a piece of text as input and outputs a sequence of text tokens in an autoregressive manner.

More formally, we denote the input to the T5 encoder as a text sequence $s = [w_1, \ldots, w_l]$ and the previously generated tokens from the T5 decoder as $t_{1:k-1}$ during the autoregressive decoding process. We formalize the T5 model structure as:

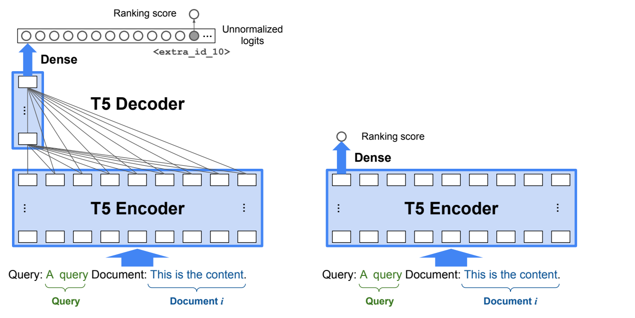
$$\mathbf{p}_k = \text{T5}(s, t_{1:k-1}) = \text{Softmax}(\text{Dense}(\text{Dec}(\text{Enc}(s), t_{1:k-1})))$$

where the output is a vector with the length of the vocabulary size $\mathbf{p}_k \in \mathbb{R}^{|\mathcal{V}|}$, representing the predictive probability of each token in the vocabulary being generated at the k-th position; $\text{Enc}(\cdot)$ and $\text{Dec}(\cdot, \cdot)$ are the encoder and the decoder of T5 respectively; $\text{Dense}(\cdot)$ is a dense layer; $\text{Softmax}(\cdot)$ is a softmax transformation layer that normalizes the vector into a probability distribution.

4 RANKT5 MODELING

4.1 Model structure

We propose to directly obtain the numerical ranking score as the model output, so that the model can be directly fine-tuned by ranking losses to optimize ranking metrics. We present two variants based on T5: an encoder-decoder and an encoder-only model.



(a) RankT5 Encoder-Decoder (b) RankT5 Encoder-Only

Figure 1: Model structures of the two variants of RankT5.

Input sequence. For each candidate document d_{ij} and its query q_i, we concatenate them with prefix "Document:" and "Query:" respectively to construct the input text s_{ij}:

$$s_{ij} = \text{Query: } q_i \text{ Document: } d_{ij}$$

The construction of input sequence is similar to Nogueira et al. [32] except that we do not include the "Relevant:" postfix. The postfix does not affect the results in our experiments.

Encoder-decoder (EncDec). This model variant is a simple adaptation of the T5 model by using the first output token of the decoder. In this variant, we feed the input into a T5 model and obtain the unnormalized logits $\mathbf{z}$ over the entire vocabulary:

$$\mathbf{z} = \text{Dense}(\text{Dec}(\text{Enc}(s_{ij}))) \tag{1}$$

Notice that we omit the softmax layer over vocabulary so that the elements in $\mathbf{z}$ can be arbitrary real numbers. We also do not need previous tokens $t_{1:k-1}$ since we do not generate tokens.

We specify an unused token in T5 vocabulary "`<extra_id_10>`" and take its corresponding unnormalized logits as the ranking score:

$$\hat{y}_{ij} = \mathbf{z}_{(\texttt{<extra\_id\_10>})} \tag{2}$$

where we use the notation $\mathbf{z}_{(w)}$ to represent the logits corresponding to the token $w \in \mathcal{V}$. The special token can be any other unused token in the vocabulary. An illustration of this model structure can be found in Figure 1(a).

Encoder-only (Enc). We also propose an encoder-only variant since we do not need to perform autoregressive decoding. The input text s_{ij} remains the same as the encoder-decoder model. We take the output of the encoder $\text{Enc}(s_{ij})$, which is a sequence of embedding vectors $[\mathbf{e}_1, \cdots, \mathbf{e}_l]$, and apply a pooling layer $\text{Pool}(\cdot)$ to aggregate them into a single embedding vector. Then we apply the dense layer which directly projects the embedding vector to the ranking score $\hat{y}_{ij}$. More formally,

$$\hat{y}_{ij} = \text{Dense}(\text{Pool}(\text{Enc}(s_{ij}))) \tag{3}$$

Figure 1(b) summarizes the proposed model structure.

4.2 Training

For each query q_i and a list of its candidate documents D_i, we obtain the list of predicted ranking scores $\hat{\mathbf{y}}_i$ by applying the model on each query-document pair (q_i, d_{ij}) where $d_{ij} \in D_i$. Then we can train the model by optimizing a ranking-based training loss function $\ell(\mathbf{y}_i, \hat{\mathbf{y}}_i)$, defined on the two lists of predicted scores $\hat{\mathbf{y}}_i$ and relevance labels $\mathbf{y}_i$. We study the following ranking losses:

Pairwise logistic (Pair). We can train the model using a pairwise logistic ranking loss [4]:

$$\ell_{\text{Pair}}(\mathbf{y}_i, \hat{\mathbf{y}}_i) = \sum_{j=1}^{m} \sum_{j'=1}^{m} \mathbb{I}_{y_{ij} > y_{ij'}} \log\left(1 + e^{\hat{y}_{ij'} - \hat{y}_{ij}}\right)$$

where the ranking problem is converted into a binary classification problem on the order of each candidate document pair in the same list with different relevance labels.

Listwise softmax cross entropy (Softmax). We can also define a listwise softmax cross entropy loss [3], which is a simple version of ListNet [5]. It takes the entire list into account.

$$\ell_{\text{Softmax}}(\mathbf{y}_i, \hat{\mathbf{y}}_i) = -\sum_{j=1}^{m} y_{ij} \log\left(\frac{e^{\hat{y}_{ij}}}{\sum_{j'} e^{\hat{y}_{ij'}}}\right)$$

Listwise poly-1 softmax cross entropy (Poly1). We also try a recently proposed extended version of the softmax cross-entropy loss called PolyLoss [22]. The idea is to adjust the weights of polynomial terms in the Taylor expansion of a softmax cross entropy loss. A simplified version only adjusted the first polynomial term:

$$\ell_{\text{Poly1}}(\mathbf{y}_i, \hat{\mathbf{y}}_i) = \ell_{\text{Softmax}}(\mathbf{y}_i, \hat{\mathbf{y}}_i) + \sum_{j=1}^{m} \epsilon \cdot y_{ij'}\left(1 - \frac{e^{\hat{y}_{ij}}}{\sum_{j'} e^{\hat{y}_{ij'}}}\right)$$

where ϵ is a parameter specified by users, representing how much extra coefficient to be placed for the first polynomial term.

5 EXPERIMENT SETUP

5.1 Data sets

MS MARCO. We use the MS MARCO passage ranking data set [1]. The data set contains around 530,000 queries in the "train" partition and around 6,800 queries in the "dev" partition. The candidates are from a corpus with more than 8.8 million passages. For each query, relevant passages are labeled as 1, and others are labeled as 0. We use a dual-encoder retriever [31] fine-tuned on MS MARCO to retrieve the top-1000 passages for each query in both the "train" and "dev" partitions as the candidate documents.

Notice that in this paper, the term "document" is used interchangeably with "passage". We do *not* focus on long document ranking tasks such as the MS MARCO document ranking task.

Natural Questions (NQ). We use the Natural Questions data set [20] with more than 50,000 queries in the "train" partition and 8,000 in the "dev" partition. We adopt the preprocessing setup similar to Karpukhin et al. [18] to construct the corpus of passages. Similar to MS MARCO, binary relevance labels are provided. We use a dual-encoder retriever [26] fine-tuned on NQ to retrieve the top-1000 passages for each query.

Training data construction. We construct the training data by first selecting a document with label 1 for each query and then uniformly randomly sampling $(m - 1)$ documents from the top-1000 retrieved documents with label 0. We set the list size m to 36 in our experiments due to hardware constraints.

For models with pointwise training losses, we upsample documents with label 1 in each query to the same number as documents with label 0 in order to achieve the optimal performance.

Evaluation. We evaluate the performance on the "dev" partition on both data sets. We perform model inference on the top-1000 documents retrieved by the dual-encoder retriever of each data set respectively. We evaluate the performance by Mean Reciprocal Rank (MRR@10) [43], Normalized Discounted Cumulative Gain (NDCG@5, 10) [16] and Mean Average Precision (MAP).

5.2 Parameter configurations

We initialize the ranking model with pretrained T5-Large checkpoint if not specified otherwise. For the pooling layer in RankT5-Enc, we follow BERT [7] and take the embedding vector of the first token. The results do not differ when using other pooling methods like mean pooling in our experiments.

We set the maximum input sequence length to 128 and 128+256 = 384 for MS MARCO and NQ respectively. We do not find significant performance degradation compared to using 512 as the sequence length. The batch size is set to 32 lists per batch for both data sets. We use a constant learning rate of 1×10^{-4} during fine-tuning. For the MS MARCO data set, we fine-tune our models for 50,000 steps. For the NQ data set, we fine-tune most of our models for 100,000 steps, except the ones using pointwise cross-entropy loss, which achieves the best performance at 25,000 steps. For Poly1 loss, we simply set $\epsilon = 1$ for all of our experiments.

The model implementation is based on T5X[1]. All the ranking losses are implemented in Rax[2] [15]. Selected checkpoints of fine-tuned RankT5 are released publicly[3].

6 RESULTS

Overall comparison. We compare the performance of our proposed rankers with different model structures and different training losses. We train the monoT5 model [32] and BERT [12] models on our data sets and report their performance as baselines. The BERT models are initialized from BERT-Large checkpoints and fine-tuned with pointwise cross-entropy classification loss (PointCE) and the listwise softmax cross-entropy ranking loss (Softmax) respectively. We also include results where our proposed RankT5 models are fine-tuned with the pointwise cross-entropy classification loss. All results are presented in Table 1.

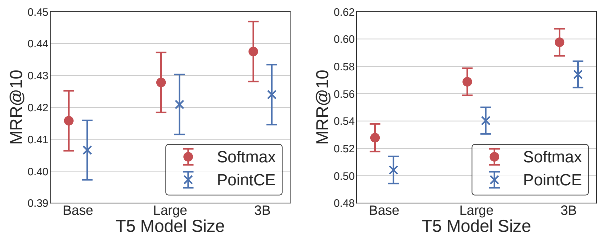
We observe that fine-tuning with ranking losses substantially helps RankT5 improve the ranking performance. RankT5 with Softmax and Poly1 consistently outperform other baselines, including RankT5 with the classification loss (PointCE). On both data sets, the best performing RankT5 improves around +2% in all metrics compared to monoT5.

We also verify that the initial checkpoint of T5 shows advantage over the initial checkpoint of BERT. RankT5-Enc and BERT have similar model size and structure but different pretrained checkpoints. When fine-tuned with the Softmax loss, RankT5-Enc outperforms BERT with a large margin on both data sets (+3.7% to +4.7% in MRR@10). This demonstrates the importance of using state-of-the-art initial checkpoints and justifies the necessity of this work to adapt sequence-to-sequence models like T5 for ranking.

[1]https://github.com/google-research/t5x
[2]https://github.com/google/rax
[3]https://github.com/google-research/google-research/tree/master/rankt5

Table 1: Comparing ranking performances of different ranking models. The best performance for each data set is bolded. Results with † are statistically significantly ($p \leq 0.05$) better than monoT5.

Model	Loss	MS MARCO				NQ			
		MRR@10	NDCG@5	NDCG@10	MAP	MRR@10	NDCG@5	NDCG@10	MAP
BERT	PointCE	0.3867	0.4127	0.4530	0.3932	0.5157	0.5515	0.5733	0.5228
	Softmax	0.3928	0.4173	0.4580	0.3978	0.5213	0.5566	0.5791	0.5276
monoT5	Generation	0.4156	0.4448	0.4843	0.4206	0.5406	0.5861	0.6079	0.5434
RankT5-EncDec	PointCE	0.4209	0.4496	0.4895†	0.4260†	0.5403	0.5833	0.6079	0.5457†
	Pair	0.4177	0.4456	0.4852	0.4229	0.5574†	0.5957†	0.6198†	0.5629†
	Softmax	0.4278†	0.4573†	0.4960†	0.4326†	0.5687†	**0.6068**†	**0.6291**†	0.5740†
	Poly1	**0.4343**†	**0.4640**†	**0.5025**†	**0.4383**†	0.5647†	0.6032†	0.6255†	0.5701†
RankT5-Enc	PointCE	0.4216†	0.4509†	0.4888	0.4269†	0.5441	0.5851	0.6099	0.5496†
	Pair	0.4206	0.4513†	0.4891†	0.4256	0.5595†	0.5980†	0.6215†	0.5650†
	Softmax	0.4305†	0.4582†	0.4987†	0.4347†	0.5620†	0.6018†	0.6231†	0.5674†
	Poly1	0.4296†	0.4586†	0.4970†	0.4344†	**0.5689**†	**0.6068**†	0.6279†	**0.5744**†



(a) MS MARCO (b) NQ

Figure 2: Comparing performance with different T5 model sizes. Ranking models are RankT5-EncDec fine-tuned with different losses. The 95% confidence intervals are plotted.

Table 2: Zero-shot performance comparison. Ranking models are RankT5-Enc fine-tuned on the MS MARCO data set with different losses. The performance is measured by NDCG@10. The best performance for each data set is bolded.

Data set	PointCE	Softmax	Data set	PointCE	Softmax
TREC-COVID	0.7522	**0.8071**	Touché-2020	**0.4594**	0.4401
BioASQ	0.5346	**0.5635**	Quora	0.8221	**0.8309**
NFCorpus	0.3263	**0.3810**	DBPedia	0.4345	**0.4422**
NQ	0.5959	**0.6142**	SCIDOCS	**0.1821**	0.1806
HotpotQA	**0.7126**	0.7100	FEVER	**0.8352**	0.8316
FiQA-2018	0.4156	**0.4450**	Climate-FEVER	0.2062	**0.2152**
Signal-1M	0.3153	**0.3200**	SciFact	0.7493	**0.7499**
ArguAna	0.2232	**0.3300**	**Average**	0.5024	**0.5241**

We do not find a consistent winner between the encoder-decoder (EncDec) and the encoder-only (Enc) model structure. A possible explanation is that the T5 decoder is less important when the model is fine-tuned for text ranking tasks with sufficient training data.

Model size comparison. We examine how the T5 model size affects the ranking performance. We fine-tune the RankT5-EncDec model with the Softmax and the PointCE loss with different sizes of T5 model checkpoints ("Base", "Large" and "3B"). We evaluate the model performance on both data sets measured by MRR@10. Results are plotted in Figure 2.

The first observation is that the performance consistently improves when the model size increases (+7% for 3B vs. Base on NQ), highlighting the potential to enable even larger language models [6] in a similar method for better ranking performance. Another observation is that models with Softmax consistently outperform PointCE (all statistically significant with $p \leq 0.05$) and the gaps remain relatively stable across different model sizes. This might suggest that the extra benefits brought by using ranking loss cannot be compensated for by simply using larger models.

Zero-shot results. We also compare the zero-shot performance of our ranking models fine-tuned with different ranking losses. We use a subset of BEIR [42] data sets[4] with easily accessible corpus. We take the RankT5-Enc models fine-tuned on the MS MARCO data set with the PointCE loss and the Softmax loss respectively, and apply them to rerank top-1000 documents returned by BM25 [39]. Table 2 summarizes ranking performance measured by NDCG@10.

The ranking model fine-tuned with the Softmax loss outperforms the PointCE loss on 11 out of the 15 data sets. On average, the Softmax loss achieves more than +2.1% NDCG@10 (statistically significant with $p \leq 0.05$) which indicates that using the Softmax loss produces ranking models that generalize better to out-of-domain data. In particular, using the Softmax loss achieves larger improvement on data sets with drastically different corpus (e.g., TREC-COVID, BioASQ, NFCorpus), implying that fine-tuning the model with appropriate ranking losses can enforce the model to put less emphasis on memorization, and thus to better learn the abstract concept of "relevance" [21], regardless of what the underlying corpus is.

7 CONCLUSION

In this paper, we investigate the use of pretrained T5 models for text ranking. We propose two T5 model variants that directly output ranking scores. We then fine-tune these models with ranking losses, which significantly improves ranking metrics on the MS MARCO and the NQ data sets. We also show that this improvement is maintained in the zero-shot setting on out-of-domain data sets.

[4]Notice that the NQ data set in BEIR is has a different corpus and query set from the NQ data set we used earlier.

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] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems*, Vol. 33. 1877–1901.

[3] Sebastian Bruch, Xuanhui Wang, Michael Bendersky, and Marc Najork. 2019. An Analysis of the Softmax Cross Entropy Loss for Learning-to-Rank with Binary Relevance. In *Proceedings of the 2019 ACM SIGIR International Conference on Theory of Information Retrieval*. 75–78.

[4] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. 2005. Learning to rank using gradient descent. In *Proceedings of the 22nd International Conference on Machine learning*. 89–96.

[5] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. 2007. Learning to Rank: From Pairwise Approach to Listwise Approach. In *Proceedings of the 24th International Conference on Machine Learning*. 129–136.

[6] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2022. PaLM: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311* (2022).

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. 4171–4186.

[8] Cicero dos Santos, Xiaofei Ma, Ramesh Nallapati, Zhiheng Huang, and Bing Xiang. 2020. Beyond [CLS] through Ranking by Generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*. 1722–1727.

[9] Luyu Gao, Zhuyun Dai, and Jamie Callan. 2020. Modularized Transfomer-based Ranking Framework. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*. 4180–4190.

[10] Luyu Gao, Zhuyun Dai, and Jamie Callan. 2021. Rethink training of BERT rerankers in multi-stage retrieval pipeline. In *European Conference on Information Retrieval*. Springer, 280–286.

[11] Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. SimCSE: Simple Contrastive Learning of Sentence Embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 6894–6910.

[12] Shuguang Han, Xuanhui Wang, Mike Bendersky, and Marc Najork. 2020. Learning-to-Rank with BERT in TF-Ranking. *arXiv preprint arXiv:2004.08476* (2020).

[13] Kai Hui, Honglei Zhuang, Tao Chen, Zhen Qin, Jing Lu, Dara Bahri, Ji Ma, Jai Prakash Gupta, Cicero Nogueira dos Santos, Yi Tay, and Don Metzler. 2022. ED2LM: Encoder-Decoder to Language Model for Faster Document Re-ranking Inference. In *Findings of the Association for Computational Linguistics: ACL 2022*. 3747–3758.

[14] Rolf Jagerman, Zhen Qin, Xuanhui Wang, Michael Bendersky, and Marc Najork. 2022. On Optimizing Top-K Metrics for Neural Ranking Models. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2303–2307.

[15] Rolf Jagerman, Xuanhui Wang, Honglei Zhuang, Zhen Qin, Michael Bendersky, and Marc Najork. 2022. Rax: Composable Learning-to-Rank using JAX. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 3051–3060.

[16] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems* 20, 4 (2002), 422–446.

[17] Jia-Huei Ju, Jheng-Hong Yang, and Chuan-Ju Wang. 2021. Text-to-text Multi-view Learning for Passage Re-ranking. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1803–1807.

[18] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense Passage Retrieval for Open-Domain Question Answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*. 6769–6781.

[19] Omar Khattab and Matei Zaharia. 2020. Col-BERT: Efficient and Effective Passage Search via Contextualized late Interaction over BERT. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 39–48.

[20] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics* 7 (2019), 453–466.

[21] John Lafferty and Chengxiang Zhai. 2003. Probabilistic relevance models based on document and query generation. In *Language modeling for information retrieval*. Springer, 1–10.

[22] Zhaoqi Leng, Mingxing Tan, Chenxi Liu, Ekin Dogus Cubuk, Jay Shi, Shuyang Cheng, and Dragomir Anguelov. 2022. PolyLoss: A Polynomial Expansion Perspective of Classification Loss Functions. In *International Conference on Learning Representations*.

[23] Jimmy Lin, Rodrigo Nogueira, and Andrew Yates. 2021. *Pretrained transformers for text ranking: BERT and beyond*. Morgan & Claypool Publishers.

[24] Binsheng Liu, Hamed Zamani, Xiaolu Lu, and J Shane Culpepper. 2021. Generalizing discriminative retrieval models using generative tasks. In *Proceedings of the Web Conference 2021*. 3745–3756.

[25] Tie-Yan Liu. 2009. *Learning to Rank for Information Retrieval*. Now Publishers Inc.

[26] Jing Lu, Gustavo Hernández Ábrego, Ji Ma, Jianmo Ni, and Yinfei Yang. 2021. Multi-stage Training with Improved Negative Contrast for Neural Passage Retrieval. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. 6091–6103.

[27] Yuxiang Lu, Yiding Liu, Jiaxiang Liu, Yunsheng Shi, Zhengjie Huang, Shikun Feng Yu Sun, Hao Tian, Hua Wu, Shuaiqiang Wang, Dawei Yin, et al. 2022. ERNIE-Search: Bridging Cross-Encoder with Dual-Encoder via Self On-the-fly Distillation for Dense Passage Retrieval. *arXiv preprint arXiv:2205.09153* (2022).

[28] Sean MacAvaney, Franco Maria Nardini, Raffaele Perego, Nicola Tonellotto, Nazli Goharian, and Ophir Frieder. 2020. Efficient document re-ranking for transformers by precomputing term representations. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 49–58.

[29] Sean MacAvaney, Nicola Tonellotto, and Craig Macdonald. 2022. Adaptive re-ranking with a corpus graph. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 1491–1500.

[30] Jianmo Ni, Gustavo Hernandez Abrego, Noah Constant, Ji Ma, Keith Hall, Daniel Cer, and Yinfei Yang. 2022. Sentence-T5: Scalable Sentence Encoders from Pre-trained Text-to-Text Models. In *Findings of the Association for Computational Linguistics: ACL 2022*. 1864–1874.

[31] Jianmo Ni, Chen Qu, Jing Lu, Zhuyun Dai, Gustavo Hernández Ábrego, Ji Ma, Vincent Y Zhao, Yi Luan, Keith B Hall, Ming-Wei Chang, et al. 2022. Large dual encoders are generalizable retrievers. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*.

[32] Rodrigo Nogueira, Zhiying Jiang, Ronak Pradeep, and Jimmy Lin. 2020. Document Ranking with a Pretrained Sequence-to-Sequence Model. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*. 708–718.

[33] Rodrigo Nogueira, Wei Yang, Kyunghyun Cho, and Jimmy Lin. 2019. Multi-stage document ranking with BERT. *arXiv preprint arXiv:1910.14424* (2019).

[34] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Gray, et al. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*.

[35] Ronak Pradeep, Rodrigo Nogueira, and Jimmy Lin. 2021. The expando-mono-duo design pattern for text ranking with pretrained sequence-to-sequence models. *arXiv preprint arXiv:2101.05667* (2021).

[36] Zhen Qin, Le Yan, Honglei Zhuang, Yi Tay, Rama Kumar Pasumarthi, Xuanhui Wang, Michael Bendersky, and Marc Najork. 2021. Are Neural Rankers Still Outperformed by Gradient Boosted Decision Trees?. In *International Conference on Learning Representations*.

[37] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research* 21 (2020), 1–67.

[38] Ruiyang Ren, Yingqi Qu, Jing Liu, Wayne Xin Zhao, Qiaoqiao She, Hua Wu, Haifeng Wang, and Ji-Rong Wen. 2021. RocketQAv2: A Joint Training Method for Dense Passage Retrieval and Passage Re-ranking. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. 2825–2835.

[39] Stephen Robertson and Hugo Zaragoza. 2009. *The probabilistic relevance framework: BM25 and beyond*. Now Publishers Inc.

[40] Devendra Singh Sachan, Mike Lewis, Mandar Joshi, Armen Aghajanyan, Wen tau Yih, Joëlle Pineau, and Luke Zettlemoyer. 2022. Improving Passage Retrieval with Zero-Shot Question Generation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*.

[41] Yi Tay, Vinh Q Tran, Mostafa Dehghani, Jianmo Ni, Dara Bahri, Harsh Mehta, Zhen Qin, Kai Hui, Zhe Zhao, Jai Gupta, et al. 2022. Transformer memory as a differentiable search index. In *Advances in Neural Information Processing Systems*.

[42] Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. BEIR: A Heterogeneous Benchmark for Zero-shot Evaluation of Information Retrieval Models. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

[43] Ellen M Voorhees. 1999. The TREC-8 Question Answering Track Report. In *Proceedings of the 8th Text REtrieval Conference (TREC-8)*, Vol. 99. 77–82.

[44] Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2022. SimLM: Pre-training with representation bottleneck for dense passage retrieval. *arXiv preprint arXiv:2207.02578* (2022).

[45] Shitao Xiao and Zheng Liu. 2022. RetroMAE v2: Duplex Masked Auto-Encoder For Pre-Training Retrieval-Oriented Language Models. *arXiv preprint arXiv:2211.08769* (2022).

[46] Shitao Xiao, Zheng Liu, Yingxia Shao, and Zhao Cao. 2022. RetroMAE: Pre-Training Retrieval-oriented Language Models Via Masked Auto-Encoder. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 538–548.

[47] Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul N Bennett, Junaid Ahmed, and Arnold Overwijk. 2020. Approximate Nearest Neighbor Negative Contrastive Learning for Dense Text Retrieval. In *International Conference on Learning Representations*.

[48] Yanzhao Zhang, Dingkun Long, Guangwei Xu, and Pengjun Xie. 2022. HLATR: enhance multi-stage text retrieval with hybrid list aware transformer reranking. *arXiv preprint arXiv:2205.10569* (2022).

[49] Shengyao Zhuang, Hang Li, and Guido Zuccon. 2021. Deep Query Likelihood Model for Information Retrieval. In *European Conference On Information Retrieval*. Springer, 463–470.

[50] Shengyao Zhuang and Guido Zuccon. 2021. TILDE: Term Independent Likelihood moDEl for Passage Re-ranking. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1483–1492.