Comparing Score Aggregation Approaches for Document Retrieval with Pretrained Transformers

Xinyu Zhang[1(✉)] , Andrew Yates[2], and Jimmy Lin[1]

[1] David R. Cheriton School of Computer Science, University of Waterloo, Waterloo, Canada
[2] Max Planck Institute for Informatics, Saarbrücken, Germany

Abstract. While BERT has been shown to be effective for passage retrieval, its maximum input length limitation poses a challenge when applying the model to document retrieval. In this work, we reproduce three passage score aggregation approaches proposed by Dai and Callan [5] for overcoming this limitation. After reproducing their results, we generalize their findings through experiments with a new dataset and experiment with other pretrained transformers that share similarities with BERT. We find that these BERT variants are not more effective for document retrieval in isolation, but can lead to increased effectiveness when combined with "pre–fine-tuning" on the MS MARCO passage dataset. Finally, we investigate whether there is a difference between fine-tuning models on "deep" judgments (i.e., fewer queries with many judgments each) vs. fine-tuning on "shallow" judgments (i.e., many queries with fewer judgments each). Based on available data from two different datasets, we find that the two approaches perform similarly.

1 Introduction

In the context of text retrieval, pretrained transformers such as BERT [6] have been shown to substantially improve ranking effectiveness across many domains, tasks, and settings [10]. Adapting BERT to passage retrieval is straightforward: it can be used as a classifier to predict the relevance of a passage with respect to a query, and such a relevance prediction model can be used to rerank candidate passages retrieved by an efficient first-stage keyword-based ranking method like BM25. However, BERT's maximum length limitation of 512 tokens prevents this approach from directly being applied to longer input texts like full-length documents. Several solutions have been proposed to address this issue by breaking a document into passages and then aggregating passage-level relevance to arrive at a document relevance score [1,5,9,12].

In this paper, we reproduce one such approach proposed by Dai and Callan [5]. Their approach segments documents into passages that can each be scored independently. At inference time, Dai and Callan [5] use one of three approaches to aggregate passage-level scores, called FirstP, MaxP, and SumP, which either

D. Hiemstra et al. (Eds.): ECIR 2021, LNCS 12657, pp. 150–163, 2021.
https://doi.org/10.1007/978-3-030-72240-1_11

takes the score of the first passage as the document score, the score of the maximum passage, or the sum of all passage scores, respectively. Dai and Callan [5] considered title and description queries on the Robust04 and ClueWeb09 test collections, finding that taking the maximum passage score as the document score (i.e., MaxP) was the most effective approach except when using description queries on ClueWeb09. However, the differences between MaxP and SumP were small in all settings.

Instead of replicating these results using the code[1] provided by Dai and Callan [5], we first independently reproduce their findings on Robust04 by implementing their approach with the Capreolus toolkit [18]. Note that our focus here is not to exactly obtain the same ranking metrics as their paper, but to attempt to reproduce their findings about the relative effectiveness of the various score aggregation approaches. Our Tensorflow v2 implementation is completely independent from the original code, which used Tensorflow v1 with an entirely different pipeline. In addition to the three approaches proposed in the paper, we introduce a new aggregation approach, AvgP, to compare with SumP and investigate the impact of document length. Our results show that the original findings are reproducible, though we observe much larger differences between MaxP and SumP than in the original work. In our results, MaxP consistently and significantly outperforms FirstP, SumP, and AvgP. As in the original work, we also find that BERT is more effective with description queries than with keyword queries.

Given that we are able to reproduce the results of Dai and Callan [5] on Robust04, we omit experiments on the ClueWeb09 collection. Instead, to further generalize the above findings and to provide a reference for the community, we apply the four aggregation approaches to the GOV2 test collection.[2] While we continue to observe a larger gap between MaxP and SumP than previously reported, our findings on GOV2 are consistent with those on Robust04: (1) MaxP is more effective than FirstP, SumP, and AvgP, and (2) description queries are more effective than keyword queries.

Since Dai and Callan [5] first demonstrated the effectiveness of MaxP for document retrieval, several BERT variants have been proposed that claim to improve BERT's effectiveness on NLP tasks by making architectural changes, e.g., sharing the same weights across all transformer layers [8] and changes to the pretraining setup such as removing the next sentence prediction task [11]. It is natural to ask whether retrieval can benefit from these model improvements and, if so, how much of an increase in effectiveness can be provided by using an improved variant. To answer this question, we repeated the above experiments with MaxP, the most effective aggregation approach, with different pretrained neural language models: RoBERTa [11], ALBERT [8], and ELECTRA [4].

In addition to the finding that pretrained language models improve effectiveness on ranking tasks, Dai and Callan [5] found that "pre–fine-tuning" BERT on Bing search log data further improves effectiveness (i.e., fine-tuning BERT on Bing data before further fine-tuning on the target dataset). Li et al. [9] pro-

[1] https://github.com/AdeDZY/SIGIR19-BERT-IR.
[2] http://ir.dcs.gla.ac.uk/test_collections/gov2-summary.htm.

vide further support for the benefit of pre–fine-tuning, and found that the MS MARCO passage dataset is more effective for this task than the Bing search logs. Furthermore, Zhang et al. [20] found that pre–fine-tuning $\text{BERT}_{\text{Base}}$ improves effectiveness regardless of the amount of data used to fine-tune for the downstream task. To validate these findings and to compare the impact of pretraining and pre–fine-tuning, we additionally consider whether the effectiveness of MaxP increases with pre–fine-tuning on MS MARCO.

Finally, we investigate the impact of different strategies for gathering relevance judgments on the effectiveness of MaxP. Traditionally, the Text REtrieval Conferences (TRECs) build test collections with "deep" judgments, in which a large number of judgments are obtained for a relatively small number of queries (typically, around 50). However, neural models are often trained on relevant query–document pairs or triples (queries with positive and negative instances), so it is unclear whether the "deep" approach of TREC is preferable to using many more queries but with fewer judgments per query (i.e., a "shallow" judgment approach). The recent MS MARCO dataset takes this shallow approach by providing a large number of queries that are associated with only one relevant document on average [3]. This dataset has become popular for training neural models. Similarly, the TREC 2007 Million Query dataset [2] provides shallow judgments and has also been used to train neural models for this reason [7,14]. To provide a more comprehensive view of how to best apply the BERT–MaxP model, we investigate the effectiveness of these two types of training data. Interestingly, based on available data from two different datasets, we find that the two approaches perform similarly (unlike Yilmaz and Robertson [19]).

In summary, the contributions of this work are:

1. We reproduce and confirm the findings of Dai and Callan [5] on Robust04 and further generalize the findings to the GOV2 test collection.
2. We investigate two approaches to obtaining "free" improvements in ranking effectiveness: using improved BERT variants or "pre–fine-tuning" on another retrieval dataset. The different BERT variants we examined bring no significant improvements, but pre–fine-tuning with MS MARCO data does improve effectiveness.
3. We investigate the impact of "deep" vs. "shallow" judgments on BERT–MaxP. At least for the datasets and sample sizes we explore, both approaches obtain similar levels of effectiveness.

2 Related Work

2.1 Passage Aggregation

Prior work has investigated several approaches for overcoming BERT's maximum length limitation by segmenting long documents into shorter passages. However, no consensus has been reached on how per-passage results should be aggregated. Dai and Callan [5] were the first to propose and evaluate different strategies

for aggregating document scores. To do so, Dai and Callan [5] segment each document into N overlapping passages; each passage receives the relevance label of the document at training time. They compared three approaches to aggregate passage-level scores at inference time: FirstP, MaxP, and SumP. Given N passage scores from the same document, FirstP uses the score of the first passage as the document score, MaxP uses the highest passage score, and SumP uses the sum of all passage scores. Even though FirstP only uses the first passage from the document when computing document scores, it is not identical to truncating all documents in the corpus since the model is trained using all passages from the document. That is, although most passages do not directly contribute to the document score, they contribute to model fine-tuning.

Birch [1], another approach for aggregating passage scores, improves effectiveness by interpolating the top-k sentence-level scores, where $k \in 1, 2, 3$. To train the Birch model, datasets with passage-level judgments are used (e.g., MS MARCO and tweets). The model is then adapted for a target domain with longer documents by learning only the weights for the top-k scores as well as an interpolation weight for the first-stage ranker. Note that before these approaches, monoBERT [13] considered passage datasets where all "documents" were shorter than the model length limit, and thus the entire text can be fed into BERT at both training and inference time.

Rather than aggregating passage scores, MacAvaney et al. [12] concatenate the term representations BERT produces for each passage in order to form a document vector. This document vector is then used to construct a similarity matrix, which is used to compute a relevance score. Some variants of this approach additionally include the average of BERT's [CLS] representation of each passage. Li et al. [9] investigate additional approaches for aggregating passage representations instead of aggregating passage scores directly. They find that several strategies can improve over score aggregation.

In this work, we reproduce and extend the experiments in Dai and Callan [5] on different aggregation approaches. Note that since we apply other BERT variants to initialize this model (see Sect. 2.2), we use MaxP when referring to the general model architecture to avoid ambiguity and only use BERT–MaxP when the model is initialized with $\text{BERT}_{\text{Base}}$.

2.2 BERT Variants

While Devlin et al. [6] proposed several BERT variants with different model sizes (e.g., 110M weights with $\text{BERT}_{\text{Base}}$ and 330M weights with $\text{BERT}_{\text{Large}}$), additional variants have been proposed that purport to improve the model in different ways. RoBERTa [11] found that BERT's effectiveness on NLP tasks can be improved by modifying the training data and tuning pretraining hyperparameters. Additionally, RoBERTa eliminates the Next Sentence Prediction (NSP) objective as it was found to be ineffective for improving downstream tasks.

ALBERT [8] proposed to reduce BERT's parameters by factorizing word embedding into smaller matrices and sharing the parameters of each BERT layer. They found that, while these strategies compress the model size and accelerate

pretraining given the same model configuration, the pretrained model still performs roughly on par with $\text{BERT}_{\text{Base}}$. This work additionally replaced the NSP task with Sentence Ordering Prediction (SOP), where the model is given two segments from the same document and learns to discriminate whether the two segments have been swapped. They found that the SOP task improves effectiveness on most of the downstream NLP tasks considered.

ELECTRA [4] improved representation learning efficiency by replacing the Masked Language Modeling (MLM) task with a new task called replaced token detection. In this task the model classifies whether *each* output token was generated by another small "generator" model or was the original token. The generator is a small two-layer BERT model that predicts masked tokens. While this approach requires training the generator model as well as the ELECTRA model, the new objective enables the model to learn from the output at all the positions, rather than just the 15% of the positions that are randomly masked in BERT's pretraining.

3 Experimental Setup

In this section, we describe in detail the BERT score aggregation approaches in our study, our approach for experimenting with other BERT variants, our methodology for generating deep and shallow judgments, and finally the experiment configurations.

3.1 BERT with MaxP, FirstP, SumP, and AvgP Aggregation

To apply BERT as a relevance classifier for text ranking, Nogueira and Cho [13] proposed feeding a query q and passage p to BERT to obtain a vector $\mathtt{E}_{\mathtt{CLS}}$ representing the interactions between them. To do so, a special `[CLS]` token is prepended to the input sequence, and a special `[SEP]` token is placed before and after the passage. This usage of the `[CLS]` vector follows the approach for applying a pretrained BERT model to classification tasks proposed by Devlin et al. [6]. This `[CLS]` vector is then fed to a fully-connected layer with two outputs followed by a softmax. The score of the positive class serves as the relevance score s used to rank the passages.

This approach is referred to as monoBERT. BERT's maximum input length limitation of 512 tokens[3] prevents this strategy from being directly applied to longer documents, however. In the work we are reproducing, Dai and Callan [5] proposed overcoming this limitation by converting a document d into a series of passages p_i, applying BERT as a relevance classifier to each passage p_i to obtain a series of relevance scores s_i, and then applying a score aggregation approach

[3] The length of BERT's inputs cannot exceed 512 tokens. This includes the query, the passage, and the three special tokens. This limitation comes from the fact that position embeddings are used to encode BERT's input; these position embeddings were only pretrained for sequences up to length 512.

to arrive at a final document relevance score s_d. To generate the passages, Dai and Callan [5] used a sliding window of 150 terms with a stride of 75.

Given this sequence of passages, one of three aggregation approaches was applied: taking the maximum passage score as the document score (MaxP), taking the first passage's score (FirstP), or taking the sum of all passage scores (SumP). We additionally consider an AvgP variant in which the sum of scores is divided by the number of passages in the document.

3.2 BERT Variants

In the original work, Dai and Callan [5] used $\text{BERT}_{\text{Base}}$ as a relevance classifier to obtain the scores s_i for aggregation. In addition to conducting experiments in this setting, we also experiment with using the larger $\text{BERT}_{\text{Large}}$ model provided by Devlin et al. [6], as well as the RoBERTa [11], ALBERT [8], and ELECTRA [4] models in their "base" sizes. Apart from the general-purpose pretrained models, we fine-tune $\text{BERT}_{\text{Base}}$ and $\text{ELECTRA}_{\text{Base}}$ using the MS MARCO passage dataset and add these pre–fine-tuned weights into our comparisons. These models can be viewed as drop-in replacements for BERT; to use them, we simply replace $\text{BERT}_{\text{Base}}$ with a different variant when computing $\mathtt{E}_{\mathtt{CLS}}$.

In the experiments investigating each pretrained model, we use the models available in the HuggingFace model hub [15], with names bert-base-uncased, bert-large-uncased, google/electra-base-discriminator, albert-base-v2 and roberta-base. For the experiment investigating the impact of MS MARCO pre–fine-tuning, we use the $\text{BERT}_{\text{Base}}$ weights provided by Nogueira and Cho [13] and the $\text{ELECTRA}_{\text{Base}}$ weights provided by Li et al. [9].

3.3 Deep and Shallow Sampling

In order to investigate whether it is preferable to use "deep and narrow" or "shallow and wide" judgments for training, we sample judgments from an existing test collection to simulate both cases. To accomplish this, we prepare ten smaller datasets from each of the Robust04[4] and GOV2[5] datasets by sampling the relevance judgments in a "shallow" or "deep" manner with a sampling rate r, described below.

Given the same number of judgments, the shallow setting contains more queries and fewer labeled documents per query, whereas the deep setting contains fewer queries with more labeled documents per query. The shallow setting is used in MS MARCO [3], whereas the deep setting is traditionally used in TREC evaluations. Shallow and deep sampling are two sampling schemes that we adopted to simulate these two labeling styles, respectively. The sampling approach we adopted in previous work [20] can be viewed as deep sampling, which provides a reference point for this paper.

[4] https://trec.nist.gov/data/robust/04.guidelines.html.
[5] http://ir.dcs.gla.ac.uk/test_collections/gov2-summary.htm.

Specifically, given a dataset with Q queries, D documents per query, and M judgments in total, where $M = Q \cdot D$, the r-sampled dataset always contains $r \cdot D$ judgments. Deep sampling accomplishes this by dropping queries with higher priority, while shallow sampling only drops the documents associated with each query and always preserves the original number of queries.

We achieve this with a two-step process. In the first step, deep sampling randomly preserves around $\lceil r \cdot Q \rceil$ queries, and shallow sampling randomly preserves $\lceil r \cdot D \rceil$ documents per query. At this point, both sampling mechanisms should produce slightly more than $r \cdot M$ judgments. In the second step, we eliminate the extra judgments by looping over the queries and randomly dropping one of its labeled documents until exactly $r \cdot D$ judgments are left.

Note that we use cross-validation in our experiments and the sampled datasets are only used in the training and validation folds. Test folds always contain the original judgments. That is, while the model is trained and validated on sampled data, it is evaluated with all available judgments to make fair comparisons.

3.4 Experimental Details

All the configurations are run on both the Robust04 and GOV2 datasets. Robust04 is a TREC collection with documents from the news domain that the original work [5] used in their evaluation. GOV2 contains documents crawled from `.gov` websites, which forms a different domain from Robust04. As in the original work, we use 5-fold cross-validation for Robust04 collection, with three folds for training, one fold for validation, and the other fold for evaluation. While Dai and Callan [5] did make their Robust04 folds available,[6] we opted to instead use the folds from Yang et al. [17] in order to ensure that the choice of folds does not affect the original findings. We randomly assign the queries in GOV2 into three groups and applied 3-fold cross-validation, with one fold for training, one fold for validation, and the other fold for evaluation.

We implement our experiments with the Capreolus toolkit [18]. To produce candidate documents for reranking, we use the Anserini BM25 implementation [16] with default parameters $k_1 = 0.9$ and $b = 0.4$ (i.e., the first-stage ranker). At training time we construct training instances from the top 1000 documents retrieved by BM25. We consider the top 100 documents at inference time since this setting is substantially more efficient (i.e., reranking 1000 documents takes ten times longer). This setting differs from the original work, which used a query-likelihood model as the first-stage ranker. As with the change in folds used, this allows us to provide evidence that the original work's findings are robust to minor changes in the experimental setup.

Following the original work, we generate passages from each document using a 150-term sliding window with a 75-term stride. The maximum number of passages per document is set to 30. During training, passages after the first

[6] http://boston.lti.cs.cmu.edu/appendices/SIGIR2019-Zhuyun-Dai/.

passage are randomly preserved with probability 0.1.[7] We use pairwise hinge loss and fine-tune the models over 36 epochs, with each epoch containing 256 batches of 16 training triples (i.e., a query, a positive document, and a negative document). We run validation every 4 epochs and preserve the best model in terms of nDCG@20 to mitigate overfitting. All experiments are fine-tuned using the Adam optimizer with $lr = 10^{-3}$ for non-BERT parameters and $lr = 10^{-5}$ for BERT parameters. The dropout rate for all fully-connected layers[8] is set to 0.1 except for ALBERT, where the dropout rate is set to 0.

For the reproduction and BERT variant experiments, we consider both keyword queries (title field) and description queries (desc field) on both datasets and report mAP, P@20, and nDCG@20, whereas for experiments comparing sampling mechanisms, we only report nDCG@20 on keyword queries. Our code and instructions are available on GitHub.[9]

4 Results and Discussion

4.1 Reproduction and Generalization of Aggregation Approaches

We report results from our attempts to reproduce Dai and Callan [5] in Table 1, which consist of the FirstP, MaxP, SumP, and AvgP score aggregation approaches with both keyword and description queries on the Robust04 and GOV2 datasets. All models are initialized from $\text{BERT}_{\text{Base}}$. Table 1a shows the Robust04 results copied from the original paper; Table 1b presents our results.

The nDCG@20 column under Robust04 in Table 1b shows that the original work's finding that MaxP outperforms FirstP on Robust04 is reproducible. In fact, we achieve slightly higher results for both methods, which confirms the correctness of our implementation. While MaxP continues to outperform SumP, the difference between these two methods is greater than in the original work. That is, Table 1a shows a tiny difference between the two with both approaches outperforming FirstP. However, in our results, SumP is not more effective than FirstP. Given that the implementation differences between these approaches are very small,[10] we attribute this finding to changes in our experimental setup (e.g., different folds and a different first-stage ranker). This suggests that MaxP is a more robust approach. In our results, MaxP almost always significantly outperforms the other approaches regardless of the query type or the dataset.

4.2 MaxP with BERT Variants

Results when initializing MaxP from different pretrained and pre–fine-tuned (denote "pFT") checkpoints are shown in Table 2. From the table, it can be

[7] https://github.com/AdeDZY/SIGIR19-BERT-IR/blob/master/run_qe_classifier.py#L468-L471.

[8] The hidden_dropout_prob configuration in HuggingFace's library.

[9] https://github.com/crystina-z/MaxP-Reproduction.

[10] See line 58 of `tools/bert_passage_result_to_trec.py` in the original code.

Table 1. Results from the original work and our experiments for each passage score aggregation approach.

Title			Desc		
FirstP	MaxP	SumP	FirstP	MaxP	SumP
0.444	0.469	0.467	0.491	0.529	0.524

(a) The nDCG@20 metrics reported by Dai and Callan [5] on Robust04.

		Robust04			GOV2		
		mAP@100	P@20	nDCG@20	mAP@100	P@20	nDCG@20
Title	FirstP	0.2163†	0.3821	0.4493	0.1730	0.5564	0.4911
	MaxP	**0.2384**	**0.4068**	**0.4767**	**0.1855**	**0.603**	**0.5175**
	SumP	0.2123†	0.3837	0.4476†	0.1679	0.5423	0.4679
	AvgP	0.2083†	0.3749	0.4383	0.1732	0.5594	0.4826
Desc	FirstP	0.2445†	0.4239†	0.5095	0.1811†	0.5803†	0.5213
	MaxP	**0.2646**	**0.4504**	**0.5303**	**0.1942**	**0.6292**	**0.5480**
	SumP	0.2113†	0.3821†	0.4436 †	0.1686†	0.5567†	0.4825†
	AvgP	0.2356†	0.4161†	0.4931†	0.1796†	0.5896	0.5099

(b) Our results with BERT–FirstP, BERT–MaxP, BERT–SumP, and BERT–AvgP. The underlined scores correspond to the values reported in Table 1a (i.e., nDCG@20 on Robust04). The best results are in **bold**. The † symbol indicates the score is significantly lower than the corresponding MaxP score according to a two-tailed t-test ($p < 0.01$) after Bonferroni correction.

Table 2. Results of MaxP models initialized with various pretrained or pre–fine-tuned weights. The † symbol indicates the score is significantly higher than the corresponding $\text{BERT}_{\text{Base}}$ score ($p < 0.01$) after Bonferroni correction. The best results among all pretrained models are underlined, and the best among all are in **bold**. Note that the $_{Base}$ subscript is omitted when there is no ambiguity.

		Robust04			GOV2		
		mAP@100	P@20	nDCG@20	mAP@100	P@20	nDCG@20
Title	$\text{BERT}_{\text{Base}}$	0.2384	0.4068	0.4767	0.1855	0.6030	0.5175
	$\text{BERT}_{\text{Large}}$	0.2424	0.4120	0.4875	0.1865	0.5990	0.5161
	ELECTRA	0.2437	0.4253	0.4959	0.1810	0.5718	0.4841
	RoBERTa	0.2425	0.4259	0.4938	0.1696	0.5591	0.4679
	ALBERT	0.2326	0.4006	0.4632	0.1925	0.6114	0.5354
	$\text{BERT}_{\text{Base}}$ (pFT)	0.2401	0.4207	0.4857	0.1958	0.6322	0.5473
	ELECTRA (pFT)	**0.2575**	**0.4482**†	**0.5225**†	**0.1998**	**0.6466**	**0.5624**
Desc	$\text{BERT}_{\text{Base}}$	0.2646	0.4504	0.5303	0.1942	0.6292	0.5480
	$\text{BERT}_{\text{Large}}$	0.2672	0.4655	0.5448	0.1968	0.6272	0.5420
	ELECTRA	0.2726	0.4584	0.5480	0.1895	0.6081	0.5152
	RoBERTa	0.2692	0.4671	0.5489	0.1928	0.6195	0.5370
	ALBERT	0.2637	0.4542	0.5400	0.1977	0.6309	0.5459
	$\text{BERT}_{\text{Base}}$ (pFT)	0.2719	0.4624	0.5476	0.2046†	0.6550	0.5788
	ELECTRA (pFT)	**0.2865**†	**0.4779**†	**0.5741**†	**0.2100**†	**0.6822**†	**0.6062**†

observed that although each BERT variant *can* achieve an improvement over BERT, such improvements are neither significant nor consistent across datasets or query types. On Robust04, $\text{BERT}_{\text{Large}}$, ELECTRA, and RoBERTa show some improvement over $\text{BERT}_{\text{Base}}$ for both query types, but their results on GOV2 are only on par with or even worse than $\text{BERT}_{\text{Base}}$. On the other hand, ALBERT is less effective than BERT on Robust04 with keyword queries and GOV2 with description queries, but improves over $\text{BERT}_{\text{Base}}$ in the other settings.

Compared with the inconsistent improvements brought by different BERT variants, the benefits of pre–fine-tuning on MS MARCO are much more stable. While the differences are significant only on GOV2, the pre–fine-tuned $\text{BERT}_{\text{Base}}$ numerically outperforms the vanilla $\text{BERT}_{\text{Base}}$ across different query types and datasets. Moreover, the pre–fine-tuned ELECTRA yields an improvement with significant increases in a variety of settings.

4.3 Deep vs. Shallow Relevance Judgments

Table 3 shows the training effectiveness of MaxP across a spectrum of training and validation data sizes. Table 3a shows several baselines to put the results in context, including a BERT-MaxP model fine-tuned on only the MS MARCO collection (i.e., the pre–fine-tuned setting without further fine-tuning on the target domain) and the BERT-MaxP scores previously reported by Dai and Callan [5] and Li et al. [9]. Table 3b shows the BERT-MaxP metrics obtained by fine-tuning with each deep or shallow sampled dataset at different sampling

Table 3. Results of deep and shallow sampling experiments.

	Robust04	**GOV2**
BM25	0.4240	0.4740
BM25RM3	0.4310	0.4851
Zero-shot [20]	0.4751	0.5007
$\text{BERT}_{\text{Base}}$ [5]	0.4690	-
$\text{BERT}_{\text{Base}}$ (pFT) [9]	0.4931	0.5600

(a) nDCG@20 of baselines and prior work. The zero-shot setting uses the $\text{BERT}_{\text{Base}}$ checkpoint fine-tuned on the MS MARCO passage dataset.

	Robust04				**GOV2**			
	$\text{BERT}_{\text{Base}}$		**$\text{BERT}_{\text{Base}}$ (pFT)**		**$\text{BERT}_{\text{Base}}$**		**$\text{BERT}_{\text{Base}}$ (pFT)**	
	shallow	**deep**	**shallow**	**deep**	**shallow**	**deep**	**shallow**	**deep**
$r = 0.1$	0.4087	0.4111	0.4314	0.4427	0.4692	0.4538	0.5034	0.5197
$r = 0.3$	0.4592	0.4507	0.4658	0.4702	0.4861	0.4923	0.5290	0.5236
$r = 0.5$	0.4730	0.4593	0.4807	0.4725	0.5062	0.5039	0.5402	0.5384
$r = 0.7$	0.4779	0.4750	0.4821	0.4807	0.5106	0.5169	0.5358	0.5373
$r = 0.9$	0.4812	0.4711	0.4867	0.4787	0.5222	0.5202	0.5488	0.5429
$r = 1.0$	0.4767		0.4857		0.5175		0.5473	

(b) nDCG@20 on original and pre–fine-tuned (pFT) $\text{BERT}_{\text{Base}}$: results of fine-tuning BERT-MaxP using "deep" and "shallow" r-sampled judgments.

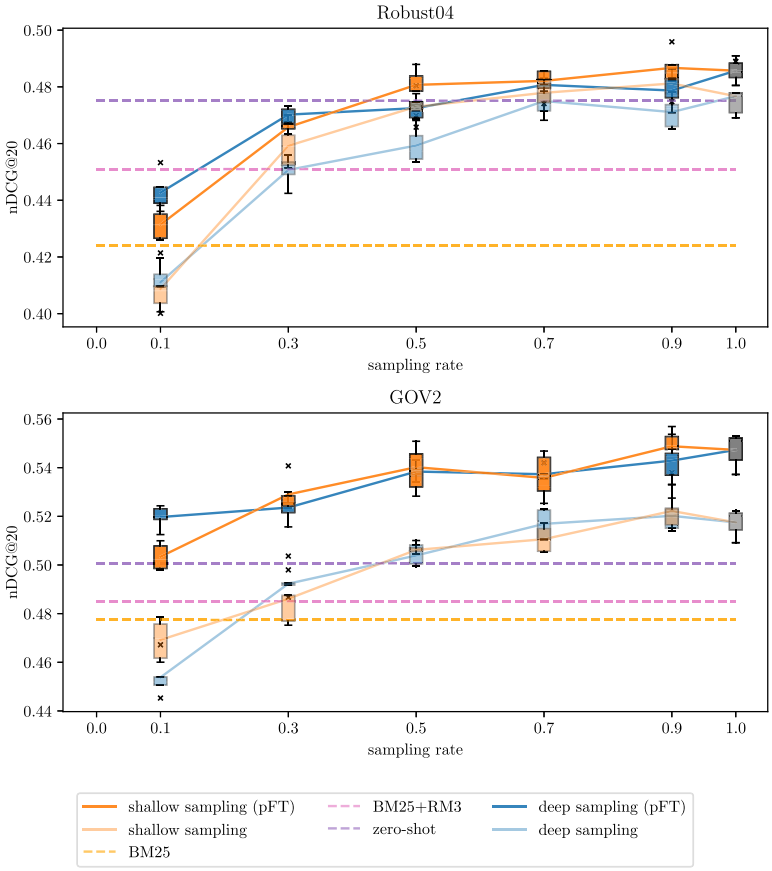

Fig. 1. Plots of baselines and our experiments with deep and shallow sampling.

rates r. As mentioned in Sect. 3.4, we report the median nDCG@20 of the five experiments under the same settings.

By plotting the scores in Fig. 1, it can be observed that while effectiveness benefits from more training and validation labels, there is no clear trend in terms of the superiority of the two schemes. It is not the case that one judgment scheme consistently yields better effectiveness than the other. This observation applies regardless of whether the model is pre–fine-tuned on MS MARCO. This is an interesting finding that differs from the results of Yilmaz and Robertson [19], who conducted similar experiments, but in a feature-based learning-to-rank context. Note that an important caveat here is that our sampling schemes apply only to sampling *training* data—in all cases, our test data are "complete". We have not explored the case where the test data are also sampled, in which case there may be differences between the two schemes for *evaluating* effectiveness.

5 Conclusion

In this work, we reproduced the three passage score aggregation approaches proposed in Dai and Callan [5]. We found that the MaxP aggregation approach is the most effective, and furthermore, the differences between MaxP and AvgP are larger than in the original work. We generalized this finding by conducting the same experiments on the GOV2 dataset and reaching the same conclusion. We found that MaxP can further benefit from pre–fine-tuning the model on the MS MARCO passage dataset, but does not necessarily benefit from replacing BERT with a newer variant. While none of the *general-purpose pretrained* models consistently improved over BERT, the pre–fine-tuned ELECTRA model achieved significant improvements under many settings. Finally, we explored the impact of fine-tuning BERT with shallow or deep judgments via sampling, finding that the model performed similarly regardless of which judgment scheme was used.

Acknowledgments. This research was supported in part by the Canada First Research Excellence Fund and the Natural Sciences and Engineering Research Council (NSERC) of Canada. In addition, we would like to thank Google Cloud and TensorFlow Research Cloud for credits to support this work.

References

1. Akkalyoncu Yilmaz, Z., Yang, W., Zhang, H., Lin, J.: Cross-domain modeling of sentence-level evidence for document retrieval. 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), pp. 3481–3487 (2019)
2. Allan, J., Carterette, B., Aslam, J.A., Pavlu, V., Dachev, B., Kanoulas, E.: Million query track 2007 overview. In: Proceedings of TREC 2007 (2007)
3. Bajaj, P., et al.: MS MARCO: a human generated machine reading comprehension dataset. arXiv preprint arXiv:1611.09268v3 (2018)
4. Clark, K., Luong, M.T., Le, Q.V., Manning, C.D.: ELECTRA: pre-training text encoders as discriminators rather than generators. arXiv preprint arXiv:2003.10555 (2020)
5. Dai, Z., Callan, J.: Deeper text understanding for IR with contextual neural language modeling. In: Proceedings of the 42nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2019), pp. 985–988 (2019)
6. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: 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), pp. 4171–4186 (2019)
7. Fan, Y., Guo, J., Lan, Y., Xu, J., Zhai, C., Cheng, X.: Modeling diverse relevance patterns in ad-hoc retrieval. In: Proceedings of the 41st International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 375–384 (2018)

8. Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., Soricut, R.: ALBERT: a lite BERT for self-supervised learning of language representations. arXiv preprint arXiv:1909.11942 (2019)
9. Li, C., Yates, A., MacAvaney, S., He, B., Sun, Y.: PARADE: passage representation aggregation for document reranking. arXiv preprint arXiv:2008.09093 (2020)
10. Lin, J., Nogueira, R., Yates, A.: Pretrained transformers for text ranking: BERT and beyond. arXiv preprint arXiv:2010.06467 (2020)
11. Liu, Y., et al.: RoBERTa: a robustly optimized BERT pretraining approach. arXiv preprint arXiv:1907.11692 (2019)
12. MacAvaney, S., Yates, A., Cohan, A., Goharian, N.: CEDR: contextualized embeddings for document ranking. In: Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 1101–1104 (2019)
13. Nogueira, R., Cho, K.: Passage re-ranking with BERT. arXiv preprint arXiv:1901.04085 (2019)
14. Pang, L., Lan, Y., Guo, J., Xu, J., Xu, J., Cheng, X.: DeepRank: a new deep architecture for relevance ranking in information retrieval. In: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, pp. 257–266 (2017)
15. Wolf, T., et al.: HuggingFace's transformers: state-of-the-art natural language processing. arXiv preprint arXiv:1910.03771 (2019)
16. Yang, P., Fang, H., Lin, J.: Anserini: enabling the use of Lucene for information retrieval research. In: Proceedings of the 40th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2017), pp. 1253–1256 (2017)
17. Yang, W., Lu, K., Yang, P., Lin, J.: Critically examining the "neural hype" weak baselines and the additivity of effectiveness gains from neural ranking models. In: Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2019), pp. 1129–1132 (2019)
18. Yates, A., Jose, K.M., Zhang, X., Lin, J.: Flexible IR pipelines with Capreolus. In: Proceedings of the 29th ACM International Conference on Information and Knowledge Management, pp. 3181–3188 (2020)
19. Yilmaz, E., Robertson, S.E.: Deep versus shallow judgments in learning to rank. In: Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2009), pp. 662–663 (2009)
20. Zhang, X., Yates, A., Lin, J.: A little bit is worse than none: Ranking with limited training data. In: Proceedings of SustaiNLP: Workshop on Simple and Efficient Natural Language Processing, pp. 107–112 (2020)