Document Ranking with a Pretrained Sequence-to-Sequence Model

Rodrigo Nogueira,[*] **Zhiying Jiang,**[*] **Ronak Pradeep,** and **Jimmy Lin**
David R. Cheriton School of Computer Science
University of Waterloo
{rodrigo.nogueira, zhiying.jiang, rpradeep, jimmylin}@uwaterloo.ca

Abstract

This work proposes the use of a pretrained sequence-to-sequence model for document ranking. Our approach is fundamentally different from a commonly adopted classification-based formulation based on encoder-only pretrained transformer architectures such as BERT. We show how a sequence-to-sequence model can be trained to generate relevance labels as "target tokens", and how the underlying logits of these target tokens can be interpreted as relevance probabilities for ranking. Experimental results on the MS MARCO passage ranking task show that our ranking approach is superior to strong encoder-only models. On three other document retrieval test collections, we demonstrate a zero-shot transfer-based approach that outperforms previous state-of-the-art models requiring in-domain cross-validation. Furthermore, we find that our approach significantly outperforms an encoder-only architecture in a data-poor setting. We investigate this observation in more detail by varying target tokens to probe the model's use of latent knowledge. Surprisingly, we find that the choice of target tokens impacts effectiveness, even for words that are closely related semantically. This finding sheds some light on why our sequence-to-sequence formulation for document ranking is effective. Code and models are available at pygaggle.ai.

1 Introduction

A simple, straightforward formulation of ranking is to convert the task into a classification problem, and then sort the candidate items to be ranked based on the probability that each item belongs to the desired class. Applied to the document ranking problem in information retrieval—where given a query, the system's task is to return a ranked list of documents from a large corpus that maximizes some ranking metric such as average precision or nDCG—the simplest formulation is to deploy a classifier that estimates the probability each document belongs to the "relevant" class, and then sort all the candidates by these estimates.

Deep transformer models pretrained with language modeling objectives, exemplified by BERT (Devlin et al., 2019), have proven highly effective in a variety of classification and sequence labeling tasks in NLP; Nogueira and Cho (2019) are the first to demonstrate their effectiveness in ranking tasks. Since it is impractical to apply inference to *every* document in a corpus with respect to a query, these techniques are typically applied to *rerank* a list of candidates. In a typical end-to-end system, these candidates are taken from the results of a keyword search based on a "classic" IR scoring function such as BM25 (Robertson et al., 1994). This leads to the standard multi-stage pipeline architecture where first-stage retrieval is followed by reranking using one or more machine learning models (Asadi and Lin, 2013; Nogueira et al., 2019a). This architecture underlies nearly all transformer-based approaches to document retrieval today, for example, CEDR (MacAvaney et al., 2019), BERT–MaxP (Dai and Callan, 2019), Birch (Yilmaz et al., 2019), and PARADE (Li et al., 2020).

Applying BERT (and its variants) to document ranking can be characterized as a classification-based encoder-only approach. In contrast, we explore the use of a sequence-to-sequence encoder–decoder architecture—specifically, T5 (Raffel et al., 2020)—to ranking, which requires a trick to coax relevance probabilities out of model-generated "target tokens". We show that in a data-rich setting, with sufficient training examples, our approach outperforms a classification-based encoder-only model. However, our sequence-to-sequence model appears to be far more data-efficient, significantly outperforming BERT with few training examples

[*]Equal contribution.

in a data-poor setting. The main advantage of our approach is that by "connecting" fine-tuned latent representations of relevance to output target tokens, we can exploit the model's latent knowledge (e.g., of semantics, linguistic relations, etc.) that has been captured through pretraining. We describe probing experiments that attempt to verify our intuitions by deliberately altering the target tokens to capture different aspects of "semantic relatedness".

The contribution of this work is to present a novel approach to document ranking using a pretrained sequence-to-sequence model. While ranking with classification-based encoder-only architectures (BERT and variants) is commonplace today, we are the first to describe ranking with encoder–decoder architectures and articulate its advantages. Additional ablation and contrastive experiments reveal new insights on fundamental differences between these two approaches, and our technique to probe model behavior by manipulating the output target tokens is also methodologically novel.

2 Seq2Seq Ranking

The main idea behind the Text-to-Text Transfer Transformer (T5) by Raffel et al. (2020) is to cast *every* natural language processing task—for example, machine translation, question answering, and classification—as feeding a sequence-to-sequence model some input text and training it to generate some output text. These include tasks that can be naturally viewed as "sequence in, sequence out" (e.g., machine translation) as well as tasks for which a sequence-to-sequence formulation seems unnatural (e.g., coreference resolution). The T5 architecture can be viewed as a natural progression of "vanilla transformers" by Vaswani et al. (2017), but with pretraining inspired by BERT's masked language model objective. Like BERT, a pretrained T5 model is then fine-tuned on various downstream tasks, where each task is associated with a specific "input template". For example, to translate text from English to German, the sentence to be translated is prefixed with the literal phrase "translate English to German:".

We follow the same approach and formulate document ranking as a relevance prediction problem, i.e., the task is to estimate a relevance score that quantifies the extent to which a candidate document is relevant to a query. We devise the following input template to capture this task:

$$\text{Query: } \texttt{[Q]} \text{ Document: } \texttt{[D]} \text{ Relevant:} \quad (1)$$

where `[Q]` and `[D]` are replaced with the query and document texts, respectively. The model is fine-tuned to produce the tokens "true" or "false" depending on whether the document is relevant or not to the query. That is, "true" and "false" are the target tokens (i.e., ground truth predictions in the sequence-to-sequence transformation).

It is, however, not obvious exactly how, at inference time, such a fine-tuned model can be used for ranking. All the tasks that Raffel et al. (2020) detail for T5 are, at a high-level, functions of a single inference pass: for translation, there is only a single sentence to be translated, and for natural language entailment and related tasks, hypothesis pairs are encoded into a single input template. For ranking, the setup is different, as it is not feasible to encode *all* the candidate documents (from first-stage retrieval) into a single input template. Thus, ranking necessitates multiple inference passes with the model and somehow aggregating the outputs.

After some amount of empirical exploration we arrived at an effective solution (see Section 5.3 for more details). At inference time, to extract useful probabilities from the model, we apply a softmax only on the logits of the "true" and "false" tokens. In other words, we compute $\mathrm{Pr}(\text{relevant} = 1|q, d)$, as the probability assigned to the "true" token normalized in this manner. This estimate is interpreted as the relevance score for each query–document pair. Each candidate document from first-stage retrieval is independently fed to the model, and the final document ranking is simply a permutation of the initial candidate documents based on these estimated probabilities in descending order.

Although this trick may seem obvious in retrospect, we are quite certain of its novelty—a lead author of the T5 paper (Raffel), in personal communication, affirmed that the authors never tried anything along these lines before because there was no need for the tasks that they were tackling.

Note that T5 tokenizes sequences using the SentencePiece model (Kudo and Richardson, 2018), which might split a word into subwords. We choose target tokens ("true" and "false") that are represented as single words; thus, each class is represented by a single logit. In the case where target tokens are split in multiple subwords, we would need a method to aggregate their logits into a single score; we thought it best to avoid this complexity.

Our formulation naturally begs the question: Why "true" and "false" as the target tokens? We discuss this question in Section 5.4. However, as a preview, we find that the choice of target tokens has a large impact on effectiveness in some circumstances, and these experiments shed light on *why* T5 works well for document ranking.

True to the original motivation of Raffel et al. (2020), we explore the transfer capabilities of T5 (recall, the model name stands for Text-to-Text *Transfer* Transformer) by experimenting with zero-shot document ranking on different datasets. To summarize, we fine-tune the model on the MS MARCO passage dataset and directly apply it on three other test collections commonly used by the information retrieval community. This requires a modification to rank long documents at inference time, which we describe below.

Finally, while our experiments only examine T5, we note that our method can be used with any other pretrained sequence-to-sequence model such as BART (Lewis et al., 2020), MASS (Song et al., 2019), UniLM (Dong et al., 2019), and Pegasus (Zhang et al., 2020). We leave explorations of these models for future work.

3 Experimental Setup

3.1 Datasets

We use the following datasets in our experiments:

MS MARCO passage (Bajaj et al., 2016) is a ranking dataset with 8.8M passages obtained from Bing search engine results with around 1M natural language questions. Note that for terminological consistency, we refer to each "unit" in the corpus as a document, even though they are in reality paragraph-length passages. The training set contains approximately 530K (query, relevant document) pairs, with on average one relevant passage per unique query; non-relevant documents are also provided as part of the training set. The development and test sets contain approximately 6,900 queries each, but relevance labels are only publicly available for the development set. Effectiveness on the test set requires submission to the leaderboard.

Robust04 (Voorhees, 2004) is the test collection from the TREC 2004 Robust Track. It comprises 249 topics, with relevance judgments on a collection of $\sim$528K documents (TREC Disks 4 and 5).

Core17 (Allan et al., 2017) is the test collection from the TREC 2017 Common Core Track, with relevance judgments for 50 topics on $\sim$1.86M articles from the New York Times Annotated Corpus.

Core18 (Allan et al., 2018) is the test collection from the TREC 2018 Common Core Track, with relevance judgments for 50 topics on $\sim$600K articles from the TREC Washington Post Corpus.

For Robust04, Core17, and Core18, we use the topic "titles" (short keyword phrases, much like the input to a search engine) as queries to our bag-of-words retrieval methods (see Section 3.3) and the topic "descriptions" (sentence-length statements of information needs) as input to our sequence-to-sequence models. These topic descriptions are more similar to MS MARCO's natural language questions, and others have found that using well-formed questions improves the effectiveness of pretrained reranking models (Dai and Callan, 2019).

A point worth reemphasizing: our models are *not* trained on Robust04, Core17, or Core18 data. We use their queries and relevance judgments only as held-out test sets; thus, for those collections, our evaluation adopts a zero-shot transfer setting.

3.2 Training and Inference

We fine-tune our T5 models (base, large, and 3B) with a constant learning rate of 10^{-3} for 100K iterations (approx. ten epochs) with class-balanced batches of size 128. We are not able to conduct experiments with T5-11B due to its computational cost. To simplify our training procedure (and related hyperparameters) as well as to eliminate the need for convergence checks, we simply train for a fixed number of iterations, selected based on the computational demands of our largest model and the (self-allotted) time for running experiments. We report results using the model state at the final checkpoint. This procedure is consistent with the advice of Kaplan et al. (2020) and recommendations by Dodge et al. (2019), since we quantify effectiveness for a particular computational budget. We use a maximum of 512 input tokens and two output tokens (one for the target token and another for the end-of-sequence token). In the MS MARCO passage dataset, none of the inputs exceed this length limitation. Training T5 base, large, and 3B take approximately 12, 48, and 160 hours overall, respectively, on a single Google TPU v3-8.

For inference, we adopt greedy decoding. Since we only use the logits of the first decoding step, beam search and top-k random sampling (Fan et al., 2018) would give the same results.

Because Robust04, Core17, and Core18 contain full-length documents, during inference it is not possible to directly feed the *entire* text at once to our model due to length restrictions. To address this issue, we first segment each document into passages by applying a sliding window of 10 sentences with a stride of 5. We then obtain a relevance probability for each passage by classifying it independently. We select the highest probability among these passages as the relevance probability of the document; that is, we do not use the original (BM25) retrieval scores.[1] This procedure is the same as the MaxP technique of Dai and Callan (2019) although our definition of passages differs.

3.3 Baselines

We compare against the following baselines:

BM25: For a baseline bag-of-words retrieval method, we use the BM25 implementation in the Anserini open-source IR toolkit (Yang et al., 2017),[2] which is based on Lucene. We adopt all the default settings. At inference time, we retrieve the top 1000 documents per query.

BM25+RM3: To examine the effects of query expansion, we apply the BM25+RM3 model as described in Yang et al. (2019), where it is shown to be a competitive baseline for pre-BERT neural ranking models. We use the implementation in Anserini, with all default settings.

BM25+BERT-large: We additionally compare our method against the BERT-large condition from Nogueira et al. (2019a), which is a two-stage pipeline with bag-of-words retrieval (BM25) followed by a BERT reranker. Architecturally, it is the same as our method, the only difference being BERT vs. T5 as the reranking model. Nogueira et al. (2019a) can be characterized as the baseline of the best methods from the official MS MARCO passage leaderboard; all higher-ranked submissions can be described as improvements upon this basic approach, and thus it represents a competitive comparison point. Note that we do not apply reranking on top of BM25+RM3 because RM3 is known to reduce effectiveness when evaluated using these relevance judgments (Nogueira et al., 2019b).

Our T5 rerankers are applied directly to the output of BM25 (and BM25+RM3) from Anserini (1000 hits), thus providing a contrastive setup that isolates the impact of our method.

[1]We also examined the alternative of interpolating model scores with retrieval scores, but this did not improve effectiveness and additionally introduces an extra parameter to tune.

[2]http://anserini.io/

		MS MARCO Passage	
	# Params	Dev	Test
BM25	-	0.184	0.186
+ BERT-large	340 M	0.372	0.365
+ T5-base	220 M	0.381	-
+ T5-large	770 M	0.393	-
+ T5-3B	3 B	**0.398**	0.388

Table 1: MRR@10 figures on the MS MARCO passage, with BERT-large figures from Nogueira et al. (2019a). Model sizes are also shown.

4 Results

Main results on the MS MARCO passage retrieval task are shown in Table 1, comparing BERT-large (Nogueira et al., 2019a) to T5 models of different sizes. MRR@10 is the official metric for the task. Based on the Student's paired t-test, the effectiveness of T5-3B (bolded) on the development set is significantly better ($p < 0.01$) than T5-large. Effectiveness increasing with larger models is an expected trend, and with T5-11B we might obtain an even higher MRR@10; unfortunately, we are not able to run these experiments due to their high computational costs.

Results on Robust04, Core17, and Core18 are shown in Table 2, where we apply our T5 reranker on top of retrieval results from BM25 and BM25+RM3 (see Section 3.2). The T5-3B results in bold are significantly better ($p < 0.05$) than T5-large, T5-base, and the corresponding baseline (BM25 or BM25+RM3), based on the Student's paired t-test with Bonferroni corrections. We compare our model with Birch (Yilmaz et al., 2019), BERT–MaxP (Dai and Callan, 2019), and PARADE (Li et al., 2020), which are BERT-based models that represent the state of the art. BERT–MaxP and PARADE results are from fine-tuning on the MS MARCO data and then fine-tuning again on Robust04 (via cross-validation).[3] Birch uses Robust04, Core17, and Core18 for tuning weighting parameters. In contrast, we apply inference directly using our model trained on the MS MARCO passage data; Robust04, Core17, and Core18 relevance judgments are only used as a test set, which makes our results zero-shot. To our knowledge, our T5-3B model produces the highest known scores reported on these test collections.

[3]MaxP numbers are from the reimplementation by Li et al. (2020), which are higher than the original paper due to additional fine-tuning on MS MARCO.

	Robust04			Core17			Core18		
Model	AP	nDCG@20	Jdg@20	AP	nDCG@20	Jdg@20	AP	nDCG@20	Jdg@20
Birch	0.3697	0.5325	-	0.3323	0.5092	-	0.3522	0.4953	-
BERT–MaxP	-	0.5453	-	-	-	-	-	-	-
PARADE	-	0.5713	-	-	-	-	-	-	-
BM25	0.2531	0.4240	0.9770	0.2087	0.3877	0.9550	0.2495	0.4100	0.9620
+ T5-base	0.3279	0.5298	0.9158	0.2758	0.5180	0.8840	0.3125	0.4741	0.8020
+ T5-large	0.3288	0.5345	0.8906	0.2799	0.5356	0.9090	0.3330	0.5057	0.8200
+ T5-3B	**0.3876**	**0.6091**	0.9632	**0.3193**	**0.5629**	0.9260	**0.3749**	**0.5493**	0.8600
BM25 + RM3	0.2903	0.4407	0.9764	0.2823	0.4467	0.9620	0.3135	0.4604	0.9390
+ T5-base	0.3340	0.5532	0.9058	0.3067	0.5203	0.8840	0.3364	0.4698	0.7990
+ T5-large	0.3382	0.5287	0.8840	0.3109	0.5299	0.8880	0.3557	0.5007	0.8070
+ T5-3B	**0.4062**	**0.6122**	0.9588	**0.3564**	**0.5612**	0.9100	**0.3998**	**0.5492**	0.8540

Table 2: Results on Robust04, Core17, and Core18. The T5 models are trained only on MS MARCO passage data and thus represent zero-shot transfer. Jdg@20 is the percentage of top-20 retrieved documents that were judged.

Note that results from our T5 models have lower proportions of judged documents in the top-20 (Jdg@20) than BM25 and BM25+RM3. In other words, our models are retrieving documents that have never been evaluated, for which we have no relevance labels. Since standard evaluation tools such as `trec_eval` treat "unknown" as not relevant, the results for our models represent a lower bound on true effectiveness. This finding confirms recent observations that test collections built before the advent of BERT-based rerankers place transformer-based models at a disadvantage (Yilmaz et al., 2020).

As we expect, effectiveness increases with larger models, but in all cases T5 improves over both a bag-of-words as well as a query expansion baseline. Note that the latter is considered to be a strong baseline, even for pre-BERT neural ranking models (Yang et al., 2019). In many cases, we notice that the effectiveness improvement of T5-large over T5-base is small; we investigate this curious finding further in Section 5.2.

5 Analysis

5.1 Effect of Model Size and Training Data

Results from the MS MARCO passage ranking task (Table 1) represent a direct comparison between BERT and T5 since the retrieval pipeline is otherwise the same. For Robust04, Core17, and Core18 (Table 2), we adopt a different architecture than PARADE, BERT–MaxP, and Birch, but effectiveness clearly improves as the size of the T5 model increases. While T5 achieves better results, it is possible that the improvements come from simply having a bigger model, as opposed to any intrinsic advantages over an encoder-only architecture. Since we do not have pretrained T5 and BERT models of comparable sizes, it is difficult to conduct a fair empirical comparison. However, we do note from Table 1 that T5-base outperforms the larger BERT-large model.

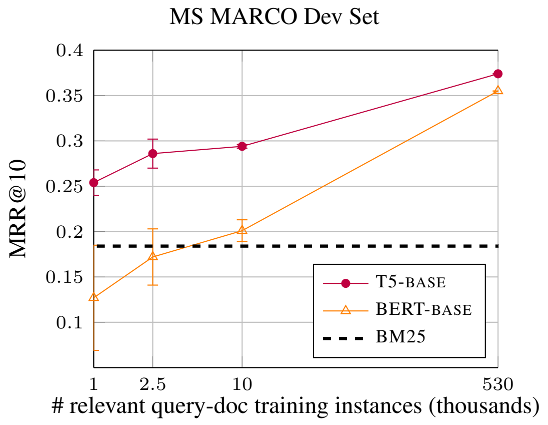

Figure 1: Comparisons between T5-base and BERT-base trained with different numbers of training instances (note the log scale in the x-axis). Results report means and 95% confidence intervals over five trials.

Another important dimension of size is the amount of training data available, as it is often expensive to annotate high-quality data for information retrieval. In Figure 1, we report the results of experiments fine-tuning BERT-base and T5-base with 1K, 2.5K, and 10K positive instances (and an equal number of negative instances) sampled from the full MS MARCO passage dataset. We select these two "base" models due to their more modest computational demands for fine-tuning. We train

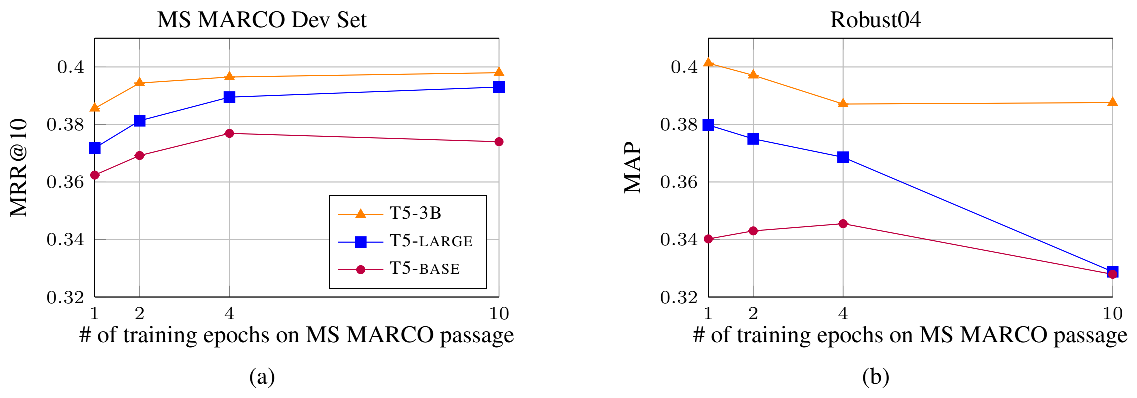

Figure 2: **(a)** MRR@10 vs. number of training epochs on MS MARCO. **(b)** MAP on Robust04 vs. number of training epochs on MS MARCO.

them using a batch size of 32 for three epochs. For BERT, we use a learning rate 10^{-6} and no warm-up step. For T5, we use a learning rate of 10^{-3}. Note that these differences in experimental methodology render the results not directly comparable to those in Table 1. For all conditions (2K, 5K, and 20K samples in total), we repeat the experiment five times, drawing different samples each time; the 95% confidence intervals are shown in Figure 1. We run the setting with 530K training instances only once due to its high computational cost.

As we expect, effectiveness improves as we fine-tune both models with more data. Interestingly, in a data-poor setting with only a modest amount of training data, T5 can learn far more effectively than BERT. We see clearly that with the same amount of *limited* training data (10K positive instances is only about 2% of the entire dataset), T5 is significantly more effective than BM25. In fact, with only 1K positive and 1K negative training instances, BERT performs worse than the BM25 baseline (i.e., worse than just exact term matching), while T5 is 7 points better than the BM25 baseline. With 10K training instances, BERT is able to modestly improve upon BM25, but remains nine points behind T5 fine-tuned on the same amount of data. Interestingly, T5 is able to achieve roughly 10 points above the BM25 baseline, which accounts for nearly 60% of its total gain, with only 2% of the training data.

5.2 Effect of Checkpoint Selection

The application of our T5 approach to Robust04, Core17, and Core18 is zero shot since the model is never exposed to labeled training data from those collections.[4] We apply the fine-tuning procedure described in Section 3.2 and directly evaluate on those test collections. Results in Table 2, however, revealed an oddity: the effectiveness of T5-large is not substantially better than T5-base, contrary to our expectations. Further investigation reveal this to be an issue of "how much to fine-tune".

In Figure 2(a), we show MRR@10 vs. number of training epochs on MS MARCO, and in Figure 2(b), a similar graph for MAP on Robust04 (reranking BM25 results). On MS MARCO, effectiveness increases overall as we fine-tune the model for more epochs, with the exception of T5-base, which exhibits signs of over-training. These findings are expected. On Robust04, however, exhibits signs of over-training for all model sizes. It makes sense that fine-tuning more and more on a specific dataset would reduce the model's ability to generalize to other domains. This observation also suggests that we can obtain even better results than those in Table 2 if we apply our model on an earlier checkpoint.

Proper checkpoint selection, however, requires in-domain validation data, which no longer qualifies as zero shot. We emphasize that this diagnostic experiment was conducted *after* obtaining the zero-shot results reported in Table 2 and thus does not invalidate our zero-shot claims. We are unsure if our observations are merely idiosyncrasies of document ranking, or a more general problem with transfer learning using transformers. Nevertheless, this is an issue deserving further exploration.

5.3 Effect of Logit Normalization

There does not appear to be a principled reason why normalizing only "true" and "false" logits via a softmax would be more effective than a number of equally sensical alternatives. For example,

[4]It is possible, however, that during pretraining the model was exposed to documents from the target corpus.

Logit Normalization Technique	MRR@10
(1) None (“true” logit only)	0.026
(2) Softmax on all logits	0.379
(3) Softmax on “true”/“false” logits only	0.381

Table 3: T5-base results on the development set of the MS MARCO passage dataset comparing different logit normalization techniques.

we could rerank documents according to the logit of the “true” token only or using logits of all tokens to compute the softmax. Here, we investigate the effectiveness of these alternative normalization techniques.

In Table 3, we show T5-base results on the development set of the MS MARCO passage dataset. In the first row, we simply use the logit of the “true” token as the score of the document. This method performs poorly, with an MRR@10 close to zero. Normalizing with a softmax over either all logits (row 2) or only the “true” and “false” logits (row 3) yields similarly high MRR@10 figures. These results demonstrate that the logits of a particular token (in this case, the “true” token) are not comparable across different examples, but they become comparable once normalized appropriately. The method in row 3 is the default method throughout the paper because it gives slightly better results.

5.4 Target Token Probing Experiments

The experimental results above immediately raise two questions:

1. Why is our approach more data-efficient than BERT? That is, why does T5 significantly outperform BERT when fine-tuned with far fewer training examples?
2. How is our approach fundamentally different from classification with an encoder-only model, given that the softmax in our case reduces the model to a binary classifier?

We believe these two issues are closely related. Specifically addressing the second question: At a high level, both neural models are learning latent representations important to the task at hand (in this case, relevance classification), starting from a pretrained model, and then mapping these latent representations into task-specific decisions. Thus, end-to-end effectiveness depends on a combination of the knowledge imparted via pretraining (already present at the start) and the knowledge gained via fine-tuning on task-specific data. In the classification-based approach using BERT proposed by Nogueira and Cho (2019), the model relies on a single fully-connected layer to map the latent representation (i.e., the `[CLS]` token) into this binary decision. While the approach can exploit pretrained knowledge when fine-tuning the latent representations, the final mapping (i.e., the fully-connected layer) needs to be learned from scratch (since it is randomly initialized).[5]

In contrast, T5 can exploit both pretrained knowledge and knowledge gleaned from fine-tuning in learning task-specific latent representations as well as the mapping to relevance decisions; specifically, we note that T5 is pretrained with tasks whose outputs are “true” and “false”. Unlike the fully-connected layer in the encoder-only approach, T5 can exploit the part of the network used for generating output tokens. Embedded in that neural machinery is latent knowledge about semantics, linguistic relations, and lexical features that are necessary to generate fluent text. In other words, T5 has access to an additional source of knowledge that BERT does not.

This explanation, we believe, also answers the first question. With plenty of training data, BERT has no trouble learning the final fully-connected layer (mapping latent representations to decisions), even from scratch (i.e., random initialization). However, faced with few training examples, BERT still must learn the classification layer, but without any benefit from pretraining—and the experiments in Figure 1 show that it is unable to do so effectively. In contrast, in a low-data setting, T5 can “fall back” on pretrained neural machinery for generating fluent textual output. In other words, the pretraining objective in T5 seems to transfer well to *generating* relevance labels.

To turn our intuition into testable hypotheses, we can vary the target tokens used as the prediction targets and manipulate their “linguistic relatedness”—to deliberately “disrupt” linguistic knowledge that may be captured in the model. As Puri and Catanzaro (2019) show, the choice of target tokens impacts effectiveness. Recall that in our baseline, “true” indicates a relevant document and “false”, a non-relevant document. We investigate the following contrastive variants:

[5]While other models such as PARADE (Li et al., 2020) layer additional neural components on top of BERT, our basic argument still holds since these additional parts of the model are also randomly initialized.

	Target Token		Training Size (query-relevant doc pairs)		
Type	Relevant	Non-Relevant	1K	10K	530K (all)
Baseline	true	false	0.254 ±0.014	0.294 ±0.002	0.374
Alternate	yes	no	0.218 ±0.040	0.301 ±0.004	0.378
Reverse	false	true	0.243 ±0.025	0.282 ±0.006	0.374
Antonyms	hot	cold	0.240 ±0.021	0.246 ±0.005	0.375
Related Words	apple	orange	0.206 ±0.026	0.260 ±0.003	0.376
Unrelated Words	hot	orange	0.194 ±0.018	0.242 ±0.005	0.377
Subwords	_ab	_de	0.179 ±0.014	0.228 ±0.005	0.377

Table 4: Results with T5-base on the development set of the MS MARCO passage dataset comparing different target token manipulations.

- **"Alternate".** Instead of "true" and "false", we use "yes" and "no", respectively. Here we are probing with an equally intuitive formulation of the targets, except that these words have not been used in pretraining, and thus the model is less likely to have strong prior associations.

- **"Reverse".** We swap the target tokens; that is, "false" indicates a relevant document and "true", a non-relevant document. If the model is indeed exploiting latent knowledge about linguistic relations, then forcing the model to make opposite associations on the same polarity scale should lower effectiveness with respect to the baseline.

- **"Antonyms".** We map a relevant document to "hot" and a non-relevant document to "cold". This preserves the use of adjectives at opposite ends of a polarity scale, but a scale that is completely unrelated to relevance. If the model is exploiting latent knowledge, we would expect effectiveness to be lower than the baseline.

- **"Related Words".** We map a relevant document to "apple" and a non-relevant document to a related word "orange". These words are semantically related, but do not present a polarity contrast as before. We would expect effectiveness to be lower than the baseline.

- **"Unrelated Words".** We map a relevant document to "hot" and a non-relevant document to a completely unrelated word "orange". Thus, we force the model to build an arbitrary semantic mapping. We would expect effectiveness to be lower than the baseline and also lower than using related words.

- **"Subwords".** We map a relevant document to the subword "_ab" and a non-relevant document to the subword "_de". Note that we carefully select single tokens after tokenization by SentencePiece. Here, we remove all "semantics" from the input–output mapping and thus expect effectiveness to be lower than the above conditions.

Using these target token configurations, we conduct experiments on T5-base with either 1K (or 10K) positive and 1K (or 10K) negative instances sampled from the full MS MARCO passage dataset, same as in Section 5.1. Once again, for each of the conditions, we repeat the experiment five times, drawing different samples every time. For reference, we also fine-tune with all available data. Note that the effectiveness of T5-base is different from the values in Table 1 because we use slightly different (more computationally-efficient) hyperparameters: here, we train for 40K steps using a batch of size 256. Experimental results are shown in Table 4, with means and 95% confidence intervals.

When fine-tuning with all available data, the choice of target tokens has negligible impact on effectiveness. These small differences can be explained by the stochastic nature of the training process. This does appear consistent with our hypothesis that with sufficient training data, T5 is able to learn arbitrary mappings between document relevance and target tokens.

In the data-poor setting, the results are also consistent with our hypotheses. With minimal amounts of training data (the 1K condition), the confidence intervals from different samples mostly overlap (with the exception of subwords), so we do not have the benefit of greater certainty that comes with statistical significance. In the 10K condition, our target token manipulations all significantly reduce effectiveness, except for the "Alternate" condition, which performs slightly better than the baseline condition. This seems somewhat idiosyncratic, but we suspect that the prevalence of the target tokens in the data used for pretraining might have an impact: yes/no appear more often in the pretraining corpus than true/false. Overall, it is clear that the semantics of the target tokens, even small differences, can affect the overall effectiveness of the model. The "Unrelated Words" and "Subwords" conditions are clearly less effective. Finally, we note that the 95% confidence intervals are smaller under the 10K condition, which illustrates the greater instability in effectiveness when training on smaller datasets (which is expected).

These results support our hypothesis that T5 is exploiting latent knowledge to aid in predicting relevance. As the strongest piece of evidence, in the 1K condition, "Subwords" performs worse than the BM25 baseline; i.e., it exhibits difficulty achieving any predictive power at all. There are at least two potential factors at play: we are removing semantic associations, as the subwords are token fragments, and furthermore, we are forcing the model to produce tokens in an order (and context) that it has not encountered during pretraining. We are unable to tease apart these effects currently, but either explanation is consistent with our intuitions. For all other target token manipulations, we are at least able to beat the BM25 baseline under the 1K condition.

Finally, our experiments are inconclusive regarding the importance of having a polarity scale in the low-data setting. Quite clearly, reversing "true" and "false" has a noticeable impact (especially in the 10K condition), but T5 is more effective learning targets that are semantically related but do not present a polarity contrast ("apple" and "orange") than targets that encode an unrelated polarity contrast ("hot" and "cold"). Due to computational limitations (primarily from the number of trials necessary to obtain confidence intervals), we experiment with only one target token pair for each category; additional trials with different targets will be required to draw firmer conclusions.

6 Related Work

As with natural language processing, the advent of deep learning has transformed the information retrieval community. Prior to deep learning, researchers and practitioners mostly adopt the paradigm known as "learning to rank", which is heavily driven by manual feature engineering (Liu, 2009; Li, 2011). For example, commercial web search engines are known to incorporate thousands of features (or more) in their models. The introduction of continuous vector space representations coupled with neural models was exciting as it provides a potential path away from the need for handcrafted features. Well-known early neural ranking models include DRMM (Guo et al., 2016), DUET (Mitra et al., 2017), KNRM (Xiong et al., 2017), and Co-PACRR (Hui et al., 2018); the literature is too vast for an exhaustive review here, and thus we refer readers to past overviews (Onal et al., 2018; Mitra and Craswell, 2019). Interestingly, however, a meta-analysis by Yang et al. (2019) finds that without sufficient training data, these neural models still perform worse than well-tuned bag-of-words query expansion baselines.

However, in the past year or so, we have witnessed a dramatic shift to ranking models based on BERT, starting with Nogueira and Cho (2019). The current state of the art models (Yilmaz et al., 2019; Dai and Callan, 2019; Li et al., 2020) serve as the points of comparisons in Table 2. Our work belongs to this large family of models based on transformers, although our exploration of a sequence-to-sequence ranking formulation based on encoder–decoder architectures sets us apart from previous classification-based formulations using encoder-only architectures.

7 Conclusion

The main contribution of this paper is to introduce a novel generation-based approach to document ranking using pretrained sequence-to-sequence models. Our models outperform a classification-based encoder-only approach, especially in the data-poor setting with limited training data. We attempt to explain these observations in terms of hypotheses about the knowledge that a model gains from pretraining vs. fine-tuning on task-specific data. These hypotheses are operationalized into target token probing experiments, where we demonstrate that the model appears to exploit knowledge from its ability to generate fluent natural language text. Exactly how remains an open research question and the focus of ongoing work.

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 for credits to support this work.

References

James Allan, Donna Harman, Evangelos Kanoulas, Dan Li, Christophe Van Gysel, and Ellen Voorhees. 2017. TREC 2017 common core track overview. In *Proceedings of the Twenty-Sixth Text REtrieval Conference (TREC 2017)*.

James Allan, Donna Harman, Evangelos Kanoulas, and Ellen Voorhees. 2018. TREC 2018 common core track overview. In *Proceedings of the Twenty-Seventh Text REtrieval Conference (TREC 2018)*.

Nima Asadi and Jimmy Lin. 2013. Effectiveness/efficiency tradeoffs for candidate generation in multi-stage retrieval architectures. In *Proceedings of the 36th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2013)*, pages 997–1000.

Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, Mir Rosenberg, Xia Song, Alina Stoica, Saurabh Tiwary, and Tong Wang. 2016. MS MARCO: A human generated MAchine Reading COmprehension dataset. *arXiv:1611.09268*.

Zhuyun Dai and Jamie Callan. 2019. 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)*, pages 985–988, Paris, France.

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)*, pages 4171–4186.

Jesse Dodge, Suchin Gururangan, Dallas Card, Roy Schwartz, and Noah A. Smith. 2019. Show your work: Improved reporting of experimental results. 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)*, pages 2185–2194.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. In *Advances in Neural Information Processing Systems*, pages 13063–13075.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. *arXiv:1805.04833*.

Jiafeng Guo, Yixing Fan, Qingyao Ai, and W. Bruce Croft. 2016. A deep relevance matching model for ad-hoc retrieval. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 55–64.

Kai Hui, Andrew Yates, Klaus Berberich, and Gerard de Melo. 2018. Co-PACRR: A context-aware neural IR model for ad-hoc retrieval. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining (WSDM 2018)*, pages 279–287.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv:2001.08361*.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pretraining for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880.

Canjia Li, Andrew Yates, Sean MacAvaney, Ben He, and Yingfei Sun. 2020. PARADE: Passage representation aggregation for document reranking. *arXiv:2008.09093*.

Hang Li. 2011. *Learning to Rank for Information Retrieval and Natural Language Processing*. Morgan & Claypool Publishers.

Tie-Yan Liu. 2009. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331.

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*, pages 1101–1104.

Bhaskar Mitra and Nick Craswell. 2019. An introduction to neural information retrieval. *Foundations and Trends in Information Retrieval*, 13(1):1–126.

Bhaskar Mitra, Fernando Diaz, and Nick Craswell. 2017. Learning to match using local and distributed representations of text for web search. In *Proceedings of the 26th International Conference on World Wide Web (WWW 2017)*, pages 1291–1299.

Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage re-ranking with BERT. *arXiv:1901.04085*.

Rodrigo Nogueira, Wei Yang, Kyunghyun Cho, and Jimmy Lin. 2019a. Multi-stage document ranking with BERT. *arXiv:1910.14424*.

Rodrigo Nogueira, Wei Yang, Jimmy Lin, and Kyunghyun Cho. 2019b. Document expansion by query prediction. *arXiv:1904.08375*.

Kezban Dilek Onal, Ye Zhang, Ismail Sengor Altingovde, Md Mustafizur Rahman, Pinar Karagoz, Alex Braylan, Brandon Dang, Heng-Lu Chang, Henna Kim, Quinten McNamara, Aaron Angert, Edward Banner, Vivek Khetan, Tyler McDonnell, An Thanh Nguyen, Dan Xu, Byron C. Wallace, Maarten de Rijke, and Matthew Lease. 2018. Neural information retrieval: At the end of the early years. *Information Retrieval*, 21(2–3):111–182.

Raul Puri and Bryan Catanzaro. 2019. Zero-shot text classification with generative language models. *arXiv:1912.10165*.

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*, 21:1–67.

Stephen E. Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford. 1994. Okapi at TREC-3. In *Proceedings of the 3rd Text REtrieval Conference (TREC-3)*, pages 109–126.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2019. MASS: Masked sequence to sequence pre-training for language generation. In *Proceedings of the 36th International Conference on Machine Learning*, pages 5926–5936.

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*, pages 5998–6008.

Ellen M. Voorhees. 2004. Overview of the TREC 2004 Robust Track. In *Proceedings of the Thirteenth Text REtrieval Conference (TREC 2004)*, pages 52–69.

Chenyan Xiong, Zhuyun Dai, Jamie Callan, Zhiyuan Liu, and Russell Power. 2017. End-to-end neural ad-hoc ranking with kernel pooling. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2017)*, pages 55–64.

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*, pages 1253–1256.

Wei Yang, Kuang Lu, Peilin Yang, and Jimmy Lin. 2019. 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*, pages 1129–1132.

Emine Yilmaz, Nick Craswell, Bhaskar Mitra, and Daniel Campos. 2020. On the reliability of test collections for evaluating systems of different types. In *Proceedings of the 43rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2020)*, pages 2101–2104.

Zeynep Akkalyoncu Yilmaz, Wei Yang, Haotian Zhang, and Jimmy Lin. 2019. 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)*, pages 3481–3487.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter J. Liu. 2020. PEGASUS: Pre-training with extracted gap-sentences for abstractive summarization. In *Proceedings of the 37th International Conference on Machine Learning (ICML 2020)*, pages 2021–2032.