InPars-Light: Cost-Effective Unsupervised Training of Efficient Rankers

Leonid Boytsov*
leo@boytsov.info
Amazon AWS AI Labs
Pittsburgh, USA

Preksha Patel†
Vivek Sourabh†
Riddhi Nisar†
Sayani Kundu†
Ramya Ramanathan†
Carnegie Mellon University
Pittsburgh, USA

Eric Nyberg
ehn@cs.cmu.edu
Carnegie Mellon University
Pittsburgh, USA

ABSTRACT

We carried out a reproducibility study of InPars recipe for unsupervised training of neural rankers [4]. As a by-product of this study, we developed a simple-yet-effective modification of InPars, which we called InPars-light. Unlike InPars, InPars-light uses only a freely available language model BLOOM and 7x-100x smaller ranking models. On all five English retrieval collections (used in the original InPars study) we obtained substantial (7-30%) *and* statistically significant improvements over BM25 in nDCG or MRR using only a 30M parameter six-layer MiniLM ranker. In contrast, in the InPars study only a 100x larger MonoT5-3B model consistently outperformed BM25, whereas their smaller MonoT5-220M model (which is still 7x larger than our MiniLM ranker), outperformed BM25 only on MS MARCO and TREC DL 2020. In a purely unsupervised setting, our 435M parameter DeBERTA v3 ranker was roughly at par with the 7x larger MonoT5-3B: In fact, on three out of five datasets, it slightly outperformed MonoT5-3B. Finally, these good results were achieved by re-ranking only 100 candidate documents compared to 1000 used in InPars. We believe that InPars-light is the first truly cost-effective prompt-based unsupervised recipe to train and deploy neural ranking models that outperform BM25.

CCS CONCEPTS

• **Information systems → Retrieval models and ranking**.

KEYWORDS

unsupervised training, neural information retrieval, ranking

1 INTRODUCTION

Training effective neural IR models often requires abundant *in-domain* training data, which can be quite costly to obtain: Judging a single document-query pair takes at least one minute on average [13, 19] and a single query typically needs at least 50 of such judgements [7].[1] In that, models trained on out-of-domain data and/or fine-tuned using a small number of in-domain queries are often worse or perform only marginally better [31, 47] than simple non-neural BM25 rankers.

A recent trend to deal with this problem consists in generating *synthetic in-domain* training data via prompting of Large Language Models (LLMs) [4, 9, 43], However, proposed solutions are not cost effective. Furthermore, because researchers used primarily proprietary LLMs—whose training procedure was not controlled by the scientific community—or instruction-finetuned LLMs there is a question of whether information retrieval capabilities emerge, indeed, from a large scale training with a next-token-prediction objective.

To enable a practical solution for a large scale LLM-based generation of training data for IR systems, as well as to answer this key scientific question, we carry out a reproducibility study of InPars [4] using medium-size (at most 7B parameters) open-source LLMs [44, 50]. In addition, our study employs more practical and more commonly used BERT-based cross-encoder models having as little as 30 million parameters. In contrast, the original study employed large MonoT5 rankers of which only MonoT5-3B ranker (with 3 billion parameters) performed well [35].

We discover that in a purely unsupervised setting we can replace an impractical three-billion parameter MonoT5-3B model [35] with a 7x smaller bi-directional BERT model while obtaining comparable results. Moreover, unlike the original InPars study where a 220M million MonoT5 model fails to outperform BM25 on three out of five datasets, we show that a much smaller MiniLM model with only 30 million parameters [53] can always outperform BM25 by 7-30% in key metrics (nDCG@K or MRR).

In that, obtaining these good results required re-ranking only 100 candidate documents compared to 1000 in the InPars study [4]. Compared to InPars, our recipe—which we call *InPars-Light*—is substantially more cost effective in three dimensions: (a) synthetic data generation, (b) model training, and (c) inference.

In our study we ask the following research questions:

- **RQ1:** Do information retrieval (IR) capabilities emerge merely from a large-scale next-token-prediction training? Although, prior studies (see § 2) provide limited evidence to answer this question positively, we believed that additional evidenced was required.
- **RQ2:** Are open-source models more or less useful for generation of synthetic IR training data compared to the similar-size GPT-3 model?

*Work done outside of the scope of employment.
†Equal contribution: Work done while studying at Carnegie Mellon University.
[1] Robust04 and TREC-COVID collections used in our study have about 1K judgements per query.

- **RQ3:** Does consistency checking proposed by Dai et al. [9], indeed, beneficial. Is it applicable in the re-ranking setting (as opposed to using it with retrievers as in [9])?
- **RQ4:** Can we match performance of a large MonoT5-3B ranker using much smaller BERT models?
- **RQ5:** In particular, can we substantially outperform BM25 using a small and fast ranker such as a MiniLM ranker with merely 30 million parameters?

We obtain the following results:

- **RQ1 & RQ2**: Not only open-source LLMs BLOOM [44] and GPT-J [50] trained in a fully unsupervised fashion can be prompted to generate effective synthetic queries, but using them leads to consistent improvement compared to GPT-3 Curie model [6]. At the same time, we estimate that generation of synthetic queries using open-source models is about 10x cheaper. Note that in the concurrent work, Jeronymo et al. [18] also obtained good results using an open-source generation models, but they do not have an ablation that focuses specifically on replacing GPT-3 Curie with open-source models.
- **RQ3**: Although we confirm the effectiveness of the InPars training recipe, fine-tuning the model using consistency-checked data always produced a better model;
- **RQ4 & RQ5**: We confirm the finding of Bonifacio et al. [4] that the InPars-recipe (even if combined with consistency checking) does not work well for small models. We nevertheless discover that a combination of pre-training on all queries (from *all* datasets) and in-domain finetuning on consistency-checked data produces a small MiniLM-30M model that always outperformed BM25 by 7-30% in key metrics (nDCG@k or MRR);

2 RELATED WORK

Neural information retrieval has become a busy research area. An overview of the recent approaches and trends we address the reader to the survey by Lin et al. [24]. Likewise, prompting methods have gained quite a bit of popularity in NLP (see, e.g., [26] for a recent survey), but they were scarcely used in IR: We know only three papers directly related to our work.

Sachan et al. [43] were probably the first to demonstrate effectiveness of LLMs in the document ranking task. In their approach, which they named UPR, they concatenate a document, a special prompt such as "please write a question for this document" and the query itself. Then, UPR uses a pre-trained LLM model to compute the likelihood of generating the query given the passage text. However, they evaluate only on QA (but not IR) datasets and their main results are for an *impractical* three-billion parameter *instruction-finetuned* model, which is used essentially as a re-ranker (in a zero-shot sceanrio). Because this LLM was instruction fine-tuned these experiments do not permit definitive conclusions about **RQ1**. Although they also demonstrated the effectiveness of a standard (i.e., just next-token-prediction training) 2.7B GPT-Neo LLM [3], this was done only for a single QA dataset, namely Natural Questions [19].

The smallest model that they used had 250 million parameters (compared to our 30-million MiniLM model): The evaluated it *only* on the Natural Questions [19] collection, where it outperformed BM25 only by about 10%.

Bonifacio et al. [4] proposed an *InPars* method, which relied on few-shot prompting. The study had a convincing evaluation on five datasets where only one dataset, namely NQ, is a typical QA collection. Unlike Sachan et al. [43] Bonifacio et al. used few-shot prompting to distill an LLM into smaller rankers. To this end, few-shot prompting was used to generate synthetic queries for randomly sampled documents. For each collection they generated 100K synthetic queries and retained only 10K with the highest average log-probabilities.

However, they obtained good results only for a huge MonoT5-3B parameter model. Moreover, the used a proprietary GPT-3 model, which can be quite costly to use. In a follow-up study, which is concurrent with this work, Jeronymo et. al [18] introduced a modification of InPars—dubbed InPars v2—where GPT-3 Curie [6] was replaced with an open-source model GPT-J [50]. However, this model swap is “entangled” with at least two other modifications in the training recipe:

- A new filtering condition that employs an MS MARCO trained MonoT5-3B model.
- The vanilla prompt (which was used in InPars and our experiments) was replaced with the Guided by Bad Question prompt (introduced in [4]).

Thus, it is not possible to fairly assess the impact of replacing GPT-3 Curie with GPT-J [50].

An important disadvantage of InPars v2 recipe is that it is still not cost-effect as authors use a huge MonoT5-3B model. The filtering check uses a MonoT5-3B model trained on MS MARCO corpus, which is not always possible in a commercial setting due to licensing issues (MS MARCO is a research-only collection). Moreover, the MonoT5-3B model trained on MS MARCO—albeit being impractical—has excellent zero-shot transferability: Fine-tuning MonoT5-3B model trained on MS MARCO with InPars v2 only improves the average BEIR score only by 2.4%: from 0.538 to 0.551.

Dai et al. [9] used an InPars-like method called Promptagator and created synthetic training data using a huge proprietary FLAN model with 137 billion parameters. Although they used modestly sized models with 110 million parameters, Dai et al. [9] generated as many as million synthetic training queries for each dataset. In contrast, InPars used only 100K queries per dataset, which is much less expensive (see a discussion in § 5.2).

Importantly, Dai et al [9] proposed to use consistency checking [1] to filter-out potentially spurious queries, which was not previously done in the IR context. We use a variant of this procedure as well.

In addition, to prompt-based generation of training data, there are multiple proposals for self-supervised adaptation of out-of-domain models using generative pseudo-labeling [22, 38, 51]. To this end, questions or queries are generated using a pretrained seq2seq model (though an LLMs can be used as well) and negative examples are mined using either BM25 or an out-of-domain retriever or ranker. Unsupervised domain adaptation is complementary to the approaches considered in this work.

The disadvantage of such approaches is that they may need a reasonably effective out-of-domain model. However, such models

Table 1: The format of the vanilla 3-shot InPars prompt

Example 1:
Document: <text of the first **example** document>
Relevant Query: <text of the first relevant query>

Example 2:
Document: <text of the second **example** document>
Relevant Query: <text of the second relevant query>

Example 3:
Document: <text of the third **example** document>
Relevant Query: <text of the third relevant query>

Example 4:
Document: <**real in-domain** document text **placeholder**>
Relevant Query:

Notes: To generate a synthetic query, we first insert a text of a chosen real in-domain document after the prefix "Document:" in example four. Then, we "ask" an LLM to generate a completion.

can be hard to obtain due to licensing issues and poor transferability from other domains. For example, MS MARCO models have reasonable transferability [31, 47], but MS MARCO cannot be used to train models in a commercial context (without extra licensing from Microsoft). In contrast, the Natural Questions (NQ) collection [19] has a permissive license[2], but models trained on NQ can fail to transfer to datasets that are not based on Wikipedia [31].

Another potentially complementary approach is LLM-assisted query expansion. In particular Gao et al. prompt a 175B InstructGPT model to generate a hypothetical answer to a question [12]. Then this answer is encoded and together with the encoding of the original question they are compared with encoded documents. In a purely unsupervised setting—-using the Contriever bi-encoder training without supervision [17]—they were able to outperform BM25 by as much as 20%.

Despite strong results, a serious disadvantage of this approach is its dependence on the *external proprietary* model that is costly and has long generation times. Although we could not find any reliable benchmarks, a folklore opinion is that generation latency is a few seconds. To verify this, we used the OpenAI playground[3] to generate a few hypothetical answers using the prompt in Gao et al. [12] and a sample of TREC DL 2020 queries. With a maximum generation length of 256 tokens (a default setting), the latency exceeded four seconds.

Quite interestingly, Gao et al. [12] tried to replace a 175B GPT-3 model with smaller open-source models on TREC DL 2019 and TREC DL 2020 (see Table 4 in their study), but failed to obtain consistent and substantial gains with models having fewer than 50B parameters.

3 METHODS

3.1 Information Retrieval Pipeline

We use a variant of a classic filter-and-refine multi-stage retrieval pipeline [30, 36, 52], where top-k candidate documents retrieved by a fast BM25 ranker [40] are re-ranked using a slower neural re-ranker. For collections where document have titles, the BM25 retriever itself has two stages: In the first stage we retrieve 1K documents using a Lucene index built over the concatenation of all document fields. In the second stage, these candidates are re-ranked using equally weighted BM25 scores computed separately for each field.

Our neural rankers are cross-encoder models [24, 34], which operate on queries concatenated with documents. Concatenated texts are passed to a backbone bi-directional *encoder-only* Transformer model [10] equipped with an additional ranking head (a fully-connected layer), which produces a relevance score (using the last-layer contextualized embedding of a CLS-token [34]). In contrast, authors of InPars [4] use a T5 [37] cross-encoding re-ranker [35], which is a *full* Transformer model [48]. It uses both the encoder and the decoder. The T5 ranking Transformer is trained to generate labels "true" and "false", which represent relevant and non-relevant document-query pairs, respectively.

Backbone Transformer models can differ in the number of parameters and pre-training approaches (including pre-training datasets). In this paper we evaluated the following models, all of which were pre-trained in the self-supervised fashion without using IR-specific pre-training or supervision with IR datasets:

- A six-layer MiniLM-L6 model [53]. It is a tiny (by modern standards) 30-million parameter model, which was distilled [15, 21, 41] from Roberta [27]. We download model L6xH384 MiniLMv2 from the Microsoft website.[4]
- A 24-layer (large) ERNIE v2 model from the HuggingFace hub [46][5]. It has 335 million parameters.
- A 24-layer (large) DeBERTA v3 model with 435 million parameters [14] from the HuggingFace hub [6].

We choose ERNIE v2 and DeBERTA v3 because (from our prior experience) we knew they had strong performance on the MS MARCO dataset (better than BERT large [10] and several other models that we tested in the past). Both models performed comparably well (see Table 4) in the preliminary experiments, but we chose DeBERTA for main experiments because it is more effective on MS MARCO and TREC-DL 2020.

However, both of these models are quite large and we aspired to show that an InPars-like training recipe can be used with smaller models too. In contrast, Bonifacio et al. [4] were able to show that *only* a big T5-3B model with 3B parameters outperformed BM25 on all five datasets, while the smaller (but still quite large) T5-200M ranker with "merely" 200 million parameters did not outperform BM25 on three datasets (it only worked on MS MARCO and TREC-DL 2020).

[2]https://github.com/google-research-datasets/natural-questions/blob/master/LICENSE
[3]https://beta.openai.com/playground
[4]https://github.com/microsoft/unilm/tree/master/minilm
[5]https://huggingface.co/nghuyong/ernie-2.0-large-en
[6]https://huggingface.co/microsoft/deberta-v3-large

3.2 Generation of Synthetic Training Data

In-domain training data is generated using a well-known few-shot prompting approach introduced by Brown et al. [6]. In the IR domain, it was first used by Bonifacio et al. [4] who re-branded it as *InPars*. The key idea is to "prompt" a large language model with a few-shot textual demonstration of known relevant query-document pairs. To produce a novel query-document pair, Bonifacio et al. append an in-domain document to the prompt and "ask" the model to complete the text. We use the so-called vanilla prompt (see Table 1) created by Bonifacio et al. [4]. Like in the InPars study, we generated 100K queries for each dataset with exception of MS MARCO and TREC DL. Because both datasets use the same set of passages they share the same set of 100K generated queries.

Repeating this procedure for many in-domain documents produces a large training set, but it can be quite imperfect: We carried out spot-checking and found quite a few queries that were spurious or only tangentially relevant to the passage from which they were generated.

Many spurious queries can be filtered out automatically. To this end, in InPars Bonifacio et al. [4] used only 10% of the queries with the highest log-probabilities (averaged over query tokens). In Promptagator Dai et al. [9] introduced a different filtering procedure, which was a variant of consistency checking [1]. Dai et al. first trained a retriever model using all the generated queries. Then, for each query they retrieved a set of documents. The query passed the consistency check if the first retrieved document was the document from which the query was generated.

Dai et al. [9] used consistency checking with bi-encoding retrieval models, but it is applicable to cross-encoding re-ranking models as well. Another straightforward modification of this approach is to check if a generated document is present in a set of top-k ($k > 1$) candidates with the highest relevance scores (as computed by the re-ranking or retrieval model).

3.3 InPars-Light Training Recipe

The InPars-Light is a modification of the original InPars, but it is substantially more cost effective for generation of synthetic queries, training the models, and inference. InPars-Light has the following main "ingredients":

- Using open-source models instead of GPT-3;
- Using much smaller ranking models;
- Fine-tuning models on consistency-checked training data;
- Optional pre-training of models using *all* generated queries from *all* collections.
- Re-ranking only 100 candidate documents instead of 1000;

To obtain consistency-checked queries for a given dataset, a model trained on InPars-generated queries (for this dataset) was used to re-rank output of *all* original queries (for a given dataset). Then, all the queries where the query-generating-document did *not* appear among top-k scored documents was discarded. In our study, we experimented with k from one to three (but *only* on MS MARCO).[7] Although $k = 1$ worked pretty well, using $k = 3$ lead to a small boost in accuracy. Consistency-checking was carried out using DeBERTA-v3-435M [14]. We want to emphasize that consistency-checked training data was used *in addition* to original InPars-generated data (but not instead), namely, to fine-tune a model initially trained on InPars generated data.

Also, quite interestingly, a set of consistency-checked queries had only a modest (about 20-30%) overlap with the set of queries that were selected according to their average log-probabilities. Thus, consistency-checking increased the amount of available training data. It might seem appealing to achieve the same objective by simply picking a larger number of queries (with highest average log-probabilities). However, preliminary experiments on MS MARCO showed that a naive increase of the number of queries degraded effectiveness (which is consistent with the InPars study [4]).

Although, the original InPars recipe with open-source models and consistency checking allowed us to train strong DeBERTA-v3-435M models, performance of MiniLM models was lackluster (roughly at BM25 level for all collections).

Because bigger models performed quite well, it may be possible to distill [15, 21, 41] their parameters into a much smaller MiniLM-30M model. Distillation is known to be successful in the IR domain [16, 25]. However, it failed in our case due to overfitting. We left investigation of the failure reasons for the future and used the following workaround:

- First we carried out an *all-domain* pre-training *without any filtering* (i.e., using *all* queries from *all* collections);
- Then, we fine-tuned all-domain pre-trained models on the consistency-checked in-domain data.

3.4 Miscellaneous

Experiments were carried out using the framework FlexNeuART [5], which provided support for basic indexing, retrieval, and neural ranking. The neural models are implemented using PyTorch and Huggingface [54]. The models were trained using an InfoNCE loss [20]. In a single training epoch, we selected randomly one pair of positive and three negative examples per query (negatives are sampled from 1000 documents with highest BM25 scores). Note that, however, that during inference we re-ranked only 100 documents. A number of negatives was not tuned: We used as much as we can while ensuring we do not run out of GPU memory during training on any collection.

We used an AdamW [28] optimizer with a small weight decay (10^{-7}), a warm-up schedule, and a batch size of 16.[8] We used different base rates for the fully-connected prediction head ($2 \cdot 10^{-4}$) and for the main Transformer layers ($2 \cdot 10^{-5}$). Also note that the loss reduction mode was "sum": To use this recipe with the reduction mode "mean" the learning rates need to be multiplied by the batch size.

We trained each model using three seeds and reported the *average results* (unless specified otherwise). To compute statistical significance, we first obtained an "average" run where for each query we averaged query-specific metric values over three seeds. Note that we compute exactly the same metric values as in [4]. For statistical significance testing we used a paired two-sided t-test. For query sets with a large number of queries (name MS MARCO development set and BEIR Natural Questions) we used a lower threshold

[7] We did not want to optimize this parameter for all collections and, thus, to commit a sin of tuning hyper-parameters on the complete test set.

[8] The learning rate grows linearly from zero for 20% of the steps until it reaches the base learning rate [32, 45] and then goes back to zero (also linearly).

of 0.01. For small query sets (Robust04 and TREC-COVID), the statistical significance threshold was set to 0.05.

We implemented our query generation module using the AutoModelForCasualLM interface from HuggingFace. We used a three-shot vanilla prompt template used by [4] (also shown in Table 1). The output was generated via greedy decoding. The maximum number of new tokens generated for each example was set to 32.

4 DATASETS

Because we aimed to reproduce the main results of InPars [4], we used exactly the same set of queries and datasets, which are described below. Except MS MARCO (which was processed directly using FlexNeuART [5] scripts), datasets were ingested with a help of the IR datasets packages [29].

Some of the collections below have multiple text fields, which were used differently by a BM25 and neural ranker. All collections except Robust04 have exactly one query field. Robust04 queries have the following parts: title, description, and narrative. For the purpose of BM25 retrieval and ranker, we use only the title field, but the neural ranker is used the description field (which is consistent with [4]). The narrative field is not used.

Two collections have documents with both the title and the main body text fields. The neural rankers operate on concatenation of these fields. If this concatenation is longer than 477 BERT tokens, it is truncated (queries longer than 32 BERT tokens are truncated as well). For BM25 scoring, we index the concatenated fields as well (in Lucene). However, after retrieving 1000 candidates, we re-rank them using the sum of BM25 scores computed for the title and the main body text fields separately (using FlexNeuART [5]).

Synthetically Generated Training Queries. For each of the datasets, Bonifacio et al. [4] provided both the GPT-3-generated queries (using GPT-3 Curie model) and the documents that are used to generate the queries. This permits an apples-to-apples comparison of the quality of training data generated using GPT-3 Curie with the quality of synthetic training data generated using open-source models GPT-J [50] and BLOOM [44]. According to the estimates of Bonifacio et al. [4], the Curie model has 6B parameters, which is close to the estimate made by by Gao from EleutherAI [11]. Thus, we used GPT-J [50] and BLOOM [44] models with 6 and 7 billion parameters, respectively. Although other open-source models can potentially be used, generation of synthetic queries is quite expensive and exploring other open-source options is left for future work.

MS MARCO sparse and TREC DL 2020. MS MARCO is collection of 8.8M passages extracted from approximately 3.6M Web documents, which was derived from the MS MARCO reading comprehension dataset [2, 8]. It “ships“ with more than half a million of question-like queries sampled from the Bing search engine log with subsequent filtering. The queries are not necessarily proper English questions, e.g., “lyme disease symptoms mood”, but they are answerable by a short passage retrieved from a set of about 3.6M Web documents [2]. Relevance judgements are quite sparse (about one relevant passage per query) and a positive label indicates that the passage can answer the respective question.

The MS MARCO collections has several development and test query sets of which we use only a development set with approximately 6.9K sparsely-judged queries and the TREC DL 2020 [8] collection of 54 densely judged queries. Henceforth, for simplicity when we discuss the MS MARCO development set we use a shortened name MS MARCO, which is also consistent with Bonifacio al. [4].

Note that the MS MARCO collection has a large training set, but we do not use it in the fully unsupervised scenario. We do use it though in the hybrid setting (see § 5.1).

Robust04 [49] is a small (but commonly used) collection that has about 500K news wire documents. It comes with a small but densely judged set of 250 queries, which have about 1.2K judgements on average.

Natural Questions (NQ) BEIR [19] is an open domain Wikipedia-based Question Answering (QA) dataset. Similar to MS MARCO, it has real user queries (submitted to Google). We use a BEIR’s variant of NQ [47], which has about 2.6M short passages from Wikipedia and 3.4K sparsely-judged queries (about 1.2 relevant documents per query).

TREC COVID BEIR [39] is a small corpus that has 171K scientific articles on the topic of COVID-19 and 50 densely-judged queries (1.3K judged documents per query on average). It was created for a NIST challenge whose objective was to develop information retrieval methods tailored for the COVID-19 domain (with a hope to be a useful tool during COVID-19 pandemic). We use the BEIR’s version of this dataset [47].

5 RESULTS

5.1 Main Results

Our main experimental results are presented in Table 2. We organize them into multiple sections, where we show effectiveness numbers for various supervised, unsupervised, and hybrid approaches. In addition to our own measurements, we copy key results from the work by Bonifacio et al. [4], which include results for BM25, re-ranking using OpenAI API, as well as results for Mono-T5 rankers [35] trained in the unsupervised, supervised, and hybrid manner. Because there is a substantial variability of results among seeds (including one case of extremely poor convergence), for unsupervised-only training we also present our-best seed results in Table 3. In Table 4, we show effectiveness of InPars for three types of models to generate synthetic training queries (including OpenAI GPT-3 model). In our experiments, we statistically test several statistical hypotheses, which are explained separately at the bottom of each table.

BM25 baselines. Comparing effectiveness of FlexNeuART [5] BM25 with effectiveness of Pyserini [23] BM25—used the InPars study [4]—we can see that on all datasets except TREC DL 2020 we closely match (within 1.5%) Pyserini numbers. On TREC DL 2020 our BM25 is 6% more effective in nNDCG@10 and 25% more effective in MAP.

Reproducibility Notes. In addition to BM25 performance, we can reproduce some of the key findings from prior work. In what follows we discuss these (and other findings) in more detail:

Table 2: Main Results (averaged over three seeds)

	MS MARCO	TREC DL 2020		Robust04		NQ	TREC COVID
	MRR	MAP	nDCG@10	MAP	nDCG@20	nDCG@10	nDCG@10
BM25 [4]	0.1874	0.2876	0.4876	0.2531	0.4240	0.3290	0.6880
BM25 (ours)	0.1867	0.3612	0.5159	0.2555	0.4285	0.3248	0.6767
OpenAI Ranking API: re-ranking 100 Documents [4]							
Curie (6B) [4]	$	0.3296	0.5422	0.2785	0.5053	0.4171	0.7251
Davinci (175B) [4]	$	0.3163	0.5366	0.2790	0.5103	$	0.6918
Unsupervised: InPars-based Training Data							
monoT5-220M (InPars) [4]	0.2585	0.3599	0.5764	0.2490	0.4268	0.3354	0.6666
monoT5-3B (InPars) [4]	**0.2967**	0.4334	0.6612	**0.3180**	0.5181	**0.5133**	0.7835
MiniLM-L6-30M (InPars)	^{ba}0.2117	^{b}0.3482	^{b}0.4953	^{ba}0.2263	^{ba}0.3802	^{ba}0.2187	^{b}0.6361
MiniLM-L6-30M (InPars ► consist. check)	^{cba}0.2336	^{cb}0.3769	^{cb}0.5543	^{cb}0.2556	^{cb}0.4440	^{cb}0.3239	^{cb}0.6926
MiniLM-L6-30M (InPars **all** ► consist. check)	^{ca}0.2468	^{ca}0.3929	^{ca}0.5726	^{c}0.2639	^{ca}0.4599	^{ca}0.3747	^{ca}0.7688
DeBERTA-v3-435M (InPars)	^{ba}0.2746	^{ba}0.4385	^{a}0.6649	^{ba}0.2811	^{ba}0.4987	^{ba}0.4476	^{a}0.8022
DeBERTA-v3-435M (InPars ► consist. check)	^{cba}0.2815	^{cba}**0.4446**	^{ca}**0.6717**	^{cba}0.3009	^{cba}**0.5360**	^{cba}0.4621	^{ca}**0.8183**
DeBERTA-v3-435M (InPars **all** ► consist. check)	^{ca}0.1957	^{c}0.3607	^{c}0.5007	^{c}0.2518	^{c}0.4320	^{c}0.3267	^{c}0.6953
Supervised and Hybrid: transfer from MS MARCO with an optional fine-tuning on consistency-checked InPars data							
MiniLM-L6-30M (MS MARCO)	^{da}0.3080	^{a}0.4370	^{a}0.6662	^{da}0.2295	^{da}0.3923	^{da}0.4646	^{da}0.7476
MiniLM-L6-30M (MS MARCO ► consist. check)	^{da}0.2944	^{a}0.4311	^{a}0.6501	^{da}0.2692	^{da}0.4730	^{da}0.4320	^{da}0.7898
DeBERTA-v3-435M (MS MARCO)	^{da}0.3508	^{a}0.4679	^{da}0.7269	^{a}0.2986	^{a}0.5304	^{da}0.5616	^{a}0.8304
DeBERTA-v3-435M (MS MARCO ► consist. check)	^{da}0.3166	^{a}0.4553	^{da}0.6912	^{a}0.3011	^{a}0.5371	^{da}0.5075	^{a}0.8165
monoT5-220M (MS MARCO) [35]	0.3810	0.4909	0.7141	0.3279	0.5298	0.5674	0.7775
monoT5-3B (MS MARCO) [35]	**0.3980**	**0.5281**	**0.7508**	0.3876	0.6091	**0.6334**	0.7948
monoT5-3B (MS MARCO ► InPars) [4]	0.3894	0.5087	0.7439	**0.3967**	**0.6227**	0.6297	**0.8471**

OpenAI API ranking results are copied from Bonifacio et al. [4]. In that, $ denotes experiments that were too expensive to run.
InPars denotes the original query-generation method with filtering-out 90% of queries having lowest average log-probabilities.
InPars all denotes the query-generation method without query filtering, which was used in *all-domain* pretraining.
Consist. check denotes consistency filtering of *all* generated queries using a model trained on InPars-generated data.
Best unsupervised and hybrid-training results are marked by **bold** font (separately for unsupervised and hybrid-training).
Super-scripted labels denote the following statistically significant differences (thresholds are given in the main text):
a: between a neural model and BM25;
b: between training with and without fine-tuning on consistency-checked data (for the **same** model type).
c: between pre-training using *all* generated queries for *all* collections and *only* filtered *in-domain* queries (for the **same** model type).
d: between 0-shot transferring an MS MARCO model & fine-tuning this model on filtered *in-domain* queries (for the **same** model type).

- Generation of synthetic in-domain data using an InPars-like recipe can permit training very strong in-domain rankers without any human-provided supervision data;
- Without additional tricks such as all-domain pre-training and consistency checking, only a sufficiently larger model can outperform BM25;
- A consistency checking introduced by Promptagator [9] does lead to a substantial gain in accuracy;
- Replacing the proprietary GPT-3 Curie model with BLOOM [44] can improve performance, which is in line with findings of Jeronymo et al. [18]. However, unlike our study, they do not directly assess the impact of replacing the generating model alone.

Unsupervised-only training. In a purely unsupervised setting, we obtain comparable or better results using much smaller ranking models. With DeBERTA-v3-435M we obtain comparable (somewhat better or worse) effectiveness to MonoT5-3B on four collections out of five (MonoT5-3B has 7x more parameters). Our biggest gap is for the NQ collection. However, this is the collection where we already obtain a substantial 15-30% gain over BM25. Moreover, there is quite a bit of variability across model seeds, and our best-seed NQ model is quite close to the average performance of MonoT5-3B (see Table 3).

Our smallest MiniLM-L6-30M model with all-domain pretraining and finetuning on consistency-checked data (InPars **all** ► consist. check) roughly matches the 7x larger MonoT5-220M on MS MARCO and TREC DL 2020, but it is substantially better than

Table 3: Best-Seed Results for Unsupervised Training

	MS MARCO	TREC DL 2020		Robust04		NQ	TREC COVID
	MRR	MAP	nDCG@10	MAP	nDCG@20	nDCG@10	nDCG@10
BM25 (ours)	0.1867	0.3612	0.5159	0.2555	0.4285	0.3248	0.6767
MiniLM results							
MiniLM-L6-30M (InPars)	^{ba}0.2197	^{b}0.3562	^{b}0.5151	^{ba}0.2380	^{ba}0.4029	^{ba}0.2415	^{b}0.6732
MiniLM-L6-30M (InPars ► consist. check)	^{cba}0.2422	^{b}0.3844	^{ba}0.5753	^{cb}0.2615	^{cba}0.4554	^{cb}0.3297	^{ba}0.7483
MiniLM-L6-30M (InPars **all** ► consist. check)	^{ca}**0.2517**	^{a}**0.3945**	^{a}**0.5769**	^{c}**0.2671**	^{ca}**0.4691**	^{ca}**0.3800**	^{a}**0.7709**
DeBERTA results							
DeBERTA-v3-435M (InPars)	^{ba}0.2748	^{a}0.4437	^{a}0.6779	^{ba}0.2874	^{ba}0.5131	^{a}0.4872	^{a}0.8118
DeBERTA-v3-435M (InPars ► consist. check)	^{ba}**0.2847**	^{a}**0.4479**	^{a}**0.6813**	^{ba}0.3043	^{ba}0.5417	^{ca}**0.4924**	^{a}**0.8305**
DeBERTA-v3-435M (InPars **all** ► consist. check)	^{a}0.2804	^{a}0.4414	^{a}0.6575	^{a}**0.3076**	^{a}**0.5505**	^{ca}0.4746	^{a}0.8259

Super-scripted labels denote the following statistically significant differences (thresholds are given in the main text):
a: between a neural model and BM25;
b: between training with and without fine-tuning on consistency-checked data (for the **same** model type).
c: between pre-training using all generated queries and *only* filtered *in-domain* queries (for the **same** model type).
Notes: Best results (**separately** for each model are marked by **bold** font.

MonoT5-220M on the remaining datasets, where MonoT5-220M effectiveness is largely at BM25 level. MiniLM-L6-30M outperforms BM25 on all collections and all metrics. In all but one case these differences are also *statistically significant*. In terms of nDCG and/or MRR, it is 7-30% more effective.

Impact of consistency checking and all-domain pre-training. It is crucial to note, however, on its own the InPars recipe does not produce a strong MiniLM-L6-30M model, which is in line with finding of the InPars study where only MonoT5-3B (but not a much smaller MonoT5-220M) outperformed BM25 on all collections. Strong performance of MiniLM-L6-30M was due to additional training with consistency-checked data and pre-training on all-domain (all queries from all collections) data. Therefore, we carry out ablation experiments to assess effectiveness of these procedures.

We can see that for both MiniLM-L6-30M and DeBERTA-v3-435M, fine-tunining on consistency-checked data improves outcomes: For 12 measurements out of 14, these differences are also statistically significant (denoted by super-script label "b"). Moreover, all-domain pretraining (instead of training on data generated by the original InPars recipe) further boosts accuracy of MiniLM-L6-30M in all cases. Moreover, all the differences are statistically significant (denoted by super-script label "c"). However, all-domain pretraining substantially degrades performance of DeBERTA-v3-435M.

An in-depth investigation showed that for one seed (out of three), the model failed to converge properly. Although, we could have also chosen a different seed and present better results, we felt that this failure to converge (which was the only case out of many experiments we ran!) was an indication that all-domain pretraining dit not work well for DeBERTA-v3-435M. To further verify this hypothesis, we also checked the best-seed outcomes, which are presented in Table 3. For MiniLM-L6-30M, the all-domain pre-training improves the best-seed results in all cases, though we have fewer statistically significant difference now (this is expected, because using average-seed runs leads to more stable measurements). For DeBERTA-v3-435M, there is either a substantial degradation or a small decrease/increase that is not statistically significant (denoted by super-script label "c"). Thus, our biggest model—unlike 15x smaller MiniLM-L6-30M— does not benefit from all-domain pretraining. In fact this pretraining leads to performance degradation (including potential decrease in training stability).

Supervised and hybrid training. We find that for our datasets, a model trained on MS MARCO (both MiniLM-L6-30M and DeBERTA-v3-435M) transfers well to other collections, except for transfer of MiniLM-L6-30M to Robust04. However, similar to all-domain pre-training the 15x smaller MiniLM-L6-30M benefits more from in-domain fine-tuning with consistency-checked data: There are substantial and statistically significant improvements for Robust04 and TREC-COVID (but a degradation for MS MARCO, TREC DL 2020 and NQ, whereas in the case of DeBERTA-v3-435M such fine-tuning noticeably degrades accuracy in most cases.

Also note that DeBERTA-v3-435M roughly matches MonoT5-220M while still lagging behind MonoT5-3B. This is in line with prior finding that large ranking models have better zero-shot transferring effectiveness [33, 42]. However, using multi-billion parameter ranking models is not a practical choice.

Model-Type Ablation To assess the impact of replacing GPT-3 Curie with an open-source model, we carried out experiments using ERNIE-v2 model [46]. Although we generated synthetic queries only once, each ranker was trained with three different seeds. Thus, we compared systems where query-specific metric values were averaged over seeds. To our surprise (see Table 4), except for NQ where all models were nearly equally good, GPT-3 Curie underperformed both open-source models (out of 14 measurements 10 are statistically significant as denoted by super-script "b"). The difference was particularly big for Robust04.

We then computed the average gain by (1) computing relative gain separately for each datasets and key metrics (nDCG or MRR)

Table 4: Performance of InPars for Different Generating and Ranking Models (averaged over three seeds)

	MS MARCO	TREC DL 2020		Robust04		NQ	TREC COVID
	MRR	MAP	nDCG@10	MAP	nDCG@20	nDCG@10	nDCG@10
BM25 (ours)	0.1867	0.3612	0.5159	0.2555	0.4285	0.3248	0.6767
ERNIE-v2-335M OpenAI Curie (6B)	^{a}0.2538	^{a}0.4140	^{a}0.6229	^{a}0.2357	0.4016	^{a}0.4277	^{a}0.7411
ERNIE-v2-335M GPT-J (6B)	^{ba}0.2608	^{ba}0.4286	^{a}0.6367	^{cb}0.2691	^{cba}0.4724	^{a}0.4248	^{ba}0.7750
ERNIE-v2-335M BLOOM (7B)	^{dba}0.2605	^{ba}0.4286	^{a}0.6407	^{cba}**0.2852**	^{dcba}**0.5102**	^{da}0.4215	^{ba}0.7871
DeBERTA-v3-435M BLOOM (7B)	^{dba}**0.2746**	^{ba}**0.4385**	^{ba}**0.6649**	^{ba}0.2811	^{dba}0.4987	^{dba}**0.4476**	^{ba}**0.8022**

Notes: Best results are in bold. Super-scripted labels denote statistically significant differences (thresholds are given in the main text):
a: between a neural model and BM25;
b: between a given neural model trained using queries from a given open-source models and ERNIE trained on queries from GPT-3;
c: between using GPT-J-generated queries and BLOOM-generated queries (only for ERNIE);
d: between the DeBERTA model and the ERNIE model (both trained using BLOOM-generated queries).

and (2) averaging these relative gains. The resulting gains (not shown in the table) are 7.2% for BLOOM and 5.2% for GPT-J.

In addition to the generation model, we assessed the impact of using DeBERTA-v3 instead of ERNIE-v2. This time around, both models were trained using BLOOM-generated queries. We can see that DeBERTA v3 was generally better than ERNIE-v2.

5.2 Cost and Efficiency

In the following sub-section, we discuss both the ranking efficiency and query-generation cost. Although one may argue that the cost of generation using open-source models is negligibly small, in reality this is true only if one owns their own hardware and generates enough queries to justify the initial investment. Thus, we make a more reasonable assessment assuming that the user can employ a cheap cloud service.

Efficiency of Re-ranking. A rather common opinion (in particular expressed by anonymous reviewers on multiple occasions) is that using cross-encoders is not a practical option. This might be true for extremely constrained latency environments or very large models, but we think it is totally practical to use small models such as MiniLM-L6-30M for applications such as enterprise search. In particular, on a reasonably modern GPU (such as RTX 3090) and Pytorch MinLm-L6-30M re-ranking throughput exceeds 500 passages per second (assuming truncation to the first 477 characters). Thus re-ranking 100 documents has an acceptable sub-second reranking latency.

Cost of Model Training. Here, all training times are given with respect to a single RTX 3090 GPU. Training and evaluating MiniLM6-30M models had *negligible* costs dominated by all-domain pretraining, which took about two hours per seed. In contrast, the all-domain pretraining of DeBERTA-v3-435M took 28 hours. However, only about 20-30% of queries were selected for training models and fine-tuning them consistency checked data. Thus, without all-domain pretraining, the training time itself was rather small.

Aside from all-domain pre-training, the two most time-consuming operations were validation of large query sets (MS MARCO and NQ), which jointly have about 10K queries, and consistency checking (using DeBERTA-v3-435M model). The total validation time for DeBERTA-v3-435 was about 6 hours (for all collections). The consistency checking, however, took about 48 hours. In the future, we should consider carrying out consistency checking using a much faster MiniLM-L6-30M model.

Cost of Query Generation. For the original InPars [4], the cost of generation for the GPT-3 Curie model is $0.002 per one thousand tokens. The token count includes the length of the prompt and the prompting document.[9] We estimate that (depending on the collection) a single generation involves 300 to 500 tokens: long-document collections Robust04 and TREC-COVID both have close to 500 tokens per generation.

Taking an estimate of 500 tokens per generation, the cost of querying OpenAI GPT-3 Curie API can be up to $100 for Robust04 and TREC-COVID. Assuming that sampling from 137-B FLAN model to be as expensive as from the largest GPT-3 model Davinci (which has a similar number of parameters), each generation in the Promptagator study [9], was 10x more expensive compared to InPars study [4]. Moreover, because Dai et al. [4] generated one million samples per collection, the Promptagator recipe was about *two orders* of magnitude expensive compared to InPars.

In contrast, it takes only about 15 hours to generate 100K queries using RTX 3090 GPU. Extrapolating this estimate to A100, which is about 2x faster than RTX 3090[10], and using the pricing of Lambda GPU cloud, we estimate the cost of generation in our InPars-light study to be under $10 per collection. [11]

6 CONCLUSION

We carried out a reproducibility study of InPars recipe for unsupervised training of neural rankers. As a by-product of this study, we developed a simple-yet-effective modification of InPars, which we called InPars-light. Unlike InPars, InPars-light uses only a freely available language model BLOOM, 7x-100x smaller ranking models, and re-ranks only 100 candidate records instead of 1000.

Not only can we reproduce key findings from prior work, but combining the original InPars recipe [4] with (1) fine-tuning on consistency-checked data [9], (2) and all-domain pretraining, we

[9] https://chengh.medium.com/understand-the-pricing-of-gpt3-e646b2d63320
[10] https://lambdalabs.com/blog/nvidia-rtx-a6000-benchmarks
[11] https://lambdalabs.com/service/gpu-cloud#pricing

were able to train a very efficient and small model MiniLM-L6-30M that outperformed BM25 on all collections (in MRR or nDCG). Last but not least, with a larger DeBERTA-v3-435M model we could largely match performance of a 7x larger MonoT5-3B (even outperforming it on two datasets).

ACKNOWLEDGMENTS

REFERENCES

[1] Chris Alberti, Daniel Andor, Emily Pitler, Jacob Devlin, and Michael Collins. 2019. Synthetic QA Corpora Generation with Roundtrip Consistency. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Florence, Italy, 6168–6173. https://doi.org/10.18653/v1/P19-1620

[2] 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).

[3] Sid Black, Leo Gao, Phil Wang, Connor Leahy, and Stella Biderman. 2021. *GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow*. https://doi.org/10.5281/zenodo.5297715

[4] Luiz Henrique Bonifacio, Hugo Abonizio, Marzieh Fadaee, and Rodrigo Nogueira. 2022. InPars: Unsupervised Dataset Generation for Information Retrieval. In *SIGIR*. ACM, 2387–2392.

[5] Leonid Boytsov and Eric Nyberg. 2020. Flexible retrieval with NMSLIB and FlexNeuART. In *Proceedings of Second Workshop for NLP Open Source Software (NLP-OSS)*. 32–43.

[6] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. In *NeurIPS*.

[7] Chris Buckley, Darrin Dimmick, Ian Soboroff, and Ellen M. Voorhees. 2007. Bias and the limits of pooling for large collections. *Inf. Retr.* 10, 6 (2007), 491–508.

[8] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen M Voorhees. 2020. Overview of the TREC 2019 deep learning track. *arXiv preprint arXiv:2003.07820* (2020).

[9] Zhuyun Dai, Vincent Y. Zhao, Ji Ma, Yi Luan, Jianmo Ni, Jing Lu, Anton Bakalov, Kelvin Guu, Keith B. Hall, and Ming-Wei Chang. 2022. Promptagator: Few-shot Dense Retrieval From 8 Examples. *CoRR* abs/2209.11755 (2022).

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[11] Leo Gao. 2021. https://blog.eleuther.ai/gpt3-model-sizes/.

[12] Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. 2022. Precise Zero-Shot Dense Retrieval without Relevance Labels. https://doi.org/10.48550/ARXIV.2212.10496

[13] Lei Han, Eddy Maddalena, Alessandro Checco, Cristina Sarasua, Ujwal Gadiraju, Kevin Roitero, and Gianluca Demartini. 2020. Crowd Worker Strategies in Relevance Judgment Tasks. In *WSDM*. ACM, 241–249.

[14] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. Deberta: decoding-Enhanced Bert with Disentangled Attention. In *ICLR*. OpenReview.net.

[15] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the Knowledge in a Neural Network. *CoRR* abs/1503.02531 (2015).

[16] Sebastian Hofstätter, Sophia Althammer, Michael Schröder, Mete Sertkan, and Allan Hanbury. 2020. Improving Efficient Neural Ranking Models with Cross-Architecture Knowledge Distillation. https://doi.org/10.48550/ARXIV.2010.02666

[17] Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2021. Towards Unsupervised Dense Information Retrieval with Contrastive Learning. *CoRR* abs/2112.09118 (2021).

[18] Vitor Jeronymo, Luiz Bonifacio, Hugo Abonizio, Marzieh Fadaee, Roberto Lotufo, Jakub Zavrel, and Rodrigo Nogueira. 2023. InPars-v2: Large Language Models as Efficient Dataset Generators for Information Retrieval. https://doi.org/10.48550/ARXIV.2301.01820

[19] 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.

[20] Phuc H. Le-Khac, Graham Healy, and Alan F. Smeaton. 2020. Contrastive Representation Learning: A Framework and Review. *IEEE Access* 8 (2020), 193907–193934.

[21] Jinyu Li, Rui Zhao, Jui-Ting Huang, and Yifan Gong. 2014. Learning small-size DNN with output-distribution-based criteria. In *INTERSPEECH*. ISCA, 1910–1914.

[22] Minghan Li and Éric Gaussier. 2022. Domain Adaptation for Dense Retrieval through Self-Supervision by Pseudo-Relevance Labeling. *CoRR* abs/2212.06552 (2022).

[23] Jimmy Lin, Xueguang Ma, Sheng-Chieh Lin, Jheng-Hong Yang, Ronak Pradeep, and Rodrigo Nogueira. 2021. Pyserini: A Python toolkit for reproducible information retrieval research with sparse and dense representations. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2356–2362.

[24] Jimmy Lin, Rodrigo Nogueira, and Andrew Yates. 2021. *Pretrained Transformers for Text Ranking: BERT and Beyond*. Morgan & Claypool Publishers.

[25] Sheng-Chieh Lin, Jheng-Hong Yang, and Jimmy Lin. 2020. Distilling Dense Representations for Ranking using Tightly-Coupled Teachers. *CoRR* abs/2010.11386 (2020).

[26] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021. Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing. *CoRR* abs/2107.13586 (2021).

[27] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *CoRR* abs/1907.11692 (2019).

[28] Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101* (2017).

[29] Sean MacAvaney, Andrew Yates, Sergey Feldman, Doug Downey, Arman Cohan, and Nazli Goharian. 2021. Simplified Data Wrangling with ir_datasets. In *SIGIR*.

[30] Irina Matveeva, Chris Burges, Timo Burkard, Andy Laucius, and Leon Wong. 2006. High accuracy retrieval with multiple nested ranker. In *SIGIR*. ACM, 437–444.

[31] Iurii Mokrii, Leonid Boytsov, and Pavel Braslavski. 2021. A Systematic Evaluation of Transfer Learning and Pseudo-labeling with BERT-based Ranking Models. In *SIGIR*. ACM, 2081–2085.

[32] Marius Mosbach, Maksym Andriushchenko, and Dietrich Klakow. 2020. On the Stability of Fine-tuning BERT: Misconceptions, Explanations, and Strong Baselines. *arXiv preprint arXiv:2006.04884* (2020).

[33] Jianmo Ni, Chen Qu, Jing Lu, Zhuyun Dai, Gustavo Hern'andez 'Abrego, Ji Ma, Vincent Zhao, Yi Luan, Keith B. Hall, Ming-Wei Chang, and Yinfei Yang. 2021. Large Dual Encoders Are Generalizable Retrievers. *ArXiv* abs/2112.07899 (2021).

[34] Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage Re-ranking with BERT. *arXiv preprint arXiv:1901.04085* (2019).

[35] Rodrigo Nogueira, Zhiying Jiang, and Jimmy Lin. 2020. Document ranking with a pretrained sequence-to-sequence model. *arXiv preprint arXiv:2003.06713* (2020).

[36] John M. Prager. 2006. Open-Domain Question-Answering. *Found. Trends Inf. Retr.* 1, 2 (2006), 91–231.

[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. *J. Mach. Learn. Res.* 21 (2020), 140:1–140:67.

[38] Revanth Gangi Reddy, Bhavani Iyer, Md. Arafat Sultan, Rong Zhang, Avirup Sil, Vittorio Castelli, Radu Florian, and Salim Roukos. 2021. Synthetic Target Domain Supervision for Open Retrieval QA. In *SIGIR*. ACM, 1793–1797.

[39] Kirk Roberts, Tasmeer Alam, Steven Bedrick, Dina Demner-Fushman, Kyle Lo, Ian Soboroff, Ellen M. Voorhees, Lucy Lu Wang, and William R. Hersh. 2020. TREC-COVID: rationale and structure of an information retrieval shared task for COVID-19. *J. Am. Medical Informatics Assoc.* 27, 9 (2020), 1431–1436.

[40] Stephen Robertson. 2004. Understanding inverse document frequency: on theoretical arguments for IDF. *Journal of Documentation* 60, 5 (2004), 503–520. https://doi.org/10.1108/00220410410560582

[41] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. 2015. FitNets: Hints for Thin Deep Nets. In *ICLR (Poster)*.

[42] Guilherme Rosa, Luiz Bonifacio, Vitor Jeronymo, Hugo Abonizio, Marzieh Fadaee, Roberto Lotufo, and Rodrigo Nogueira. 2022. In Defense of Cross-Encoders for Zero-Shot Retrieval. https://doi.org/10.48550/ARXIV.2212.06121

[43] Devendra Singh Sachan, Mike Lewis, Mandar Joshi, Armen Aghajanyan, Wentau Yih, Joelle Pineau, and Luke Zettlemoyer. 2022. Improving Passage Retrieval with Zero-Shot Question Generation. *CoRR* abs/2204.07496 (2022).

[44] Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilic, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, Jonathan Tow, Alexander M. Rush, Stella Biderman, Albert Webson, Pawan Sasanka Ammanamanchi, Thomas Wang, Benoît Sagot, Niklas Muennighoff, Albert Villanova del Moral, Olatunji Ruwase, Rachel Bawden, Stas Bekman, Angelina McMillan-Major, Iz Beltagy, Huu Nguyen, Lucile Saulnier, Samson Tan, Pedro Ortiz Suarez, Victor Sanh, Hugo Laurençon, Yacine Jernite, Julien

Launay, Margaret Mitchell, Colin Raffel, Aaron Gokaslan, Adi Simhi, Aitor Soroa, Alham Fikri Aji, Amit Alfassy, Anna Rogers, Ariel Kreisberg Nitzav, Canwen Xu, Chenghao Mou, Chris Emezue, Christopher Klamm, Colin Leong, Daniel van Strien, David Ifeoluwa Adelani, and et al. 2022. BLOOM: A 176B-Parameter Open-Access Multilingual Language Model. *CoRR* abs/2211.05100 (2022).

[45] Leslie N. Smith. 2017. Cyclical Learning Rates for Training Neural Networks. In *WACV*. 464–472.

[46] Yu Sun, Shuohuan Wang, Yu-Kun Li, Shikun Feng, Hao Tian, Hua Wu, and Haifeng Wang. 2020. ERNIE 2.0: A Continual Pre-Training Framework for Language Understanding. In *AAAI*. AAAI Press, 8968–8975.

[47] Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. BEIR: A Heterogenous Benchmark for Zero-shot Evaluation of Information Retrieval Models. *arXiv preprint arXiv:2104.08663* (2021).

[48] 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 *NIPS*. 5998–6008.

[49] Ellen Voorhees. 2004. Overview of the TREC 2004 Robust Retrieval Track. In *TREC*.

[50] Ben Wang and Aran Komatsuzaki. 2021. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. https://github.com/kingoflolz/mesh-transformer-jax.

[51] Kexin Wang, Nandan Thakur, Nils Reimers, and Iryna Gurevych. 2022. GPL: Generative Pseudo Labeling for Unsupervised Domain Adaptation of Dense Retrieval. In *NAACL-HLT*. Association for Computational Linguistics, 2345–2360.

[52] Lidan Wang, Jimmy Lin, and Donald Metzler. 2011. A cascade ranking model for efficient ranked retrieval. In *SIGIR*. ACM, 105–114.

[53] Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. MiniLM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers. In *NeurIPS*.

[54] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-Art Natural Language Processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Association for Computational Linguistics, Online, 38–45. https://doi.org/10.18653/v1/2020.emnlp-demos.6