Annotating data for fine-tuning a neural ranker? Current active learning strategies are not better than random selection

Althammer, S.; Zuccon, G.; Hofstätter, S.; Verberne, S.; Hanbury, A.; Ai, Q.; ... ; Zobel, J.

Annotating Data for Fine-Tuning a Neural Ranker? Current Active Learning Strategies are not Better than Random Selection

Sophia Althammer
sophia.althammer@tuwien.ac.at
TU Wien, Austria

Guido Zuccon
g.zuccon@uq.edu.au
University of Queensland, Australia

Sebastian Hofstätter
sebastian.hofstaetter@tuwien.ac.at
Cohere, Austria

Suzan Verberne
s.verberne@liacs.leidenuniv.nl
Leiden University, Netherlands

Allan Hanbury
allan.hanbury@tuwien.ac.at
TU Wien, Austria

ABSTRACT

Search methods based on Pretrained Language Models (PLM) have demonstrated great effectiveness gains compared to statistical and early neural ranking models. However, fine-tuning PLM-based rankers requires a great amount of annotated training data. Annotating data involves a large manual effort and thus is expensive, especially in domain specific tasks. In this paper we investigate fine-tuning PLM-based rankers under limited training data and budget. We investigate two scenarios: fine-tuning a ranker from scratch, and domain adaptation starting with a ranker already fine-tuned on general data, and continuing fine-tuning on a target dataset.

We observe a great variability in effectiveness when fine-tuning on different randomly selected subsets of training data. This suggests that it is possible to achieve effectiveness gains by actively selecting a subset of the training data that has the most positive effect on the rankers. This way, it would be possible to fine-tune effective PLM rankers at a reduced annotation budget. To investigate this, we adapt existing Active Learning (AL) strategies to the task of fine-tuning PLM rankers and investigate their effectiveness, also considering annotation and computational costs. Our extensive analysis shows that AL strategies do not significantly outperform random selection of training subsets in terms of effectiveness. We further find that gains provided by AL strategies come at the expense of more assessments (thus higher annotation costs) and AL strategies underperform random selection when comparing effectiveness given a fixed annotation cost. Our results highlight that "optimal" subsets of training data that provide high effectiveness at low annotation cost do exist, but current mainstream AL strategies applied to PLM rankers are not capable of identifying them.

KEYWORDS

PLM-based rankers, domain adaptation, active learning

ACM Reference Format:
Sophia Althammer, Guido Zuccon, Sebastian Hofstätter, Suzan Verberne, and Allan Hanbury. 2023. Annotating Data for Fine-Tuning a Neural Ranker? Current Active Learning Strategies are not Better than Random Selection. In *Annual International ACM SIGIR Conference on Research and Development in Information Retrieval in the Asia Pacific Region (SIGIR-AP '23), November 26–28, 2023, Beijing, China.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3624918.3625333

SIGIR-AP '23, November 26–28, 2023, Beijing, China

ACM ISBN 979-8-4007-0408-6/23/11.
https://doi.org/10.1145/3624918.3625333

1 INTRODUCTION

Search methods based on Pre-trained Language Models (PLM) have shown great effectiveness gains compared to common statistical models and early neural methods [13, 20, 21, 37, 55]. These language models are pre-trained for language representation learning on a background corpus; they are then further trained for a specific task – a process commonly referred to as fine-tuning. Typically, PLM rankers are created through the fine-tuning of a PLM to the ranking task (and possibly, to a specific domain). The fine-tuning of PLM rankers typically requires a great amount of labelled training data. This can often be a challenge when considering search tasks with no or little training data available. Data annotation typically requires a large manual effort and thus is expensive, especially in domain-specific tasks where annotators should be domain experts. In real-life settings, annotation and computational budget[1] is often limited, especially for start-ups or in domain-specific contexts.

In this paper we focus on the problem of fine-tuning PLM rankers under limited training data and budget. There are alternative directions one may take to deploy a PLM ranker in a specific task for which no or limited training data is available. These include for example the zero-shot application of PLM rankers trained on another, resource-rich, retrieval task or domain [54, 60], the learning with few-shot examples [15], and approaches based on pseudo-labelling [58]. However the effectiveness of these approaches depends on the relatedness of the fine-tuning task or the pre-training domain of the language model to the target retrieval task [59]; thus their generalization capabilities remain unclear. Therefore performing domain adaptation by fine-tuning the PLM ranker on the target task with annotated training data (the setting investigated in this paper) remains favourable for a (reliable) high effectiveness [12].

It is unclear however how much annotated training data is required for training an effective PLM ranker. Furthermore, in presence of a budget constraint that restricts the amount of data that can be annotated for training, it is unclear whether it is possible to select training data to minimise annotation cost while maximising ranker effectiveness.

[1]With annotation budget we refer to the amount of money set aside for paying annotators to label pairs of queries and documents. With computational budget, we refer to the amount of money set aside for paying the computation costs arising from the training/fine-tuning of the PLM rankers. These costs may include the hardware and energy costs, or the purchase of cloud solutions.

In this paper, (1) we investigate how the amount of labelled data used for fine-tuning a PLM ranker impacts its effectiveness, (2) we adapt active learning (AL) strategies to the task of training PLM rankers, (3) we propose a budget-aware evaluation schema including aspects of annotation and computation cost, (4) we conduct an extensive analysis of AL strategies for training PLM rankers investigating the trade-offs between effectiveness, annotation budget and computational budget. We do this in the context of three common PLM ranker architectures: cross-encoders (MonoBERT [43]), single representation bi-encoders (DPR [32]) and multi-representation bi-encoders (ColBERT [33]), and two scenarios:

❶ **Scratch**: the PLM is pre-trained on a background corpus, but has yet to be fine-tuned to the target ranking task and dataset;
❷ **Re-Train**: domain adaptation of the PLM ranker is performed. The PLM is pre-trained on a background corpus and fine-tuned to a ranking task and a specific dataset, but further fine-tuning has yet to be performed to transfer the ranker to another dataset and, possibly, a ranking task with characteristics that differ from those of the first fine-tuning process.

To investigate the effect of the amount of labelled data on the effectiveness of PLM rankers, we select incremental amounts of data to fine-tune a ranker. Our empirical results show that the size of the dataset available for fine-tuning the PLM ranker greatly influences the effectiveness of the ranker. While, somewhat unsurprisingly, we find that in general more training data leads to higher effectiveness, we also find large variability in effectiveness between different randomly selected training sets of the same size. Furthermore we find that, for some training sizes, the best random selection run outperforms the worst one, and significantly. This shows that there are subsets of the training data which lead to significant improvements within the same training data size.

This variability motivates us to investigate whether we can select those "high-yield" samples using Active Learning strategies. The intuition is that a good selection strategy would lead to a smaller amount of data to be annotated, and thus a lower annotation cost, while still producing a highly effective ranker. Selection of training data has been extensively investigated in AL for machine learning. Here, common active selection strategies are based on uncertainty or diversity criteria [8, 36, 50]. We thus adapt representative methods that implement these criteria to the context of fine-tuning PLM rankers. We evaluate the representative active selection strategies in terms of their effectiveness for fine-tuning PLM rankers on different training data sizes and compare the strategies to random selection of training data as baseline. For both scenarios the active selection strategies do not offer statistically significant improvements compared to random selection. For certain scenarios and PLM rankers we find varying beneficial selection strategies, however no selection strategy shows consistent and robust higher effectiveness than random selection. In addition, the adoption of active learning requires extra computation compared to random selection.

Since it is not our goal to minimize the training data size, but actually we aim to minimize the total cost of fine-tuning PLM rankers, we revisit the results in light of a budget-aware evaluation we introduce in this paper. This evaluation includes aspects of annotation cost as well as cost of computing resources. With this, we find that the annotations are the main cost factor. Since the selection methods require a different number of assessments to annotate a training set of a certain size, we compare the number of assessments to the effectiveness of the PLM rankers for random and active selection strategies. This reveals that the (marginal, if any) effectiveness gains provided by AL strategies come at the expense of more assessments (thus higher annotation costs) and AL strategies under-perform random selection when comparing both effectiveness and associated cost.

We publish our code at: *github.com/sophiaalthammer/al-rankers*.

2 RELATED WORK

Effect of Data Size on PLM Rankers. Previous studies have observed that fine-tuning PLM rankers on subsets of the available training data decreases search effectiveness and, similarly, that increasing the size of the training data tends to improve search effectiveness. These types of observations and preliminary findings are reported for MS MARCO [12, 23, 27, 32, 68] and in the case of domain adaptation [25, 30, 42, 58]. However, these conditions have never been systematically evaluated, which we do in our study. For example, Nogueira et al. [44] observe the variability in effectiveness when fine-tuning a MonoBERT ranker on subsets of different size (1k, 2.5k, 10k); however they do not systematically investigate this variability, nor they study the effect of larger subsets or different ranking architectures. Mokrii et al. [42] investigate transfer learning for MonoBERT rankers first fine-tuned on MS MARCO and then transferred to question answering tasks in a zero-shot and full training setting, where the source and target domain hold large training sets. They also investigate the effect of training on subsets of the training data and find that the more training queries, the higher the effectiveness. Zhang et al. [69] investigate domain transfer of BERT cross-encoders in a small data regime where they transfer MonoBERT from web search (trained on MS Marco) to small domain specific retrieval tasks. Interestingly they find that small in-domain training data sometimes decreases search effectiveness compared to the zero-shot application of MonoBERT.

Active Learning for Information Retrieval. Active Learning aims to minimize the annotation cost associated with the acquisition of training labels while maximizing the effectiveness of the trained model. Uncertainty [36] and diversity-based [50] strategies form the bulk of AL methods that have been proposed and extensively validated across a variety of learning tasks and datasets. In this paper, we adapt methods belonging to these two strategies.

Specifically, we explore the use of Active Learning for selecting data for the fine-tuning of PLM rankers. Active Learning has been used in Information Retrieval across a number of tasks and settings [8, 17, 18, 35, 38, 49, 51, 52, 64, 70], but never before in the context of PLM ranker fine-tuning.

Of particular interest for this paper are the methods of Cai et al. [5] and Xu et al. [61], that we describe next, because we adapt them to our task of fine-tuning a PLM ranker. Cai et al. [5] transfer a learning-to-rank (LTR) model trained on one target domain to another source domain. For the domain adaptation training they propose to use the Query-by-Committee algorithm for active selection of queries in the target and in the source domain as well as mixing the training sets of the target and source domain. For the domain adaptation of a LTR model, QBC reaches a higher retrieval effectiveness with less training data than the random selection

strategy. Xu et al. [61] investigate different diversity-based active learning strategies for updating query relevance scoring and propose a combination of diversity and density based selection.

A variation of the AL setting that has shown success in certain domain-specific tasks is that of continuous active learning [26, 46, 63], where documents are iteratively retrieved by actively learning for one specific query, typically aiming for total recall [41]. For the task of technology assisted review (TAR), Yang et al. [62] propose a TAR cost framework, however this framework focuses on cost modeling for reviewing one specific query.

Despite previous successes in the use of AL strategies in the context of search and ranking, AL strategies have not been studied for PLM rankers. The AL strategies proposed in previous work are not directly applicable to PLM rankers – however in Section 5 we propose adaptations of these methods to our task of interest.

3 CONSIDERED PLM RANKERS

In the consider cross-encoder model, MonoBERT, query and passage text are concatenated, encoded with BERT and the CLS representation is scored with a linear layer W on top of the encoding:

$$s = W \text{ BERT}(\text{CLS}; q; \text{SEP}; p; \text{SEP})_{CLS} \quad (1)$$

where SEP is the separator token and s is the final score of passage p for query q. Empirical findings show MonoBERT reaches a high re-ranking effectiveness [43], however each passage needs to be encoded at query time and therefore this architecture is computationally resource-heavy and is characterized by high query latency [48]. For the same reason, this ranker is commonly used only in top-k re-ranking settings, and not for retrieval (i.e., scoring the whole collection for each query).

DPR [32] encodes the query and passages independently. The relevance of a passage p to a query q is estimated using the dot-product between the CLS token representation q and that of p:

$$s = \text{BERT}(\text{CLS}; q; \text{SEP})_{CLS} \cdot \text{BERT}(\text{CLS}; p; \text{SEP})_{CLS} \quad (2)$$

The independence of query and passage encoding and dot-product relevance scoring make it possible to pre-compute and store the passage representations in the index and enable efficient retrieval at query time with approximate nearest neighbor search [31, 39].

The ColBERT [33] method delays the interaction between the query and passage to after the encoding by computing the relevance score as the sum of the maximum similarity scores between all token representations of the query and passage:

$$s = \sum_j \max_i \left[\text{BERT}(\text{CLS}; q; \text{SEP})_j \cdot \text{BERT}(\text{CLS}; p; \text{SEP})_i\right] \quad (3)$$

As for single representation bi-encoder methods, also in ColBERT the passage representation can be pre-computed offline and thus the query processing is sped up. Empirical results show that ColBERT achieves a competitive effectiveness compared to MonoBERT [33].

4 TRAINING SCENARIOS & ANNOTATION MODELING

We consider two scenarios for training the PLM rankers: ❶ **Scratch** training from scratch, starting with a PLM and ❷ **Re-Train** domain fine-tuning after rank/retrieval fine-tuning of the PLM has occurred. These are common scenarios that are encountered in the practical application of PLM rankers to search problems.

In ❶ **Scratch** our objective is to train a PLM ranker "from scratch", i.e., without having already performed any fine-tuning on a retrieval task. There are many reasons this scenario could occur in the practical deployment of PLM rankers. For example, no suitable labelled data corresponding to the ranking task may be available, or the data that may be available is protected by a license that prevents its use within a product (e.g., the MS Marco dataset). We model the first scenario by starting from a pre-trained BERT model [16, 47] and training the ranker on the MS Marco dataset, a large scale web search collection commonly used to train these rankers. Note, in our experiments we assume that no labels are available for the dataset, and labels are iteratively collected (in a simulated setting) within the AL cycle.

In ❷ **Re-Train** our goal is to adapt a PLM ranker to a specific retrieval task (potentially in a specific domain). Here we assume that the PLM ranker has already undergone fine-tuning on a high-resource retrieval task (e.g., using the common MS Marco dataset), and the goal is to further fine-tune the ranker with additional data, on a different retrieval task or data domain. This is a common setting in domain-specific IR settings. The assumption is that the initial fine-tuning on the non-target retrieval task or domain data still highly contributes to the effectiveness of the ranker, especially when the target data available for fine-tuning is limited. We model the second scenario by starting from a ranker fine-tuned on MS Marco and fine-tune the ranker for a domain-specific retrieval task. In our experiments, we choose to validate the models using the retrieval task and datasets associated with health-oriented web search in the medical domain. We choose this task due to the availability of the TripClick dataset [45], a large-scale training and test set for this task. This dataset has similar characteristics to MS Marco (e.g. query length, sparse judgments). In contrast to other domain adaptation approaches [5], we do not mix the training sets of the source and the target domain, in order to (i) be able to separate the effects of mixing the training sets from the active domain adaptation strategies, and (ii) study PLM ranker development and deployment strategies that are in line with the green IR principles of reuse and recycle [48].

In order to model the real-life process of incremental annotation and training we incrementally increase our fine-tuning set D. The details of this incremental process are depicted in Algorithm 1. We start with an empty set $D = \{\}$ and in each iteration a subset S of the whole training set T ($S \subset T$) is selected to be added to D. We model the annotation process by attaining the labels from the training set qrels and adding the samples to the fine-tuning set ($D = D \cup S$). Then we train the PLM ranker on the updated set D and, based on random or active selection strategies, we select the next subset to annotate and add it to the training set.

5 ACTIVE SELECTION STRATEGIES

We consider three active selection strategies to identify training data for labelling: uncertainty-based selection [36, 65, 70], query-by-committee (QBC) [5], and diversity-based selection [61]. We consider random selection as a baseline selection strategy. Next, we describe the active selection strategies and how we adapt them for fine-tuning PLM rankers.

5.1 Uncertainty-based selection

The uncertainty-based selection strategy selects samples by measuring the model's (ranker) uncertainty in the scores it produced

and then selecting the samples with the least confidence [8, 36]. Uncertainty-based strategies are commonly applied to classification problems, and often the score provided by the classifier is used as direct indication of uncertainty: scores are in the range $[0, 1]$, the decision boundary is set to 0.5 and the confidence in the classification is measured in function of the distance to the decision boundary (the closer, the least confident) [19, 36].

This approach is not directly transferable to PLM rankers since their relevance scores are not necessarily bounded and therefore there is no clear decision boundary measuring the uncertainty in the ranking. We note that uncertainty estimation in Information Retrieval is a fundamental but largely unexplored problem [9, 14, 56], especially for rankers based on PLMS [7, 34].

In this work, we model the uncertainty distribution by means of the score distribution of the top K ranked passages for all queries in the training set $T \backslash D$ and select the query-passage pairs with the relevance score closest to the mean of the score distribution for the ranker, hence with the highest uncertainty. We leave the investigation of other upcoming approaches for future work (see Section 9 for further insights).

In order to model the annotation and training process for the selected query-passage pairs, the selected passage is assigned its label from the training set. In case the selected passage is *relevant* we sample an irrelevant passage randomly from the BM25 top 1,000 to construct a training triplet (query, positive passage, negative passage). In case the selected passage is *irrelevant*, we take the selected passage as negative for the triplet and take the first relevant passage in the BM25 top100 list re-ranked by the PLM ranker as positive passage. For each selected query-passage pair we add one triplet to the training set.

Note that the uncertainty-based selection is operating at query-passage level, while the other AL methods we consider next are operating at a query-only level.

5.2 Query-by-committee selection

The query-by-committee (QBC) method [22, 49] is a specific uncertainty-based selection strategy. In QBC, multiple committee members (models) are used to classify or rank samples; then the disagreement between the committee members on classified/ranked samples is measured and the samples with the highest disagreement are selected for annotation. A previous adaptation of QBC to information retrieval is due to Cai et al. [5] who apply QBC to Learning-to-Rank for domain adaptation. For this, they train different members of the committee by training on subsets of the currently annotated training set at hand. The disagreement between the committee members is then measured by the vote entropy of the different rankings of the members for the queries which are not yet annotated. In the vote entropy the committee members M vote on the partial order of two passages $N(p_1 \prec p_2)$ in the ranked list R, counting how many of the members rank p_1 higher than p_2. The vote entropy of a query q is then defined as:

$$VE(q) = \frac{-1}{|M|} \sum_{p_i, p_j \in R} N(p_i \prec p_j) log \left(\frac{N(p_i \prec p_j)}{|M|} \right) \quad (4)$$

The queries with the highest vote entropy are selected for annotation. In order to model the annotation process and to select training triples for training PLM rankers, for every query that selected to add to the training set we take the first relevant passage in the BM25 top100 list re-ranked by the PLM ranker as positive passage and sample a random negative passage from the BM25 top 1,000 passages. We choose to use the re-ranked list of the first member for selection. For every query we add one training triplet to the training set.

Algorithm 1 Incremental annotation and training process

Input: T whole training set, I number of iterations, s number of added samples per iteration, M is a PLM ranker/retriever
Output: D annotated training set, M PLM ranker trained on D

```
D ← {}
for i in I do
    Select subset S ⊂ T of size |S| = s with selection strategy
    Annotate S, T ← T \ S
    D ← D ∪ S
    Train M with D
end for
```

5.3 Diversity-based selection

Diversity-based selection strategies select training samples based on the diversity of the samples – typically comparing already selected samples to those yet to select. Within Information Retrieval, diversity-based selection strategies have been used for Learning-to-Rank models [50, 61, 64]. In those settings, diversity is measured by clustering queries using an external unsupervised clustering model and taking one representative query from each cluster.

In our adaptation of diversity-based selection strategies to PLM rankers, to compute diversity we consider the query representation made by the PLM ranker. This has the advantage that we leverage the model's representations to compute diversity, instead of relying on an external model. Furthermore, this representation changes in each iteration as the PLM ranker is trained incrementally through training sets of increasing size: therefore, the query representation also accounts for changes within the ranker itself. For DPR and ColBERT, we use the CLS token representation of the encoded query as query representation. For MonoBERT we encode the query without a passage and also take the CLS representation for measuring the diversity. We cluster the query representations of queries in $T \backslash D$ with the number of clusters equaling the number of training samples to be added in that iteration. From each cluster, we randomly sample one query to be annotated.

To add a training triplet to the training set for each selected query, we rely on the same annotation process used in QBC.

6 BUDGET-AWARE EVALUATION

Next we introduce a framework to evaluate active learning to PLM ranker fine-tuning within budget constraints. For this, we model the costs related to both annotation effort and computation.

Annotation Costs. For measuring the annotation costs, we count the number of assessments needed to annotate a training triplet for a single query. The number of assessments corresponds to the rank of the first relevant passage found in the ranking of the PLM ranker, which is trained in the previous iteration of the AL process. In our setting, the annotation for a query stops when one relevant passage for a selected query is found – thus we need to assess all passages in the ranking for that query up until the relevant passage is found

and also annotated.This implies that the number of assessments differs from the training data size (the number of queries), e.g., one query sample in the training data can account for 10 assessments required when the first relevant passage is found at rank 10.

The total annotation cost then is the sum of the number of assessments for all the queries added to the training set. Formally, let $A(i)$ be the number of assessments needed to create the training data of iteration i, A_h be the number of assessments an annotator finishes in one hour and A_C be the cost for an annotator per hour[2]. Then, the total annotation cost at iteration i is computed as:

$$C_A(i) = \frac{A(i)}{A_h} \cdot A_C \tag{5}$$

Computational Costs. Next we model the computational costs involved in executing the active selection strategies. These strategies usually will require both CPU and GPU based computation, which typically incur different costs and thus we account for separately. Let $H_{GPU}(i)$ be the accumulated number of GPU hours needed for training a PLM ranker for iteration i and G_h be the cost of running an GPU for one hour. Then, the total computational cost at iteration i is computed as:

$$C_C(i) = H_{GPU}(i) \cdot G_h + H_{CPU} \cdot C_h \cdot (i-1) \tag{6}$$

with H_{CPU} the number of CPU hours needed for computing the selection strategy and C_h the cost of one hour CPU.

Total Cost. Finally, the total cost at iteration i can then be computed using Equations 5 and 6:

$$\begin{aligned} C(i) &= C_A(i) + C_C(i) \\ &= \frac{A(i)}{A_h} \cdot A_C + H_{GPU}(i) \cdot G_h + H_{CPU} \cdot C_h \cdot (i-1) \end{aligned} \tag{7}$$

7 EXPERIMENTAL SETUP

Next we describe the experimental setup we have devised to study the AL strategy for PLM rankers fine-tuning we have illustrated above. We develop our investigation along the following three lines of inquiry:

RQ1 What is the effect of the size of the labelled training data on the effectiveness of PLM rankers?

RQ2 How do different active selection strategies influence the effectiveness of PLM rankers?

RQ3 What is the effect of using an active selection strategy to fine-tune a PLM ranker under a constrained budget?

7.1 Passage Collection & Query Sets

For ❶ **Scratch**, we use the MS Marco passage collection [3]. MS Marco is based on sampled Bing queries and contains 8.8 million passages; its training set contains $530k$ training triplets. We use the training portion for fine-tuning and evaluate on the TREC DL 2019 [10] and 2020 [11] with nDCG@10.

For ❷ **Re-Train**, we use the TripClick dataset. This dataset contains real user queries and click-based annotations. It consists of 1.5 million passages and $680k$ training queries. Test queries are divided with respect to their frequency into three sets of $1,750$ queries respectively; the three sets are Head, Torso, and Tail. For the Head queries a DCTR [6] click model was used to create relevance signals from the click labels. We evaluate on the Head DCTR and the Torso Raw test set as in related work [1, 28, 45].

[2]Note that certain search tasks or domain may require multiple annotators to examine the same sample: in this case A_C would be the sum of the hourly rates associated to all the annotators.

7.2 PLM ranker details

We train MonoBERT, ColBERT and DPR using training triplets with a RankNet loss [4]. The triplets consist of the query, a relevant and an irrelevant passage; negative passages are taken from the top 1000 BM25 negatives. We train DPR and ColBERT with a batch size of 100, while we use a batch size of 32 for MonoBERT due to its high computational requirements. We train all models for 200 epochs with a learning rate of 7×10^{-6} and we use early stopping. For training, we impose a maximum input length of 30 tokens for the query and 200 tokens for the passage; this setting truncates only a few outliers samples in the dataset but provides computational advantages for batching.

In ❶ **Scratch** we perform fine-tuning from scratch; as underlying PLMs we use DistilBERT [47] for DPR and ColBERT and the `bert-base-uncased` model [16] for MonoBERT both provided by Huggingface. We choose these models as starting point so that they match the fine-tuned models for ❷ **Re-Train**. In ❷ **Re-Train** we start with PLM rankers fine-tuned on MS Marco. For DPR we start from TASB [29], trained with knowledge distillation and topic-aware sampling; for ColBERT from a ColBERT DistilBERT model trained with knowledge distillation; for MonoBERT from a `bert-base-uncased` model solely trained on MS Marco.

For MonoBERT and ColBERT, we report results in a re-ranking context, i.e. using these PLM rankers to re-rank the top 1,000 results retrieved by BM25. For DPR, we instead consider a retrieval setting, where all the collection is scored and then only the top 1,000 are used for evaluation. However, the findings we observe for DPR in the retrieval setting are similar to those we obtained for the same PLM in a re-ranking setting (not reported here). We decided to report retrieval results for DPR, rather than re-ranking as for the other two PLM, because DPR is more commonly used for retrieval (while the other two for re-ranking).

7.3 Active learning details

As foundational experiment we train the PLM rankers on different subsets of the training data of differing sizes; as size, we explore the values $[1k, 5k, 10k, 20k, 50k, 100k, 200k]$ for MS Marco and $[1k, 5k, 10k, 20k, 50k]$ for TripClick. We repeat these experiments 4 times with different random seeds for sampling the subsets, so that each time we train on different subsets with the same size and we can measure variance.

In our experiments we use random selection as a baseline and increase the training set incrementally. We run the random baseline 4 times with different random seeds.

For the active learning process, we increase the training subsets incrementally as denoted in Algorithm 1. In each iteration we train the PLM ranker from scratch to exclude a potential bias from incrementally training a ranker. In the first iteration we randomly select the first subset with the same random selection across the different active learning strategies. For uncertainty and diversity selection one could select the first batch with the selection strategy, however for QBC this is not possible because different committee members for selection are not available in the first iteration. Therefore we do random selection in the first iteration to be able to fairly compare across the three strategies.

For fast and resource efficient active selection, we train the PLM rankers for 15 epochs and use the trained ranker for active selection. For the sake of evaluation and in order to compare the effectiveness at different iterations, we resume the training after 200 epochs.

For the uncertainty-selection and QBC strategies, we score the BM25 top 100 passages of each training queries and use these passages for actively selecting the queries for annotation. For the QBC selection strategy we use the same hyper-parameters as Cai et al. [5]; we use 2 members in the committee and train each member on 80% of the subset available at each iteration for training. We choose the size of the training subsets so that each 80% portion aligns with the other training set sizes.

For ❶ **Scratch** we add in each iteration 5,000 training samples to the training size. For ❷ **Re-Train** we have 5,000 samples for the first 2 iterations until the training size is 10k, from then on we use 10,000 samples for the remaining iterations in order to decrease computational cost.

7.4 Costs for Budget-Aware Evaluation

For computing the annotation cost, for each triplet added to the training we store the rank of the first relevant document in the ranked list generated by the PLM ranker trained in the previous iteration. Since we do not have a trained PLM ranker in the first iteration, we start with the initial ranking provided by BM25. For the random baseline, we also use the initial BM25 ranking for computing the annotation effort.

We conduct our experiments on servers equipped with NVIDIA A40 GPUs and measure the GPU and CPU hours spent in the training of the PLM ranker and the execution of the selection strategies.

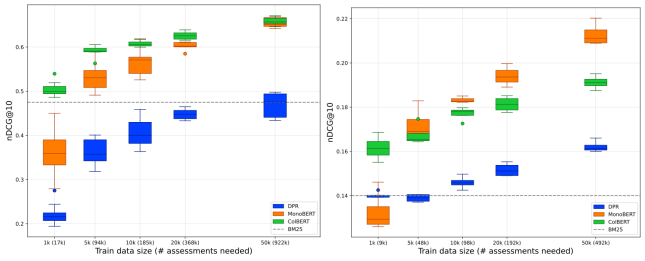
For the computational cost, we refer to common cloud computing costs[3] and set $G_h = 3.060\$$ and $C_h = 0.408\$$. For the number of annotations per hour A_h we rely on estimates from Althammer et al. [1] who conducted an annotation campaign on TripClick test set. Here annotators needed 47.7 seconds to annotate a query-passage pair on average, which corresponds to 75 assessments per hour. For the annotation cost per hour, A_C, we assume $50US\$$ as hourly rate of a domain expert annotator. We also have developed a small web tool[4] to let the reader customise computation and annotation costs and the number of assessments per hour; then the reader can observe how the results presented in terms of budget-aware evaluation change according to different cost settings.

8 RESULTS

8.1 RQ1: Effect of Size of Training Data

We visualize the effect of training data size on the effectiveness of PLM rankers for ❶ **Scratch** (Figure 1a) and for ❷ **Re-Train** (Figure 1b). The boxplots visualize the range of effectiveness when the PLM ranker is trained on different subsets of the same size.

In both cases, it is observed that as the size of the training data increases, nDCG@10 improves for all three PLM rankers. When considering effectiveness across PLM rankers, it is noteworthy to observe ColBERT and MonoBERT. Recall from the literature that MonoBERT outperforms ColBERT on MS Marco when both are trained on the whole MS Marco training data [13], and the same



(a) ❶ Scratch. (b) ❷ Re-Train.

Figure 1: Boxplot of nDCG@10 effectiveness on TREC DL 2020 (❶ Scratch, Figure 1a) and on TripClick Head DCTR test (❷ Re-Train, Figure 1b), visualizing the variability of training on different training sample sizes. PLM rankers are trained on subsets of respective sets (MS Marco/TripClick) with different sizes. To measure variability, for each train data size we repeat random sampling 4 times.

holds for TripClick [28]. However, in our experiments, we observe that ColBERT outperforms MonoBERT for smaller training data sizes. MonoBERT eventually becomes better than ColBERT but only once more than 10,000 training samples are used for the ❷ **Re-Train** scenario. For the ❶ **Scratch** scenario the two rankers becomes largely indistinguishable when the training data is 50,000 samples, and eventually MonoBERT takes the lead thereafter (not shown in the figure).

In ❶ **Scratch**, the improvement in effectiveness with increasing training size is particularly remarkable for small training subsets. For example, the improvement by adding 4k training samples from 1k to 5k samples is between 18% and 63% of the median nDCG@10. Noting the wide scale of the y-axis from 0.2 to 0.7 nDCG@10, we observe a large variability when training PLM rankers on limited data. This is particularly the case for MonoBERT, where we find a large difference from maximum and minimum nDCG@10 from 19 (0.27–0.46 nDCG@10) for 1k samples to to 7 (0.53–0.60) for 10k samples. The worst and the best MonoBERT run obtained are statistically different for train size 1k and 5k. For DPR, the interquartile range is up to a difference of 5 nDCG@10 (0.38-0.43 for 10k), thus 50% of the effectiveness points are within a range of 5 nDCG@10. A substantial variability in the effectiveness of DPR is observed when trained on 50k samples. The best and the worst runs for DPR are statistically different for 5k and 10k samples. It is noteworthy that the boxplots for 5k samples overlap in part with those for 10k, and similarly the 10k with those for 20k. This means that specific subset of training data of size 5k (10k) allow to reach the same effectiveness obtained when training the ranker on double the amount of data, i.e. 10k (20k).

For ❷ **Re-Train** (Figure 1b) we also notice variability in search effectiveness; yet, we observe a relatively smaller variability compared to ❶ **Scratch**. The differences between the worst and best runs for each training data size are not statistically significant in this scenario. We suspect that this smaller variability in effectiveness is due to starting from an already fine-tuned PLM ranker instead of training from scratch. Although our empirical results suggest a smaller variability, we still see overlaps of the boxplots, especially between 10k and 20k sample: that is, the same or even better effectiveness could have been reached with half the training data.

These results suggest that it is possible to select subsets of training data that would "speed-up" the learning: in other words, some

[3]From https://aws.amazon.com/ec2/pricing/on-demand/. Costs valid as of 02 January 2023. GPU costs refer to a `p3.2xlarge` instance and CPU costs to an `a1.4xlarge` instance.
[4]We make this publicly available on acceptance of the paper.

Table 1: nDCG@10 effectiveness across different amounts of training data for ❶ Scratch on TREC DL 2019 & 2020 and for ❷ Re-Train on TripClick Head DCTR & Torso Raw. Bold numbers denote highest effectiveness for each PLM ranker and training size. Statistical significant differences to random selection baseline (Random) is denoted with * (paired t-test; p < 0.05, Bonferroni correction with n=3). For ❶: No consistently best performing method and no statistical significant difference to Random. For ❷: for DPR, Random consistently is best; all statistical significance differences to Random are significantly lower. '-' indicates no result at that training size.

nDCG@10	❶ Scratch: MS Marco										❷ Re-Train: TripClick									
	TREC DL 2019					TREC DL 2020					Head test DCTR					Torso test raw				
Train data size	0	5k	10k	20k	50k	0	5k	10k	20k	50k	0	5k	10k	20k	50k	0	5k	10k	20k	50k
0 BM25	.501					.475					.140					.206				
MonoBERT (re-rank BM25 top 1,000)																				
1 Random	.051	.5935	.6272	.6430	.6705	.041	.5590	.5871	.6148	.6552	.036	.1715	.1833	.1941	.2129	.036	.2279	.2352	.2426	**.2710**
2 QBC		**.6193**	.6157	.6246	**.6728**		.5507	.5844	**.6443**	.6630		**.1731**	.1835	**.2065**	.2059		.2046	.2328	.2423	.2679
4 Uncertainty		.6118	.6232	**.6588**	.6509		**.5875**	.5873	.6336	.6595		-	**.1920**	.1981	**.2190**		-	**.2356**	.2362	.2705
5 Diversity		.5925	**.6341**	.6448	.6640		.5407	**.6237**	.6338	**.6670**		-	.1837	.1933	.2123		-	.2294	**.2450**	.2650
ColBERT (re-rank BM25 top 1,000)																				
6 Random	.352	.6176	**.6385**	.6352	.6614	.246	.5944	.6091	.6291	.6577	.155	**.1675**	.1770	.1813	.1912	.227	**.2300**	**.2351**	**.2397**	.2475
7 QBC		.6192	.6297	**.6541**	**.6680**		.5813	.6159	**.6511**	**.6758**		.1302*	**.1791**	**.1860**	**.1962**		.1558*	.2273	.2292	.2360
9 Uncertainty		.6257	.6034	.6089	.6370		**.5987**	.6076	.6001	.6246		-	.1645	.1536	.1753*		-	.2190	.1909	.2274
10 Diversity		**.6271**	.6239	.6402	.6644		.5912	.6038	.6211	.6363		-	.1645	.1811	.1957		-	.2187	.2362	**.2481**
DPR (full retrieval)																				
11 Random	0.0	.3674	**.4390**	.4457	.5006	0.0	.3225	.3789	.4190	.4757	.139	**.1389**	**.1459**	**.1516**	**.1621**	**.200**	**.1837**	**.1745**	**.1924**	**.2023**
12 QBC		.3465	.4343	.4628	**.5079**		.3023	.3849	.4090	.4534		.0849*	.1043	.1368	.1603		.0895	.1122	.1312	.1440
14 Uncertainty		**.3961**	.4067	.4255	.3757*		**.3660**	.3733	**.4476**	.4254		.1060	.1165	.1283	.1336		.0907	.0946	.1030	.1031
15 Diversity		.3713	.4086	.4593	.4750		.3437	.4030	.4198	**.4998**		.1059	.1150	.1163	.1458		.0907	.1041	.1217	.1473

subsets of training data can achieve the same or even higher effectiveness as using double the amount of data. This thus serves as a motivation for this paper: is it possible to identify "*high-yield*" training subsets so as to spare annotation costs but yet obtain high effectiveness? To this aim, we investigate the effectiveness of active learning strategies, which we discuss next.

8.2 RQ2: Effectiveness of Active Selection

In Table 1 we report the effectiveness of the active learning strategies from Section 5, along with the random selection baseline, when used for training MonoBERT, ColBERT and DPR across different amounts of training data in scenario ❶ **Scratch** and ❷ **Re-Train**.

For the random selection baseline we report the mean effectiveness when randomly sampling and training on different subsets of the same size multiple times – we perform four random selections for each training size. Note that because the AL selection strategies are deterministic, there is only one result for each strategy at a certain training size, not multiple runs as for the Random baseline.

Various AL strategies outperform the Random baseline at most training data sizes in ❶ **Scratch**. However these effectiveness gains are not consistent throughout all training sizes: there is no single AL strategy that always perform better than the others and, importantly, that always outperforms the random selection baseline. For example, on TREC DL 2020 the uncertainty-based selection for DPR reaches the highest effectiveness when training with 20k samples, but effectiveness drops sensibly when training with 50k samples. Furthermore, effectiveness gains across all methods are not statistically significant, nor are the improvements substantial. When evaluating the PLM rankers on MS Marco Dev, we find similar results, there are varying, non-statistically significant improvements of AL strategies to the Random baseline; due to space constraints we do not report these measures here.

The effectiveness results are more consistent across methods and training data sizes in scenario ❷ **Re-Train**. Random outperforms all AL selection strategies when using DPR. The QBC strategy reaches slightly higher effectiveness than random selection when ColBERT is used; however, none of the improvements are significant despite the large number of test queries in the TripClick Head and Torso test sets. No statistical significance is found even when the worst random selection run is considered in place of the mean of the random runs.

In summary, we found that for the task of fine-tuning PLM rankers, there is no single active learning selection strategy that consistently and significantly delivers higher effectiveness compared to a random selection of the training data. This is a surprising and interesting result. Active learning has been shown to be effective in natural language tasks [36], also for methods that rely on PLM models [19]: yet, popular AL methods do not work in the context of PLM rankers. However, RQ1 shows that there are subsets of the training data that when used for fine-tuning PLM rankers deliver sensibly higher effectiveness than others – but AL methods are unable to identify those high-yield training samples.

8.3 RQ3: Budget-aware Evaluation

Since the goal of actively selecting training data is to minimize the annotation cost, we investigate the active selection strategies in the context of constraint budgets. For this, we use the budget-aware evaluation of Section 5.3, which accounts for the number of assessments needed to annotate the training data as well as the computational cost of the training and selection.

We visualize the effectiveness and associated costs at different training set sizes for the AL strategies for the three PLM rankers in Figure 2a for ❶ **Scratch** on TREC DL 2020 and in Figure 2b for ❷ **Re-Train** on TripClick Head DCTR. The lines and the left y-axis

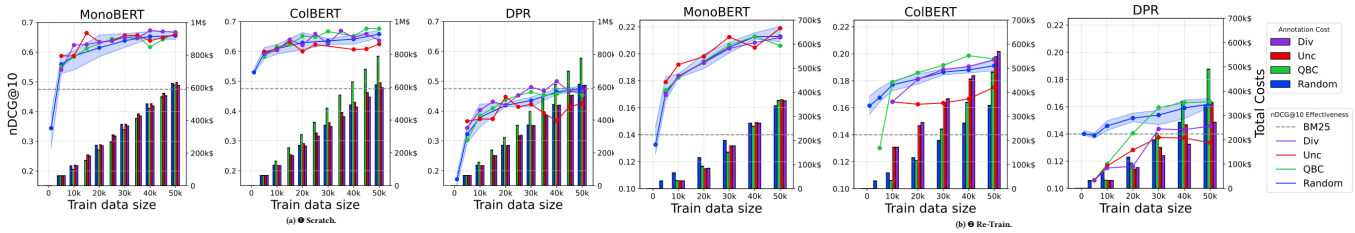


Figure 2: nDCG@10 (lines, left y-axis) and stacked annotation and computational cost (bars, right y-axis) for different train data sizes on TREC DL 2020 (❶ Scratch, Figure 2a) and on TripClick Head DCTR (❷ Re-Train, Figure 2b). For Random (4 runs) the blue line denotes mean, shaded area denotes the range between min and max effectiveness. Good results would be expected to be between mean and max of Random, bad results between mean and min. For stacked cost, only annotation cost is visible since it greatly exceed computational costs.

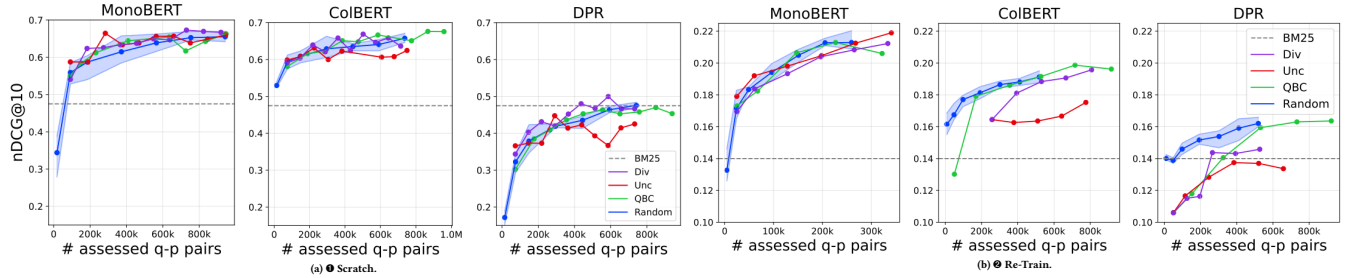


Figure 3: nDCG@10 vs. number of assessed query-passage pairs on TREC DL 2020 (❶ Scratch, Figure 3a) and on TripClick Head DCTR (❷ Re-Train, Figure 3b). Number of assessments per sample is measured with rank of highest relevant passage during selection. For the Random baseline the blue line denotes mean, blue shaded area denotes the range between max and min effectiveness versus mean of the number of assessments. Selection strategies are not consistently more effective considering the number of assessments to annotate the training samples.

refer to the rankers' effectiveness, measured as nDCG@10. The bars and the right y-axis refer to the total cost computed with the budget-aware evaluation. The bars are stacked (the annotation and computational cost), but since with our cost settings the annotation cost greatly exceeds the computational cost, the bars for the GPU and CPU costs are not visible. In all figures the blue line denotes the effectiveness of Random, with the blue shade representing the range measured between the worst and best random selection runs (recall that random selection was ran four times, and Random is the mean effectiveness of these runs).

A first observation is that the main cost factor is the annotation cost, and hence the number of assessments needed to create the training data, which largely overrules the computational cost. Because of this, in Figures 3a (**❶ Scratch**) and 3b (**❷ Re-Train**) we further visualise the effectiveness of the AL strategies relative to the number of assessments needed to reach that effectiveness.

Next, we analyse the results for **❶ Scratch** (Figures 2a and 3a). For MonoBERT, the active selection strategies often provide higher effectiveness than Random when more than 10k samples are available – these effectiveness gains are however not significant. Nonetheless, QBC and diversity require a lower budget than the Random baseline with savings of up to 15k$ when 50k query-document samples are collected. We note that the uncertainty-based strategy provides similar effectiveness to Random (especially from 20k samples), at no cost-savings.

For ColBERT, QBC consistently provides higher effectiveness than Random, however at a much higher cost. For example, when 50k training samples are selected, using QBC costs nearly $200k more than Random, requiring annotations for roughly 200k more query-document pairs. In fact, when approximately the same budget/number of annotations are used, QBC and Random obtain the same effectiveness (in Figure 3a compare the last point of Random with the third last point of QBC). Aside from QBC, all other active selection strategies deliver similar or lower effectiveness of Random, for the same or higher cost.

For DPR, the uncertain-based strategy consistently delivers inferior effectiveness than the baseline. QBC and diversity-based selection do provide effectiveness gains when the training data is in the range 10k to 40-45k samples. For QBC, however, these gains come at a large budget expense: for 30k the QBC selection requires 90k$ more annotation budget than Random. The diversity-based strategy instead does deliver some costs-savings compared to Random. For example, for Random to reach the same effectiveness of BM25, about 600k annotations are needed, while diversity delivers the same level of effectiveness with only 420k annotations. However, we note using more annotations with diversity-based sampling does not necessarily translate in a more effective model: going from 600k to about 750k annotations deteriorates the search effectiveness of the ranker.

Looking across PLM rankers, we observe that while the annotation costs across selection strategies are relatively similar for MonoBERT, they are higher for QBC than all other strategies when ColBERT and DPR are considered.

Overall, the selection strategies show relatively unstable effectiveness, the effectiveness can even decrease when training data increases. This is particularly the case for uncertainty selection for DPR: for example, its effectiveness decreases by from 0.45 to 0.37 when the amount of training data doubles from 20k to 40k.

We next analyse the results for scenario **❷ Re-Train**. While some selection strategies provide gains over random selection, these gains largely depend on which PLM is used and the training size (Figure 2b). Nevertheless, despite the specific gains in effectiveness, all

active selection strategies require more assessments, and thus a higher budget, to reach the same level of effectiveness obtained when using random selection (Figure 3b).

For MonoBERT, uncertainty selection exhibits (non-significant) improvements when training data is less than 30k. In fact, for small amounts of training data, uncertainty sampling does provide some cost savings: for example MonoBERT with uncertainty sampling needs about 65,000 query-passage pairs assessments to obtain the same effectiveness obtained with random selection with $\approx 100k$ assessed pairs. However, this effect is lost when the training data size increases further, with the budget required by uncertainty sampling becoming similar (or more in some instances of random selection) to that of Random to obtain the same level of effectiveness. All other active selection strategies, when used with MonoBERT, deliver either lower effectiveness than Random, or higher costs. This is the case particularly for QBC. In fact, although there is one setting in which QBC delivers major cost savings to reach the same effectiveness of the random baseline (QBC achieves nDCG@10 higher than 0.2 using a sensibly lower amount of annotated query-passage pairs), cost savings are not consistent across all training data sizes and larger sizes correspond to a higher number of assessments required compared to Random.

For ColBERT, QBC and diversity selection outperform the baseline from training data sizes of 20k onward. This however comes with a considerable increase of query-passage pairs to be assessed and thus of annotation cost. For example, with a training subset of 30k, random selection costs about $200,000 while diversity selection costs nearly double that – but the increase in search effectiveness is marginal. It is interesting to compare these results with that obtained for scenario ❶ **Scratch**. While in both scenarios uncertainty selection shows effectiveness losses when more training data is added, and QBC is associated with higher costs, diversity selection performs differently: it provides similar effectiveness for a similar cost in ❶ **Scratch**, and a marginal effectiveness improvement for a largely higher cost in ❷ **Re-Train**.

For DPR, all selection strategies underperform random selection, with the exception of QBC that provides marginal improvements when the training subset is larger than 30k, but this at the expense of a higher budget. The budget-aware evaluation, in fact, shows that all selection strategies require more query-passage pairs to be assessed (higher cost) than the random selection baseline to reach the same search effectiveness (and some strategies cannot even achieve that effectiveness). An example is QBC that requires 730,000 assessments to reach the same effectiveness obtained by Random with just $\approx 250k$ assessments.

In summary, in answer to RQ3, we found that the use of the investigated active selection strategies does not deliver consistent budget savings. In our experiments, the budget is largely dominated by the assessment cost and all active selection strategies tend to require a higher amount of query-passage pairs to be annotated than random selection. Even in contexts where assessment is very cheap, active selection would not provide budget savings because more assessments are required for active selection than for random selection. We do note that there are cases where specific active selection strategies provide similar search effectiveness than random selection at a reduced cost. However, these cases occur for specific choices of selection strategy, PLM ranker and training subset size and thus are unlikely to generalise in practice.

9 CONCLUSION

We investigated fine-tuning PLM rankers under limited data and budget. For this, we adapted several active selection strategies, representing different key approaches in active learning that have been shown effective in many natural language processing tasks. Surprisingly, we found that for the task of fine-tuning PLM rankers no AL strategy consistently and significantly outperformed random selection of training data. However we found that there are subsets of the training data which lead to significantly higher effectiveness than others, thus we see it as an important open challenge to be able to automatically identify those training samples. Similarly, our budget-aware evaluation showed that the investigated AL strategies do not deliver consistent *budget savings* since they require a higher amount of assessments than random selection.

One limitation of our study is that the estimation of annotation costs relies on sparse annotations of the training set. Potentially, the required number of assessments could be lower, since another relevant passage – that is not marked as relevant in the data – could be found earlier in the ranked list. We argue, however, that this should affect all selection strategies and does not benefit one strategy particularly.

Another limitation is the way uncertainty was computed in our experiments. Uncertainty estimation in Information Retrieval is a fundamental but largely unexplored problem [9, 14, 56], especially for rankers based on PLMs [7, 34]. Attempts have been made to exploit uncertainty in relevance estimation for traditional statistical models such as language models and BM25 [57, 71], but in these works the actual estimation of uncertainty is based on assumptions and heuristics such as to be related to similarities or covariance between term occurrences [57, 71, 72], to follow the Dirichlet distribution [57], or to be computed based on score distributions obtained through query term resampling [9]. Recent attempts have been made to model uncertainty for neural rankers, for example Transformer Pointer Generator Network (T-PGN) model [34], or Cohen et al.'s [7] efficient uncertainty and calibration modelling strategies based on Monte-Carlo drop-out [24], but these are not readily applicable to the PLM ranker architectures we consider. In future work we plan to adapt and investigate these uncertainty estimations.

Finally we also highlight that we only considered common baseline active learning methods. More sophisticated AL methods exist [2, 40, 67], including alternating between selection types like in AcTune, which alternates active learning and self-training [66], and Augmented SBERT which alternates random selection and kernel density estimation based selection [53]. However, each of these approaches present specific challenges to be adapted to ranking. We also were interested to understand the promise AL has for PLM-based rankers, and provide a framework, inclusive of evaluation methodologies and baselines, in which these more advanced methods could be studied.

Acknowledgements. This work is supported by the EU Horizon 2020 ITN/ETN project on Domain Specific Systems for Information Extraction and Retrieval (H2020-EU.1.3.1., ID: 860721).

REFERENCES

[1] Sophia Althammer, Sebastian Hofstätter, Suzan Verberne, and Allan Hanbury. 2022. TripJudge: A Relevance Judgement Test Collection for TripClick Health Retrieval. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management* (Atlanta, GA, USA) *(CIKM '22)*. Association for Computing Machinery, New York, NY, USA, 3801–3805. https://doi.org/10.1145/3511808.3557714

[2] Jordan T Ash, Chicheng Zhang, Akshay Krishnamurthy, John Langford, and Alekh Agarwal. 2019. Deep batch active learning by diverse, uncertain gradient lower bounds. *arXiv preprint arXiv:1906.03671* (2019).

[3] Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew Mcnamara, Bhaskar Mitra, and Tri Nguyen. 2016. MS MARCO : A Human Generated MAchine Reading COmprehension Dataset. In *Proc. of NIPS*.

[4] Christopher JC Burges. 2010. From ranknet to lambdarank to lambdamart: An overview. *MSR-Tech Report* (2010).

[5] Peng Cai, Wei Gao, Aoying Zhou, and Kam-Fai Wong. 2011. Relevant Knowledge Helps in Choosing Right Teacher: Active Query Selection for Ranking Adaptation. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Beijing, China) *(SIGIR '11)*. Association for Computing Machinery, New York, NY, USA, 115–124. https://doi.org/10.1145/2009916.2009935

[6] Aleksandr Chuklin, Ilya Markov, and Maarten de Rijke. 2015. *Click Models for Web Search*. Morgan & Claypool. https://doi.org/10.2200/S00654ED1V01Y201507ICR043

[7] Daniel Cohen, Bhaskar Mitra, Oleg Lesota, Navid Rekabsaz, and Carsten Eickhoff. 2021. Not all relevance scores are equal: Efficient uncertainty and calibration modeling for deep retrieval models. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 654–664.

[8] D. Cohn, Z. Ghahramani, and M. Jordan. 1996. Active Learning with statistical models. *Journal of Artificial Intelligence Research* 4 (1996), 129–145.

[9] Kevyn Collins-Thompson and Jamie Callan. 2007. Estimation and use of uncertainty in pseudo-relevance feedback. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*. 303–310.

[10] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, and Daniel Campos. 2019. Overview of the TREC 2019 deep learning track. In *TREC*.

[11] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, and Daniel Campos. 2020. Overview of the TREC 2020 Deep Learning Track. In *TREC*.

[12] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Jimmy Lin. 2021. MS MARCO: Benchmarking Ranking Models in the Large-Data Regime. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Virtual Event, Canada) *(SIGIR '21)*. Association for Computing Machinery, New York, NY, USA, 1566–1576. https://doi.org/10.1145/3404835.3462804

[13] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, Ellen M Voorhees, and Ian Soboroff. 2021. TREC Deep learning track: reusable test collections in the large data regime. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2369–2375.

[14] Fabio Crestani, Mounia Lalmas, and Cornelis Joost van Rijsbergen. 1998. *Information retrieval: Uncertainty and logics: Uncertainty and logics: Advanced models for the representation and retrieval of information*. Vol. 4. Springer Science & Business Media.

[15] Zhuyun Dai, Vincent Y Zhao, Ji Ma, Yi Luan, Jianmo Ni, Jing Lu, Anton Bakalov, Kelvin Guu, Keith Hall, and Ming-Wei Chang. 2023. Promptagator: Few-shot Dense Retrieval From 8 Examples. In *The Eleventh International Conference on Learning Representations*. https://openreview.net/forum?id=gmL46YMpu2J

[16] 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)*. Association for Computational Linguistics, Minneapolis, Minnesota, 4171–4186. https://doi.org/10.18653/v1/N19-1423

[17] Pinar Donmez and Jaime G. Carbonell. 2008. Optimizing estimated loss reduction for active sampling in rank learning. In *ICML*. 248–255. https://doi.org/10.1145/1390156.1390188

[18] Pinar Donmez and Jaime G. Carbonell. 2009. Active Sampling for Rank Learning via Optimizing the Area under the ROC Curve. In *Proceedings of the 31th European Conference on IR Research on Advances in Information Retrieval* (Toulouse, France) *(ECIR '09)*. Springer-Verlag, Berlin, Heidelberg, 78–89. https://doi.org/10.1007/978-3-642-00958-7_10

[19] Liat Ein-Dor, Alon Halfon, Ariel Gera, Eyal Shnarch, Lena Dankin, Leshem Choshen, Marina Danilevsky, Ranit Aharonov, Yoav Katz, and Noam Slonim. 2020. Active Learning for BERT: An Empirical Study. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Online, 7949–7962. https://doi.org/10.18653/v1/2020.emnlp-main.638

[20] Thibault Formal, Carlos Lassance, Benjamin Piwowarski, and Stéphane Clinchant. 2021. SPLADE v2: Sparse lexical and expansion model for information retrieval. *arXiv preprint arXiv:2109.10086* (2021).

[21] Thibault Formal, Benjamin Piwowarski, and Stéphane Clinchant. 2021. SPLADE: Sparse lexical and expansion model for first stage ranking. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2288–2292.

[22] Yoav Freund, H. Sebastian Seung, Eli Shamir, and Naftali Tishby. 1997. Selective Sampling Using the Query by Committee Algorithm. *Mach. Learn.* 28, 2-3 (1997), 133–168. http://dblp.uni-trier.de/db/journals/ml/ml28.html#FreundSST97

[23] Maik Fröbe, Christopher Akiki, Martin Potthast, and Matthias Hagen. 2022. How Train–Test Leakage Affects Zero-Shot Retrieval. In *String Processing and Information Retrieval*, Diego Arroyuelo and Barbara Poblete (Eds.). Springer International Publishing, Cham, 147–161.

[24] Yarin Gal and Zoubin Ghahramani. 2016. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48* (New York, NY, USA) *(ICML'16)*. JMLR.org, 1050–1059.

[25] Luyu Gao and Jamie Callan. 2021. Condenser: a Pre-training Architecture for Dense Retrieval. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Online and Punta Cana, Dominican Republic, 981–993. https://doi.org/10.18653/v1/2021.emnlp-main.75

[26] Maura R. Grossman and Gordon V. Cormack. 2011. Technology-Assisted Review in E-Discovery Can Be More Effective and More Efficient Than Exhaustive Manual Review. *Richmond Journal of Law and Technology* 17 (2011), 11.

[27] Prashansa Gupta and Sean MacAvaney. 2022. On Survivorship Bias in MS MARCO. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Madrid, Spain) *(SIGIR '22)*. Association for Computing Machinery, New York, NY, USA, 2214–2219. https://doi.org/10.1145/3477495.3531832

[28] Sebastian Hofstätter, Sophia Althammer, Mete Sertkan, and Allan Hanbury. 2022. Establishing Strong Baselines for TripClick Health Retrieval. In *Advances in Information Retrieval: 44th European Conference on IR Research, ECIR 2022, Stavanger, Norway, April 10–14, 2022, Proceedings, Part I*.

[29] Sebastian Hofstätter, Sheng-Chieh Lin, Jheng-Hong Yang, Jimmy Lin, and Allan Hanbury. 2021. Efficiently Teaching an Effective Dense Retriever with Balanced Topic Aware Sampling. In *Proc. of SIGIR*.

[30] Xiaomeng Hu, Shi Yu, Chenyan Xiong, Zhenghao Liu, Zhiyuan Liu, and Ge Yu. 2022. P3 Ranker: Mitigating the Gaps between Pre-Training and Ranking Fine-Tuning with Prompt-Based Learning and Pre-Finetuning. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Madrid, Spain) *(SIGIR '22)*. Association for Computing Machinery, New York, NY, USA, 1956–1962. https://doi.org/10.1145/3477495.3531786

[31] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data* 7, 3 (2019), 535–547.

[32] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense Passage Retrieval for Open-Domain Question Answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Online, 6769–6781. https://doi.org/10.18653/v1/2020.emnlp-main.550

[33] Omar Khattab and Matei Zaharia. 2020. ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT. In *Proc. of SIGIR*.

[34] Oleg Lesota, Navid Rekabsaz, Daniel Cohen, Klaus Antonius Grasserbauer, Carsten Eickhoff, and Markus Schedl. 2021. A modern perspective on query likelihood with deep generative retrieval models. In *Proceedings of the 2021 ACM SIGIR International Conference on Theory of Information Retrieval*. 185–195.

[35] David D. Lewis and Jason Catlett. 1994. Heterogeneous Uncertainty Sampling for Supervised Learning. In *Machine Learning Proceedings 1994*, William W. Cohen and Haym Hirsh (Eds.). Morgan Kaufmann, San Francisco (CA), 148–156. https://doi.org/10.1016/B978-1-55860-335-6.50026-X

[36] David D. Lewis and William A. Gale. 1994. A Sequential Algorithm for Training Text Classifiers. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (Dublin, Ireland) *(SIGIR '94)*. Springer-Verlag, Berlin, Heidelberg, 3–12.

[37] Jimmy Lin, Rodrigo Nogueira, and Andrew Yates. 2021. Pretrained transformers for text ranking: Bert and beyond. *Synthesis Lectures on Human Language Technologies* 14, 4 (2021), 1–325.

[38] Bo Long, Jiang Bian, Olivier Chapelle, Ya Zhang, Yoshiyuki Inagaki, and Yi Chang. 2015. Active Learning for Ranking through Expected Loss Optimization. *IEEE Transactions on Knowledge and Data Engineering* 27, 5 (2015), 1180–1191. https://doi.org/10.1109/TKDE.2014.2365785

[39] Yu Malkov and Dmitry Yashunin. 2016. Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence* PP (03 2016). https://doi.org/10.1109/TPAMI.2018.2889473

[40] Katerina Margatina, Giorgos Vernikos, Loïc Barrault, and Nikolaos Aletras. 2021. Active Learning by Acquiring Contrastive Examples. In *Proceedings of the 2021*

Conference on Empirical Methods in Natural Language Processing. 650–663.

[41] Gordon V. Cormack Maura R. Grossman and Adam Roegiest. 2016. Trec 2016 total recall track overview. In *TREC*.

[42] Iurii Mokrii, Leonid Boytsov, and Pavel Braslavski. 2021. A Systematic Evaluation of Transfer Learning and Pseudo-Labeling with BERT-Based Ranking Models. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Virtual Event, Canada) *(SIGIR '21)*. Association for Computing Machinery, New York, NY, USA, 2081–2085. https://doi.org/10.1145/3404835.3463093

[43] Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage Re-ranking with BERT. (2019). https://doi.org/10.48550/ARXIV.1901.04085

[44] Rodrigo Nogueira, Zhiying Jiang, Ronak Pradeep, and Jimmy Lin. 2020. Document Ranking with a Pretrained Sequence-to-Sequence Model. In *Findings of the Association for Computational Linguistics: EMNLP 2020*. Association for Computational Linguistics, Online, 708–718. https://doi.org/10.18653/v1/2020.findings-emnlp.63

[45] Navid Rekabsaz, Oleg Lesota, Markus Schedl, Jon Brassey, and Carsten Eickhoff. 2021. *TripClick: The Log Files of a Large Health Web Search Engine*. Association for Computing Machinery, New York, NY, USA, 2507–2513. https://doi.org/10.1145/3404835.3463242

[46] Nima Sadri and Gordon V. Cormack. 2022. Continuous Active Learning Using Pretrained Transformers. (2022). https://doi.org/10.48550/ARXIV.2208.06955

[47] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108* (2019).

[48] Harrisen Scells, Shengyao Zhuang, and Guido Zuccon. 2022. Reduce, Reuse, Recycle: Green Information Retrieval Research. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2825–2837.

[49] H. S. Seung, M. Opper, and H. Sompolinsky. 1992. Query by Committee. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory* (Pittsburgh, Pennsylvania, USA) *(COLT '92)*. Association for Computing Machinery, New York, NY, USA, 287–294. https://doi.org/10.1145/130385.130417

[50] Xuehua Shen and ChengXiang Zhai. 2005. Active Feedback in Ad Hoc Information Retrieval. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (Salvador, Brazil) *(SIGIR '05)*. Association for Computing Machinery, New York, NY, USA, 59–66. https://doi.org/10.1145/1076034.1076047

[51] Xuehua Shen and ChengXiang Zhai. 2005. Active Feedback in Ad Hoc Information Retrieval. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (Salvador, Brazil) *(SIGIR '05)*. Association for Computing Machinery, New York, NY, USA, 59–66. https://doi.org/10.1145/1076034.1076047

[52] Rodrigo Silva, Marcos Gonçalves, and Adriano Veloso. 2014. A Two-Stage Active Learning Method for Learning to Rank. *Journal of the Association for Information Science and Technology* 65 (01 2014). https://doi.org/10.1002/asi.22958

[53] Nandan Thakur, Nils Reimers, Johannes Daxenberger, and Iryna Gurevych. 2021. Augmented SBERT: Data Augmentation Method for Improving Bi-Encoders for Pairwise Sentence Scoring Tasks. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 296–310.

[54] Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. BEIR: A Heterogeneous Benchmark for Zero-shot Evaluation of Information Retrieval Models. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*. https://openreview.net/forum?id=wCu6T5xFjeJ

[55] Nicola Tonellotto. 2022. Lecture Notes on Neural Information Retrieval. *arXiv preprint arXiv:2207.13443* (2022).

[56] Howard R Turtle and W Bruce Croft. 1997. Uncertainty in information retrieval systems. *Uncertainty management in information systems: from needs to solutions* (1997), 189–224.

[57] Jun Wang and Jianhan Zhu. 2009. Portfolio theory of information retrieval. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*. 115–122.

[58] Kexin Wang, Nandan Thakur, Nils Reimers, and Iryna Gurevych. 2021. GPL: Generative Pseudo Labeling for Unsupervised Domain Adaptation of Dense Retrieval. In *Proceedings of NAACL 2021*.

[59] Shuai Wang, Harrisen Scells, Bevan Koopman, and Guido Zuccon. 2022. Neural Rankers for Effective Screening Prioritisation in Medical Systematic Review Literature Search. In *Proceedings of the 26th Australasian Document Computing Symposium (ADCS '22)*. Association for Computing Machinery.

[60] Ji Xin, Chenyan Xiong, Ashwin Srinivasan, Ankita Sharma, Damien Jose, and Paul Bennett. 2022. Zero-Shot Dense Retrieval with Momentum Adversarial Domain Invariant Representations. In *Findings of the Association for Computational Linguistics: ACL 2022*. Association for Computational Linguistics, Dublin, Ireland, 4008–4020. https://doi.org/10.18653/v1/2022.findings-acl.316

[61] Zuobing Xu, Ram Akella, and Yi Zhang. 2007. Incorporating Diversity and Density in Active Learning for Relevance Feedback. In *Proceedings of the 29th European Conference on IR Research* (Rome, Italy) *(ECIR'07)*. Springer-Verlag, Berlin, Heidelberg, 246–257.

[62] Eugene Yang, David D. Lewis, and Ophir Frieder. 2021. On Minimizing Cost in Legal Document Review Workflows. In *Proceedings of the 21st ACM Symposium on Document Engineering* (Limerick, Ireland) *(DocEng '21)*. Association for Computing Machinery, New York, NY, USA, Article 30, 10 pages. https://doi.org/10.1145/3469096.3469872

[63] Eugene Yang, Sean MacAvaney, David D. Lewis, and Ophir Frieder. 2022. Goldilocks: Just-Right Tuning of BERT for Technology-Assisted Review. In *Advances in Information Retrieval: 44th European Conference on IR Research, ECIR 2022, Stavanger, Norway, April 10–14, 2022, Proceedings, Part I* (Stavanger, Norway). Springer-Verlag, Berlin, Heidelberg, 502–517. https://doi.org/10.1007/978-3-030-99736-6_34

[64] Linjun Yang, Li Wang, Bo Geng, and Xian-Sheng Hua. 2009. Query Sampling for Ranking Learning in Web Search. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval* (Boston, MA, USA) *(SIGIR '09)*. Association for Computing Machinery, New York, NY, USA, 754–755. https://doi.org/10.1145/1571941.1572112

[65] Hwanjo Yu. 2005. SVM Selective Sampling for Ranking with Application to Data Retrieval. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining* (Chicago, Illinois, USA) *(KDD '05)*. Association for Computing Machinery, New York, NY, USA, 354–363. https://doi.org/10.1145/1081870.1081911

[66] Yue Yu, Lingkai Kong, Jieyu Zhang, Rongzhi Zhang, and Chao Zhang. 2021. AcTune: Uncertainty-aware Active Self-Training for Semi-Supervised Active Learning with Pretrained Language Models. *arXiv preprint arXiv:2112.08787* (2021).

[67] Michelle Yuan, Hsuan-Tien Lin, and Jordan Boyd-Graber. 2020. Cold-start active learning through self-supervised language modeling. *arXiv preprint arXiv:2010.09535* (2020).

[68] Jingtao Zhan, Xiaohui Xie, Jiaxin Mao, Yiqun Liu, Jiafeng Guo, Min Zhang, and Shaoping Ma. 2022. Evaluating Interpolation and Extrapolation Performance of Neural Retrieval Models. (2022). https://doi.org/10.48550/ARXIV.2204.11447

[69] Xinyu Zhang, Andrew Yates, and Jimmy Lin. 2020. A Little Bit Is Worse Than None: Ranking with Limited Training Data. In *Proceedings of SustaiNLP: Workshop on Simple and Efficient Natural Language Processing*. Association for Computational Linguistics, Online, 107–112. https://doi.org/10.18653/v1/2020.sustainlp-1.14

[70] Jingbo Zhu, Huizhen Wang, Tianshun Yao, and Benjamin K Tsou. 2008. Active Learning with Sampling by Uncertainty and Density for Word Sense Disambiguation and Text Classification. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*. Coling 2008 Organizing Committee, Manchester, UK, 1137–1144. https://aclanthology.org/C08-1143

[71] Jianhan Zhu, Jun Wang, Ingemar J Cox, and Michael J Taylor. 2009. Risky business: modeling and exploiting uncertainty in information retrieval. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*. 99–106.

[72] Guido Zuccon, Leif A Azzopardi, and Keith Van Rijsbergen. 2009. The quantum probability ranking principle for information retrieval. In *Advances in Information Retrieval Theory: Second International Conference on the Theory of Information Retrieval, ICTIR 2009 Cambridge, UK, September 10-12, 2009 Proceedings 2*. Springer, 232–240.