Training Curricula for Open Domain Answer Re-Ranking

Sean MacAvaney
IR Lab, Georgetown University, USA
sean@ir.cs.georgetown.edu

Franco Maria Nardini
ISTI-CNR, Pisa, Italy
francomaria.nardini@isti.cnr.it

Raffaele Perego
ISTI-CNR, Pisa, Italy
raffaele.perego@isti.cnr.it

Nicola Tonellotto
University of Pisa, Italy
nicola.tonellotto@unipi.it

Nazli Goharian
IR Lab, Georgetown University, USA
nazli@ir.cs.georgetown.edu

Ophir Frieder
IR Lab, Georgetown University, USA
ophir@ir.cs.georgetown.edu

ABSTRACT

In precision-oriented tasks like answer ranking, it is more important to rank many relevant answers highly than to retrieve *all* relevant answers. It follows that a good ranking strategy would be to learn how to identify the easiest correct answers first (i.e., assign a high ranking score to answers that have characteristics that usually indicate relevance, and a low ranking score to those with characteristics that do not), before incorporating more complex logic to handle difficult cases (e.g., semantic matching or reasoning). In this work, we apply this idea to the training of neural answer rankers using curriculum learning. We propose several heuristics to estimate the difficulty of a given training sample. We show that the proposed heuristics can be used to build a training curriculum that down-weights difficult samples early in the training process. As the training process progresses, our approach gradually shifts to weighting all samples equally, regardless of difficulty. We present a comprehensive evaluation of our proposed idea on three answer ranking datasets. Results show that our approach leads to superior performance of two leading neural ranking architectures, namely BERT and ConvKNRM, using both pointwise and pairwise losses. When applied to a BERT-based ranker, our method yields up to a 4% improvement in MRR and a 9% improvement in P@1 (compared to the model trained without a curriculum). This results in models that can achieve comparable performance to more expensive state-of-the-art techniques.

ACM Reference Format:
Sean MacAvaney, Franco Maria Nardini, Raffaele Perego, Nicola Tonellotto, Nazli Goharian, and Ophir Frieder. 2020. Training Curricula for Open Domain Answer Re-Ranking. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '20), July 25–30, 2020, Virtual Event, China.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3397271.3401094

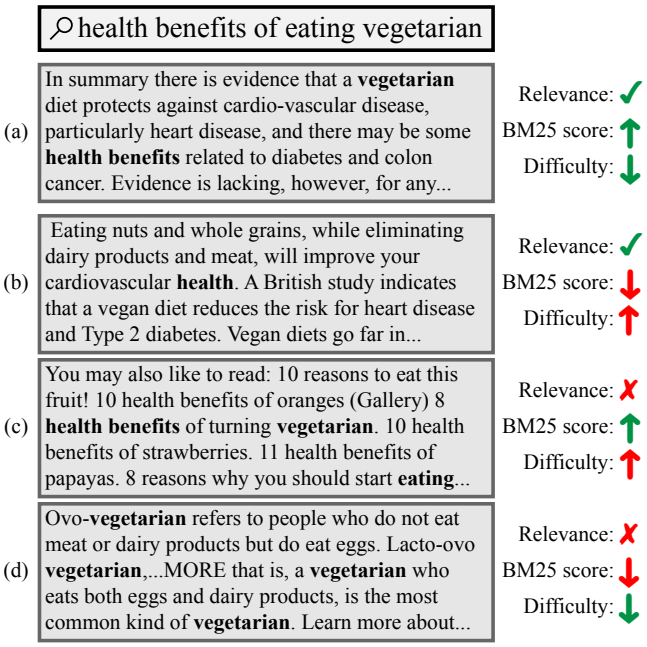

Figure 1: Example of curriculum approach from MS-MARCO dataset (question ID 199776). In this example, we predict (a) is 'easy' because it is relevant and has a high BM25 score. (d) is likewise 'easy' weight because it is non-relevant and has a low score. (b) is a 'difficult' sample because is relevant, yet has a low score due to the few term matches. We also predict (c) to be 'difficult' because it is non-relevant, yet it has a high score. Our approach begins by weighting 'easy' training samples high and 'difficult' training samples low.

1 INTRODUCTION

Deep learning techniques are of recent interest to solve information retrieval tasks such as answer ranking [26]. Most of such work focuses on designing neural network architectures that are effective at predicting answer relevance to a particular question, while comparatively little attention aims to find optimal training configurations for these networks. More so, existing literature often falls short of expressing the most basic settings of the training environment, such as the choice of the loss function and training sample selection procedures, two critical components needed for successful reproduction of results. In contrast, we focus on the neural rankers training process for information retrieval. In particular, we demonstrate that weighting training examples early in the learning process can yield significant benefits in the effectiveness of the neural ranker.

SIGIR '20, July 25–30, 2020, Virtual Event, China

ACM ISBN 978-1-4503-8016-4/20/07…$15.00
https://doi.org/10.1145/3397271.3401094

We motivate our approach with the simple intuition that some answers are easier to assess the relevance of than others. For instance, consider a question about the health impacts of vegetarianism (see Figure 1). A passage written explicitly about this topic (e.g., (a)) should be relatively straightforward to identify, as it uses many of the terms in the question. This likely yields a high ranking position using conventional probabilistic approaches, such as BM25. A passage written about the health benefits of *veganism* (a more strict version of vegetarianism) may also answer the question (b). However, it involves more complicated reasoning and inference (such as the understanding of the relationship between the two diets) and semantic understanding of the way in which the content is presented. Similarly, consider two non-relevant answers: one that matches most of the query terms (c) and one that does not (d). We argue that the former is more difficult for the ranking model to identify as non-relevant due to the large number of matching terms, and the latter is easier due to critical missing terms (e.g., health benefits).

While an ideal ranker would rank both (a) and (b) high, doing so we may add noise and complexity to the model that reduces the overall quality of ranking. Specifically, ranking (b) high may make it more difficult to identify (c) and (d) as non-relevant. Our method attempts to overcome this issue by forcing the ranker to focus primarily on the "easy" training samples before gradually moving on to learning to rank all training samples via training sample weighting.

We formulate this idea using the *curriculum learning* (CL) framework [1]. Learning through a curriculum is an idea borrowed from cognitive sciences according to which the learning process follows a multi-step training path. Initially, the learning algorithm is trained by using simple examples and smooth loss functions. Then it is progressively fine-tuned so as to deal with examples and loss functions of increasing complexity. We instantiate the CL framework in the learning to rank domain by assigning different weights to training samples through a heuristic. In early stages of training, high weights are assigned to *easy* training pairs while *difficult* samples are given low weights. As training progresses, we gradually smooth this imbalance. Eventually, all training samples are weighted equally, regardless of the estimated difficulty.

To estimate the difficulty of question-answer pairs and to choose the training weight accordingly, we consider both information from an unsupervised baseline ranker (e.g., BM25) and the human-assessed relevance of the answer to the given question (see Figure 1). When an unsupervised ranker is able to identify the example effectively (i.e., it ranks a relevant document high or a non-relevant document low) the training sample is considered as "easy". On the other hand, when the unsupervised ranker fails to correctly score them, the sample is considered as "difficult".

We show that our approach can be easily integrated into a neural ranking pipeline. We validate our approach using three weighting heuristics based on an unsupervised ranker using two leading neural ranking methods (BERT [9] and ConvKNRM [7]). Our code is available for reproducibility.[1] Our results show significant ranking improvements when tested on three open-domain (i.e., not domain-specific) answer ranking benchmarks: TREC Deep Learning (DL), TREC Complex Answer Retrieval (CAR), and ANTIQUE. These datasets vary in scale (hundreds of thousands of answers to tens of millions) and source of relevance information (graded or positive-only, human-annotated or inferred from document structure). We test using both pointwise and pairwise losses. In summary, our contributions are:

- We propose a curriculum learning scheme for open-domain answer re-ranking.
- We propose and evaluate three heuristics for weighting training samples while learning neural rankers, which utilize the ranking and score of the first-stage ranker.
- We provide a comprehensive analysis of our proposed approaches for curriculum learning of neural rankers. Our results show the superiority of our approach as compared to standard weighting of training samples.
- We show that our proposed curricula are effective on three answer re-ranking datasets. On TREC DL, our approach yields up to a 3.6% improvement in MRR and a 9.3% improvement in P@1 for a BERT-based ranker. For TREC CAR, the curricula yield a 4.2% and 3.7% boost to R-Precision and MAP, respectively, and achieves comparable performance to a larger version of BERT. For ANTIQUE, our approach yields a 3.4% and 6.0% improvement in terms of MRR and P@1.

2 BACKGROUND & RELATED WORK

In this section, we provide background information about neural ranking approaches (Section 2.1) and prior work on curriculum learning (Section 2.2).

2.1 Neural ranking

An ad-hoc neural ranking model maps a given query-document pair to a real-valued relevance score given the model parameters. For answer ranking, the question is treated as the query and answers are treated as the document. Through a set of training data, the parameters are optimized to maximize ranking performance on unseen data. Many neural architectures exist. They can be broadly classified as either *representation-focused* or *interaction-focused* [12].

Representation-focused models (also referred to as semantic matching models) learn mapping functions for the query and the document to a dense representation, and then compare these representations using a relatively simple comparison function (e.g., cosine similarity). Examples include DSSM [16], C-DSSM [36], and ARC-I [14]. These models rely on massive amounts of data (e.g., behavioral information from a query log) to learn semantic similarities. We do not focus on representation-focused models because of their general reliance on proprietary query logs, and their underperformance on standard test collections.

On the other hand, interaction-focused models (also referred to as relevance matching models) explicitly model patterns of query term occurrences in the document. DRMM [12] models this concept by building a query-document similarity matrix where each cell represents the cosine similarity score between embeddings of each query term and document term. This allows the model to capture soft semantic term similarity (i.e., semantically-similar terms have a high score, and semantically-dissimilar terms have a low score). DRMM then models the term occurrences by building a histogram

[1] https://github.com/Georgetown-IR-Lab/curricula-neural-ir

based on the similarity score for each query term and by producing a relevance score by feeding these histograms into a multi-layer perceptron. KNRM [37] works similarly, but replaces the hard histogram buckets with soft Gaussian-kernel-based buckets. Other approaches model term adjacency, such as MatchPyramid [30], DeepRank [31], PACRR [17], and ConvKNRM [7].

Contrary to recent critiques of the actual effectiveness of neural ranking architectures [21, 39], recent work with transformer-based contextualized language models (e.g., BERT [9]) on document and answer ranking have shown clear ranking superiority over prior baselines [23, 28, 29]. These methods exploit the distributional characteristics of a language learned through pre-training a model on tasks with more data available (e.g., masked language model and a next sentence prediction). Due to the self-attention mechanism of transformers, these models can also be considerd interaction-focused. MacAvaney et al. [23] further demonstrated that signals from contextualized language models can be incorporated into other interaction-focused neural ranking architectures, boosting ranking performance beyond both transformer-based rankers and the non-contextualized interaction models.

2.2 Curriculum Learning

Curriculum Learning (CL) can be considered a particular case of *Continuation Methods*, generally used when the target objective function is non-convex and its direct optimization may lead the training to converge to a poor local minimum [1, 4]. The basic idea to overcome this problem through a curriculum approach is to organize the learning process as a path where the easiest training samples are presented first and the complexity of the following ones is gradually increased. This strategy allows the learner to exploit previously seen concepts to ease the acquisition of subsequent more difficult ones. CL approaches are proved successful for training neural networks in different domains such as NLP [5, 15], language models (not used for ranking tasks) [1], image representation [3], network representation [33]. To our knowledge, the only attempt to explore how CL methods can be exploited in the document ranking domain is the one by Ferro *et al.* [11] where authors exploit the curriculum learning strategy in a gradient boosting algorithm that learns ranking models based on ensembles of decision trees. The results reported by Ferro *et al.* show that a curriculum learning strategy gives only a limited boost to the ranking performance of an ensemble of decision trees. Similar to our approach, Fidelity-weighted learning [8] applies weights to training samples for learning ranking models. However, this approach focuses on estimating the quality of weak labels (namely, treating BM25 scores as labels), rather than the difficulty of training samples with higher-quality labels (e.g., human-annotated labels).

Sachan and Xing [34] propose curriculum learning approaches for question answering, but in a closed-domain setting. In open-domain question answering, there are several challenges encountered, including that there is a much larger collection of answers to draw from (millions of answers) and multiple correct answers to a given question. Thus, we tackle this problem from an IR-perspective, utilizing signals from ranking models. Recently, Penha and Hauff [32] propose approaches for using curriculum learning to rank conversational responses, yielding up to a 2% improvement in ranking effectiveness. The curricula proposed are specific to the domain of conversational responses and are non-trivial to apply to other domains. In contrast, we propose simple heuristics based on initial retrieval ranks and scores, and we show their effectiveness across multiple ranking models, loss functions, and answer ranking datasets in an open-domain setting.

Table 1: Table of symbols.

Symbol	Definition
R_θ	Neural ranking function with parameters θ
$\mathcal{L}$	Loss function
$\mathcal{D}$	Training sample difficulty function
W	Training sample weight function
$\mathbf{q}$	Query (i.e., question)
$\mathbf{d}$	Document (i.e., answer)
$\mathbf{d}^+$	Relevant document
$\mathbf{d}^-$	Non-relevant document
$\mathbf{D}$	Set of ranked documents
s	Manual relevance assessment score
T	Collection of training data
t	Training sample from T
i	Training iteration (epoch)
m	End of curriculum iteration (hyperparameter)

3 METHODOLOGY

We present our approach for applying curriculum learning to the training of neural rankers. At a high level, our approach applies a heuristic to determine the difficulty of a particular training sample. This difficulty estimation is then used for weighting training samples. In early stages of training, samples that the heuristic predicts as easy are given a higher weight, while samples predicted to be difficult are given a lower weight. Gradually, our approach eases off this crutch (controlled by a new hyper-parameter). Eventually, all training samples are weighted equally, regardless of the estimated difficulty.

Our approach allows for fair comparisons to an unmodified training process because no changes are made to the selection of the training data itself; the effect is only on the weight of the sample during training. Furthermore, this allows for an easy integration of our approach into existing training pipelines; no changes to the data selection technique are required, and the heuristics rely on information readily available in most re-ranking settings.

Our approach degrades into the typical training process in two ways: either (1) a heuristic can be used that gives every sample a weight of 1, or (2) the hyper-parameter that drives the degradation of the approach to equal weighting can be set to immediately use equal weights.

3.1 Notation and preliminaries

A summary of the symbols used is given in Table 1. Let an ad-hoc neural ranking model be represented as $R_\theta(\mathbf{q}, \mathbf{d}) \in \mathbb{R}$, which maps a given query-document pair $(\mathbf{q}, \mathbf{d})$ to a real-valued relevance score given the model parameters θ. For simplicity, we refer to questions as queries and answers as documents. Through a set of training data points $t \in T$ and a loss function $\mathcal{L}(t)$, the model parameters θ

are optimized to maximize the ranking performance. The training data sample $t \in T$ depends on the type of loss employed. Two common techniques employed for training neural rankers rely on pointwise or pairwise loss. For pointwise loss, training data consists of triples $t_{point} = \langle \mathbf{q}, \mathbf{d}, s \rangle$, where $\mathbf{q}$ is a query, $\mathbf{d}$ is a document, and s is its relevance score, e.g., the relevance score given to the query-document pair by a human assessor. The loss for this sample often uses squared error between the predicted score and the relevance score s:

$$\mathcal{L}^{point}(\mathbf{q}, \mathbf{d}, s) = \big(s - R_\theta(\mathbf{q}, \mathbf{d})\big)^2 \tag{1}$$

On the other hand, pairwise loss uses two document samples for the same query (one relevant and one non-relevant), and optimizes to assign a higher score to the relevant document than the non-relevant one. Training triples for pairwise loss are represented as $t_{pair} = \langle \mathbf{q}, \mathbf{d}^+, \mathbf{d}^- \rangle$, where $\mathbf{q}$ is the query, $\mathbf{d}^+$ is the relevant document, and $\mathbf{d}^-$ is the non-relevant document. One common pairwise loss function is the softmax cross-entropy loss:

$$\mathcal{L}^{pair}(\mathbf{q}, \mathbf{d}^+, \mathbf{d}^-) = \frac{\exp\big(R_\theta(\mathbf{q}, \mathbf{d}^+)\big)}{\exp\big(R_\theta(\mathbf{q}, \mathbf{d}^+)\big) + \exp\big(R_\theta(\mathbf{q}, \mathbf{d}^-)\big)} \tag{2}$$

3.2 Curriculum framework for answer ranking

Let a difficulty function $\mathcal{D} : T \mapsto [0, 1]$ define a weight $\mathcal{D}(t)$ for the training sample $t \in T$. Without loss of generality we now assume that a high value of $\mathcal{D}(t)$, i.e., a value close to 1, represents an easy sample, while a low value, i.e., a value close to 0, represents a difficult sample. Note that the heuristic $\mathcal{D}(t)$ necessarily depends on the type of loss function employed: for pointwise loss, it estimates the difficulty for assigning the relevance score s to $\langle \mathbf{q}, \mathbf{d} \rangle$, while, for pairwise loss, it estimates the difficulty of scoring the relevant document pair $\langle \mathbf{q}, \mathbf{d}^+ \rangle$ above the non-relevant pair $\langle \mathbf{q}, \mathbf{d}^- \rangle$.

In our CL framework, during the first learning iteration, training samples are weighted according only to the difficulty function. To ease into the difficult samples, we employ a hyper-parameter m, which represents the training iteration at which to start to give every training sample equal weights.[2] Between the start of training and the mth training iteration, we linearly degrade the importance of the difficulty heuristic. More formally, we define the iteration-informed training sample weight $W_{\mathcal{D}}(t, i)$ given the training iteration i (0-based) as:

$$W_{\mathcal{D}}(t, i) = \begin{cases} \mathcal{D}(t) + \frac{i}{m}\big(1 - \mathcal{D}(t)\big) & i < m \\ 1 & i \geq m \end{cases} \tag{3}$$

We then define a new $\mathcal{D}$-informed loss function by including the iteration-informed weight into the standard pointwise or pairwise loss function:

$$\mathcal{L}_{\mathcal{D}}(t, i) = W_{\mathcal{D}}(t, i)\, \mathcal{L}(t) \tag{4}$$

3.3 Difficulty heuristics

In a re-ranking setting, a simple source of difficulty information can come from the initial ranking of the documents. Probability ranking models like BM25 rely on term frequency and inverse document frequency to score documents. These characteristics should generally be easy for models to learn because they can learn to identify term frequency information (either directly, as is done by models like DRMM and KNRM, or implicitly, as is done by models like BERT through self-attention) and inverse document frequency, e.g., by down-weighting the importance of frequent terms. We postulate that it is inherently more difficult to perform semantic matching needed for identifying documents that have lower initial ranking scores. These scores are also easy to obtain, as they are readily available in a re-ranking setting. Thus, we use unsupervised ranking scores as the basis for our curriculum learning heuristics.

Reciprocal rank heuristic. We define $\mathcal{D}_{recip}$ as a measure of difficulty from the reciprocal of the rank at which answers appear in a ranked list. We assume that an answer placed higher compared to the other retrieved answers is "easier" for the ranker to place in that position. A high rank makes relevant documents easier and non-relevant documents harder. In the pointwise setting, relevant documents with a high reciprocal rank are considered "easier" than relevant documents with a low reciprocal rank because the unsupervised ranker assigned a higher score. Conversely, non-relevant documents with a high rank are considered "harder" than samples that are assigned a low rank. Given $\mathbf{d}$ from a set of ranked documents $\mathbf{D}$ for query $\mathbf{q}$ we have:

$$recip_{\mathbf{q},\mathbf{D}}(\mathbf{d}) = \frac{1}{rank_{\mathbf{q},\mathbf{D}}(\mathbf{d})} \tag{5}$$

With these conditions in mind, we define $\mathcal{D}_{recip}$ for pointwise loss as:

$$\mathcal{D}^{point}_{recip}(\mathbf{q}, \mathbf{d}, s) = \begin{cases} recip_{\mathbf{q},\mathbf{D}}(\mathbf{d}) & s > 0 \quad \triangleright \textit{ relevant} \\ 1 - recip_{\mathbf{q},\mathbf{D}}(\mathbf{d}) & s \leq 0 \quad \triangleright \textit{ non-relevant} \end{cases} \tag{6}$$

For pairwise loss, we define pairs that have a large difference between the reciprocal ranks to be very difficult (when the non-relevant document is higher) or very easy (when the relevant document is higher). When the reciprocal ranks are similar, we define the difficulty as moderate, with a difficulty close to 0.5. This is accomplished by taking the difference between the scores and scaling the result within the range $[0, 1]$:

$$\mathcal{D}^{pair}_{recip}(\mathbf{q}, \mathbf{d}^+, \mathbf{d}^-) = \frac{recip_{\mathbf{q},\mathbf{D}}(\mathbf{d}^+) - recip_{\mathbf{q},\mathbf{D}}(\mathbf{d}^-) + 1}{2} \tag{7}$$

Normalized score heuristic. An alternative to using the ranks of documents by an unsupervised ranker is to use the scores from these rankers. We define $\mathcal{D}_{norm}$ as a measure of difficulty that uses the ranking score information. This allows documents that receive similar (or identical) scores to be considered similarly (or identically) in terms of difficulty. In the case of identical scores, $\mathcal{D}_{norm}$ allows to improve the reproducibility of the CL approach compared to curricula that rely on rank [22]. To account for various ranges in which ranking scores can appear, we apply min-max normalization by query to fit all scores into the $[0, 1]$ interval, eliminating per-query score characteristics. The integration of the normalized score $norm_{\mathbf{q},\mathbf{D}}(\mathbf{d})$ into both pointwise and pairwise rankers are similar to that of the reciprocal rank curriculum:

$$\mathcal{D}^{point}_{norm}(\mathbf{q}, \mathbf{d}, s) = \begin{cases} norm_{\mathbf{q},\mathbf{D}}(\mathbf{d}) & s > 0 \quad \triangleright \textit{ relevant} \\ 1 - norm_{\mathbf{q},\mathbf{D}}(\mathbf{d}) & s \leq 0 \quad \triangleright \textit{ non-relevant} \end{cases} \tag{8}$$

$$\mathcal{D}^{pair}_{norm}(\mathbf{q}, \mathbf{d}^+, \mathbf{d}^-) = \frac{norm_{\mathbf{q},\mathbf{D}}(\mathbf{d}^+) - norm_{\mathbf{q},\mathbf{D}}(\mathbf{d}^-) + 1}{2} \tag{9}$$

[2] We explore the importance of eventually converging to equal weights in Section 4.4.

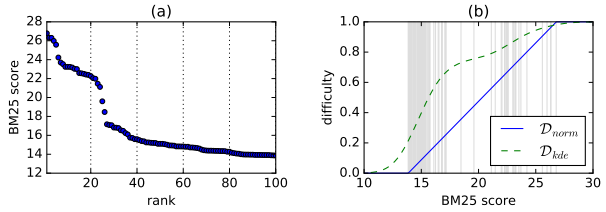

Figure 2: (a) Example of BM25 scores exhibiting non-linear behavior; there are several answers with a much higher score than others and a long tail of lower-scored answers. (b) Comparison between normalized score (solid blue) and KDE (dashed green) heuristic values by BM25 score. The grey vertical lines indicate the values from the initial ranking (from (a)). Scores are from MS-MARCO query 1000009 retrieved using Anserini's [38] BM25 implementation.

Kernel Density Estimation (KDE) heuristic. The normalized score heuristic provides weighting based on ranking score, but it fails to acknowledge an important characteristic of ranking score values: they are often non-linear. For example, it is common for a handful of scores to be comparatively very high, with a long tail of lower scored answers (e.g., with fewer query term matches, see Figure 2(a)). We hypothesize that it may be valuable to provide a greater degree of value distinction between scores in areas of high score density (e.g., in the long tail, around a score of 16 and below in Figure 2(a)) and areas with relatively low score density (e.g., around a score of 20). To this end, we construct a Gaussian Kernel Density Estimation (KDE), with the bandwidth selected using Scott's Rule [35]. We then define $\mathcal{D}_{kde}$ by using the CDF of the kernel as the basis of difficulty measure:

$$\mathcal{D}^{point}_{kde}(\mathbf{q}, \mathbf{d}, s) = \begin{cases} KDE_{\mathbf{q},\mathbf{D}}(\mathbf{d}) & s > 0 \quad \triangleright \textit{ relevant} \\ 1 - KDE_{\mathbf{q},\mathbf{D}}(\mathbf{d}) & s \leq 0 \quad \triangleright \textit{ non-relevant} \end{cases} \tag{10}$$

$$\mathcal{D}^{pair}_{kde}(\mathbf{q}, \mathbf{d}^+, \mathbf{d}^-) = \frac{KDE_{\mathbf{q},\mathbf{D}}(\mathbf{d}^+) - KDE_{\mathbf{q},\mathbf{D}}(\mathbf{d}^-) + 1}{2} \tag{11}$$

where $KDE_{\mathbf{q},\mathbf{D}}(\mathbf{d})$ yields the CDF score of the kernel density estimation for $\mathbf{d}$. An example of the difference between $\mathcal{D}_{norm}$ and $\mathcal{D}_{kde}$ for a particular query is shown in Figure 2(b). This approach has the added benefit of allowing a non-zero difficulty for positive samples that are not retrieved in the ranking list.

In summary, we propose a curriculum learning approach for answer ranking. The approach weights training samples by predicted difficulty. We propose three heuristics for estimating training sample difficulty, based on the rank or score of an unsupervised ranking model.

4 EXPERIMENTS

We conduct experiments on three large-scale answer ranking datasets — namely TREC Deep Learning (DL) [6] (Section 4.1), TREC Complex Answer Retrieval (CAR) [10] (Section 4.2), and ANTIQUE [13] (Section 4.3) — and two neural rankers (Vanilla BERT [9, 23] and ConvKNRM [7]) to answer the following research questions:

- RQ1 Are the proposed training curricula effective for training neural rankers for answer ranking? (Sections 4.1–4.3)
- RQ2 Under which conditions is each curriculum more effective (e.g., amount and quality of training data, type of neural ranker trained, etc.)? (Sections 4.1–4.3)
- RQ3 Is it important to shift to difficult samples, or can a ranker be successfully trained focusing only on easy samples? (Section 4.4)
- RQ4 Is focusing on the easy samples first more beneficial to training than focusing on the hardest samples first? (Section 4.5)

Each dataset exhibits different characteristics (summarized in Table 2), as do the neural ranking architectures employed:

- **"Vanilla" BERT** [23]. This model uses the sentence classification mechanism from a pretrained BERT contextualized model [9] (a deep transformer-based network) to model the semantic relevance between the question and answer. This model yields exceptional ranking performance at the expense of computational cost and is the foundation for most state-of-the-art answer ranking approaches. We test Vanilla BERT using both pointwise and pairwise loss, as defined in Section 3.1. In line with [23], we initialize the model using `bert-base` (12-layer transformer pretrained on English text from [9]) and fine-tune using a learning rate of 2×10^{-5} with the Adam optimizer.
- **ConvKNRM** [7]. This model learns the relationship between unigram and n-gram (via a convolutional neural network) similarity scores between the question and answer and combines the scores using Gaussian filters. This model yields competitive ranking performance and can be optimized for real-time ranking [18]. We use unigram to tri-gram encoding with cross-matching and 128 hidden nodes for score combination. Word vectors were initialized using 300-dimensional FastText [2] word embeddings trained on WikiNews with subword information. Based on preliminary experiments that showed that the ConvKNRM model fails to converge when trained using pointwise loss, we only test using pairwise loss. We train the model using the Adam optimizer and a learning rate of 10^{-3}. Furthermore, we use the score additivity technique from [39].

We train each model using training iterations consisting of 32 batches of 16 training samples uniformly selected over the re-ranking pool. We employ gradient accumulation when a training batch is unable to fit on a GPU (e.g., Vanilla BERT models). After each training iteration, the validation performance is assessed. We employ early stopping after 15 consecutive epochs with no improvement to the dataset-dependent validation metric. When training is early stopped, the model is rolled back to the version of that achieved a performance improvement. This yielded up to 130 training iterations. We test our three proposed training curricula ($\mathcal{D}_{recip}$, $\mathcal{D}_{norm}$, and $\mathcal{D}_{kde}$) on each of the datasets and neural rankers. We optimize the parameter m i.e., end of curriculum learning epoch, by fine-tuning on the validation set. For each dataset, ranker, and loss combination, we test $m \in \{1, 5, 10, 20, 50, 100\}$. To put performance of the neural rankers in context, we include the ranking effectiveness of Anserini's [38] implementation of BM25 and SDM [25], both with default parameters, tuned on the validation set ('Tuned'), and tuned on the test set (representing the optimal

Table 2: Dataset statistics. The values in parentheses indicate the average number of relevance judgments per query.

Dataset	# Answers	Train Queries (judg. per query)	Validation Queries (judg. per query)	Test Queries (judg. per query)	Test Judgments
TREC DL [27]	8.8M	504k (1.1)	200 (0.7)	43 (215.3)	Human (graded)
TREC CAR [10]	30M	616k (4.8)	2.2k (5.5)	2.4k (6.7)	Inferred (positive only)
ANTIQUE [13]	404k	2.2k (11.3)	200 (11.0)	200 (33.0)	Human (graded)

settings of parameters for this model, 'Optimized').[3] We also include relevant prior reported results and the optimal re-ranking of the results (i.e., sorting the original ranking list by relevance score, serving as an upper bound to re-ranking performance).

4.1 Web passage answer ranking

We first demonstrate the effectiveness of our training curricula on the TREC Deep Learning (DL) 2019 answer passage ranking dataset, which uses the MS-MARCO collection and queries [27].

[3]Models tuned using a grid search: BM25 $k_1 \in [0.1, 4.0]$ by 0.1 and $b \in [0.0, 1.0]$ by 0.05; SDM term, ordered and unordered weights $\in [0, 1]$ by 0.1.

Table 3: Ranking performance on the TREC DL 2019 answer passage ranking task. Significant improvements in performance when using the training curricula (as compared to no curriculum) are indicated with ↑ (paired t-test $p < 0.05$). There are no statistically-significant differences among the curricula. The top result for each model are listed in bold.

			TREC DL 2019	
	Ranker	Training	MRR@10	P@1
	ConvKNRM	Pairwise	0.6159	0.4419
		w/ $\mathcal{D}_{recip}$	**0.6834**	**0.5581**
		w/ $\mathcal{D}_{norm}$	0.6514	0.5116
		w/ $\mathcal{D}_{kde}$	0.6475	0.5116
	Vanilla BERT	Pointwise	0.8740	0.7907
		w/ $\mathcal{D}_{recip}$	**0.8942**	**0.8372**
		w/ $\mathcal{D}_{norm}$	0.8895	0.8140
		w/ $\mathcal{D}_{kde}$	0.8857	0.8140
	Vanilla BERT	Pairwise	0.8477	0.7442
		w/ $\mathcal{D}_{recip}$	0.8624	0.7674
		w/ $\mathcal{D}_{norm}$	0.8581	0.7907
		w/ $\mathcal{D}_{kde}$	**0.8837**	↑ **0.8372**
Baselines	BM25	Default	0.7024	0.5814
		Tuned	0.6653	0.5349
		Optimized	0.7555	0.6744
	SDM	Default	0.6276	0.4884
		Tuned	0.6243	0.4884
		Optimized	0.6667	0.5814
	Top TREC Re-Ranking runs [6]	1.	0.907	-
		2.	0.882	-
		3.	0.870	-
	Optimal Re-Ranker		0.9767	0.9767

The training data for this dataset consists of over a million questions collected from the Bing query log. A human annotator was presented a question and a list of 10 candidate answer passages. The annotator was asked to produce a written answer to these questions based on the passages and to indicate the passages that were most valuable in the production of this answer. For the purposes of passage ranking, these passages are considered relevant to the corresponding question. We note that this does not necessarily mean that all correct passages are annotated as relevant, nor it means that the *best* passage is annotated (better answers could exist beyond the 10 shown to the annotator). To overcome this limitation, the TREC DL track manually judged the top retrieved passages for a subset of the test collection. This evaluation setting, which uses manual relevance judgments, is more suitable for evaluation than prior works that relied on incomplete relevance judgments (e.g., [28]). These incomplete training relevance labels also make this dataset suitable for our curriculum learning approach; answers ranked highly by an unsupervised ranker may be relevant, so down-weighting these samples during training may be beneficial.

We train our models using the official MS-MARCO list of training positive and negative relevance judgments. We use a held-out set of 200 queries for validation. We re-rank the official initial test set ranking,[4] and we use the official TREC DL manual relevance judgments for our evaluation and analysis. Statistics about the training, development, and test sets are given in Table 2.

Since this work focuses on re-ranking, we evaluate using precision-oriented metrics, and leave recall to future work. We use mean reciprocal rank at 10 (MRR@10) as the validation metric, as it is the official task evaluation metric. Although not included as an official task metric, we also evaluate using P(recision)@1, which indicates the performance of the ranker in a realistic setting in which a single answer is given to a question.

We present the ranking performance for TREC DL in Table 3. We observe that under all conditions, our proposed curricula outperform the ranker when trained without a curriculum for both MRR and P@1 metrics. $\mathcal{D}_{recip}$ outperforms the other curricula for ConvKNRM and pointwise Vanilla BERT, while $\mathcal{D}_{kde}$ outperforms the other curricula for pairwise Vanilla BERT.

When the model significantly under-performs well-tuned BM25 and SDM (ConvKNRM), we observe that the curricula can improve the ranking performance to approximately the level of these baselines. When the model is already doing substantially better (Vanilla BERT), our training curricula also yield a considerable boost to ranking effectiveness. The observation that our approach can improve

[4]Another evaluation setting for TREC DL is "full ranking", in which systems perform initial retrieval in addition to re-ranking. Since this work focuses on improving the effectivness of re-raning models rather than initial stage retrieval, we compare with other re-ranking submissions.

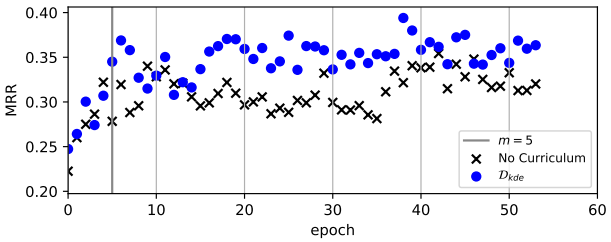

Figure 3: Validation performance comparison between Vanilla BERT model trained with pointwise loss without a curriculum (black x) and with the $\mathcal{D}_{kde}$ curriculum (blue circle) for TREC DL. The tuned m parameter for the $\mathcal{D}_{kde}$ curriculum used here is marked with a vertical line. While the variant without a curriculum quickly reaches optimal performance, the curriculum approach reaches a higher performance faster and offers a stronger foundation on which to continue training after the curriculum terminates.

the ranking effectiveness in both these cases is encouraging, and suggests that the approach is generally beneficial. When compared to the top TREC DL re-ranking results [6], our approach performs favorably. Specifically, the top approach, namely pointwise Vanilla BERT with $\mathcal{D}_{recip}$, ranks second among the submissions. It is only narrowly exceeded by a much more expensive and complicated approach of pretraining a new BERT model from scratch using a different training objective. Our results indicate that this can be avoided by simply doing a better job weighting the training samples.

To gain a better understanding of how the curriculum benefits the training process, we compare the validation performance of the pointwise Vanilla BERT model with the $\mathcal{D}_{kde}$ training curriculum to the same model when trained without a curriculum (Figure 3). This reveals that when not using a curriculum, the validation performance peaks early, suggesting that it is overfitting to difficult examples. The curriculum, however, has even stronger early performance and is in a better position to incorporate difficult samples as training continues. Note that the tuned end of curriculum epoch is $m = 5$ for this example, showing that the curriculum does not need to be in place for long to get these benefits. Also note that the training data were presented in the exact same order in both cases, showing the importance of weighting the loss effectively.

4.2 Complex answer passage ranking

We also evaluate our curriculum learning framework on the TREC Complex Answer Retrieval (CAR) dataset [10]. To compare with prior work, we use version 1.0 of the dataset. This dataset consists of topics in the form of a hierarchy of article headings (e.g., *Green Sea Turtle » Ecology and behavior » Diet*). A standard set of automatically-generated relevance judgments are provided by assuming paragraphs (passages) under a heading are relevant to the query corresponding to the heading. The automatic relevance assessments provide a large amount of training data, but can suffer from variable quality (e.g., some paragraphs are very difficult to match as they provide little context). This makes TREC CAR a good application of training curricula; it contains many positive relevance samples that are difficult to match. A set of manually-graded relevance assessments are also provided by TREC assessors. However, due to the shallow assessment pool used (due to the large number of topics), we opt to only evaluate our approach using the automatic judgments.[5] We use TREC 2017 (`Y1`) training data with `hierarchical` relevance judgments. We also compare our results to the performance reported by [28] and [24], which use BERT and the PACRR neural ranking architecture augmented with entity embeddings for classification, respectively.

Following previous work [28], we train and validate our models using the top 10 results retrieved by BM25 and test on the top 1000 results. We use the official task metric of R-Prec(ision) to validate our model. We also report MAP, another official metric for the task. We use these metrics rather than MRR and P@1 because CAR

Table 4: Ranking performance on the TREC CAR complex answer passage ranking task. Significant improvements in performance when using the training curricula (as compared to no curriculum) for each model are indicated with ↑ (paired t-test $p < 0.05$, no significant reductions observed). For Pointwise loss, $\mathcal{D}_{recip}$ significantly outperforms $\mathcal{D}_{norm}$ in terms of MAP. There are no other significant differences among the training curricula. The top results in each section are indicated in bold.

	TREC CAR			
	Ranker	Training	R-Prec	MAP
	ConvKNRM	Pairwise	0.1081	0.1412
		w/ $\mathcal{D}_{recip}$	↑ 0.1174	↑ 0.1493
		w/ $\mathcal{D}_{norm}$	↑ **0.1258**	↑ **0.1572**
		w/ $\mathcal{D}_{kde}$	↑ 0.1227	↑ 0.1553
	Vanilla BERT	Pointwise	0.2026	0.2490
		w/ $\mathcal{D}_{recip}$	↑ **0.2446**	↑ **0.2864**
		w/ $\mathcal{D}_{norm}$	↑ 0.2370	↑ 0.2764
		w/ $\mathcal{D}_{kde}$	↑ 0.2370	↑ 0.2795
	Vanilla BERT	Pairwise	0.2731	0.3207
		w/ $\mathcal{D}_{recip}$	↑ 0.2914	↑ 0.3298
		w/ $\mathcal{D}_{norm}$	↑ **0.2921**	↑ **0.3307**
		w/ $\mathcal{D}_{kde}$	↑ 0.2844	0.3254
Baselines	BM25	Default Settings	0.1201	0.1563
		Tuned	0.1223	0.1583
		Optimized	0.1231	0.1588
	SDM	Default Settings	0.1154	0.1463
		Tuned	0.1099	0.1420
		Optimized	0.1155	0.1459
	BERT Large [28]		-	0.335
	BERT Base [28]		-	0.310
	PACRR [24]		0.146	0.176
	Optimal Re-Ranker		0.6694	0.6694

[5]The track report suggests that the automatic judgments are a reasonable proxy for manual judgments as there is a strong correlation between the automatic and manual performance among the TREC submissions [10].

queries often need many relevant passages to answer the question, not just one.

We present the performance of our training curricula on TREC CAR in Table 4. We observe that in all cases, the training curricula significantly improve the ranking effectiveness. When training rankers using pairwise loss, the $\mathcal{D}_{norm}$ curriculum is most effective, and when training with pointwise loss, the $\mathcal{D}_{recip}$ curriculum is most effective. In the case of ConvKNRM, without the curriculum, the ranker under-performs the unsupervised BM25 and SDM baselines; with the curricula, it performs on-par with them. For Vanilla BERT, both when trained with pairwise and pointwise losses, the ranker outperforms the unsupervised baselines without the curricula, and improves significantly when using the curricula.

When compared with the supervised baselines, i.e., BERT and PACRR, the Vanilla BERT model trained with pairwise loss and $\mathcal{D}_{norm}$ curriculum ends up performing about as well as the large BERT baseline reported by [28] (0.3307 versus 0.335 in terms of MAP, no statistically significant difference). This is a considerable achievement because the Vanilla BERT model is half the size and about twice as fast to execute. This observation strengthens the case for using curricula when training because it can allow for similar gains as using a much larger model.

The remaining gap between our trained models and the optimal re-ranker on the CAR dataset, however, indicates that there is still room for improvement in this task. In particular, a considerable challenge is ranking passages without much context highly without adding too much noise to the model.

4.3 Non-factoid question answering

We also test our approach on the ANTIQUE non-factoid question answering dataset [13]. Unlike TREC DL and CAR, ANTIQUE has more thoroughly annotated training queries, with an around 11 graded relevance judgments per query in the training and validation collections (crowdsourced) (see Table 2). Furthermore, these include explicit labels for non-relevant answers, which are not present in the other two datasets. This more extensive annotation comes at the expense of scale, however, with far fewer queries to train upon. Nevertheless, ANTIQUE represents another valuable set of conditions under which to evaluate our curricula. We randomly sample from the top 100 BM25 results for additional negative samples during training. We validate and test by re-ranking the top 100 BM25 results, and MRR as the validation metric and P@1 as a secondary metric. We use these two official task metrics (at relevance level of 3 or higher, as specified in [13]) because the answers in ANTIQUE are self-contained, and these metrics emphasize correct answers that are ranked highly and first, respectively.

We report the curricula performance on ANTIQUE in Table 5. Similar to TREC DL, we observe that the $\mathcal{D}_{recip}$ and $\mathcal{D}_{kde}$ curricula are the most effective. For ConvKNRM, the curricula were able to overcome what would otherwise be a model that under-performs w.r.t. the BM25 and SDM unsupervised baselines. For the pointwise and pairwise Vanilla BERT models (which are already very effective), we observe gains beyond. In the case of pairwise-trained Vanilla BERT, the $\mathcal{D}_{recip}$ curriculum significantly boosted ranking performance. Despite our efforts to reproduce the effectiveness of BERT reported in [13], we were unable to do so using the experimental settings described in that work. These results are still below that of an optional re-ranking, suggesting that there is still considerable room for improvement when ranking these non-factoid answers.

Table 5: Ranking performance on the ANTIQUE non-factoid question answering task. Significant improvements in performance when using the training curricula (as compared to no curriculum) are indicated with ↑ (paired t-test $p < 0.05$). There are no statistically-significant differences among the curricula. The top results in each section are indicated in bold.

			ANTIQUE	
	Ranker	Training	MRR	P@1
	ConvKNRM	Pairwise	0.4920	0.3650
		w/ $\mathcal{D}_{recip}$	↑ **0.5617**	↑ **0.4550**
		w/ $\mathcal{D}_{norm}$	↑ 0.5523	↑ 0.4450
		w/ $\mathcal{D}_{kde}$	↑ 0.5563	↑ 0.4500
	Vanilla BERT	Pointwise	0.6694	0.5550
		w/ $\mathcal{D}_{recip}$	0.6858	0.5850
		w/ $\mathcal{D}_{norm}$	0.6888	0.5800
		w/ $\mathcal{D}_{kde}$	**0.6953**	**0.6000**
	Vanilla BERT	Pairwise	0.6999	0.5850
		w/ $\mathcal{D}_{recip}$	↑ **0.7335**	↑ **0.6450**
		w/ $\mathcal{D}_{norm}$	0.7237	0.6250
		w/ $\mathcal{D}_{kde}$	0.7244	0.6250
Baselines	BM25	Default Settings	0.5464	0.4450
		Tuned	0.5802	0.4550
		Optimized	0.6035	0.4950
	SDM	Default Settings	0.5229	0.4050
		Tuned	0.5377	0.4400
		Optimized	0.5491	0.4700
	Best prior published (BERT) [13]		0.7968	0.7092
	Optimal Re-Ranker		0.9400	0.9400

To answer RQ1 (whether the training curricula are effective), we observed that for three answer ranking datasets (TREC DL, TREC CAR, and ANTIQUE) these curricula can improve the ranking effectiveness across multiple neural rankers and loss functions. We observe that when a ranker initially underperforms standard baselines (e.g., ConvKNRM), the performance is effectively boosted to the level of those baselines. When the ranker already exceeds these baselines (e.g., Vanilla BERT), we also observe a boost to ranking effectiveness, often comparable to or approaching the state-of-the-art while being considerably faster (e.g., using BERT Base instead of BERT Large) or less complicated (e.g., not requiring an expensive pre-training step). The observation that the curricula are effective in these various conditions suggests that these curricula are generally effective. To answer RQ2 (under what conditions each curriculum is effective), we observe that $\mathcal{D}_{recip}$ and $\mathcal{D}_{kde}$ are generally more effective for natural-language questions (TREC DL and ANTIQUE), while $\mathcal{D}_{norm}$ is more effective for keyword/structured questions

(TREC CAR). One possible alternative explanation may be that the latter is better with weak relevance labels, as TREC CAR's relevance labels are obtained through a heuristic, rather than human annotators. It does not appear as if the amount of training data has an effect, as TREC DL and ANTIQUE exhibit similar characteristics, while having drastically different amounts of training data.

4.4 End of curriculum evaluation

We already observed in Figure 3 that when using a training curriculum, ranking performance not only peaks higher sooner, but also leaves the model in a better starting point for when all samples are weighted equally. However, an important question remains: Is it important to train with equal weight for all samples or can the difficulty weights be used exclusively? To this end, we perform a test that forgoes the curriculum convergence parameter m, directly using $\mathcal{D}(\cdot)$ as the training sample weight, regardless of training iteration (i.e., $m = \infty$, or equivalently $W = \mathcal{D}$ instead of Eq. 3).

We report the performance for this experiment on each dataset for each top-performing curriculum in Table 6 ($m = \infty$ setting). We observe that for all models on the TREC DL and ANTIQUE datasets, this approach leads to a drop in ranking effectiveness, suggesting that it is important to eventually perform equal sample weighting. Intuitively, this is important because if easy samples are always weighted higher than difficult samples, the model will be hindered in learning the more complicated function to rank difficult samples. Curiously, for TREC CAR, this setting sometimes leads to improved ranking effectiveness (though not a statistically significant improvement). One possible explanation is that in situations where weak labels are used (rather than human-judged labels from top retrieved results), it may be better to always apply the weighting, as some inferred positive labels may be too distant from what the model will typically encounter at inference time.

To answer RQ3 (whether shifting to difficult samples is important), we find that it is indeed beneficial to use our proposed weighting technique given in Eq. 3, rather than always applying the difficulty weighting when using manually-assessed relevance labels.

4.5 Anti-curriculum: Hardest samples first

To test whether our intuitions that "difficult" samples are harmful during early phases of training, we conduct a study using an anti-curriculum, i.e., we train our models by weighting the more difficult samples higher than the easier samples. This was applied by swapping out the difficulty function $\mathcal{D}$ with $\widehat{\mathcal{D}}(\cdot) = 1 - \mathcal{D}(\cdot)$. This has the effect of assigning high weights to samples that previously had low weights and vice versa. All usage of the difficulty function remains unchanged (e.g., the integration of the difficulty function into the weight function).

Table 6 ($\widehat{\mathcal{D}}$ setting) presents a ranking performance comparison when using the anti-curriculum. We observe that the anti-curriculum always reduces ranking effectiveness, sometimes significantly. In some cases, this can be rather severe; on TREC DL for Vanilla BERT (pairwise), the MRR is reduced by 0.0523 and P@1 is reduced by 0.1163, resulting in a model that underperforms one without any weighting at all. To answer RQ4, these results suggest that there is benefit to weighting the easiest samples higher first, rather than the more difficult samples.

Table 6: Ranker performance when the curriculum always uses difficulty scores, and never assigns equal weight to all samples (i.e., $m = \infty$), and when employing the anti-curriculum ($\widehat{\mathcal{D}}$). Significant reductions in performance are indicated with ↓ (paired t-test, $p < 0.05$).

TREC DL

Ranker	Curriculum	m	MRR@10	P@1
ConvKNRM	$\mathcal{D}^{pair}_{recip}$	20	**0.6834**	**0.5581**
	$\mathcal{D}^{pair}_{recip}$	∞	0.6744	0.5581
	$\widehat{\mathcal{D}}^{pair}_{recip}$	20	↓ 0.5414	↓ 0.3721
Vanilla BERT	$\mathcal{D}^{point}_{recip}$	10	**0.8942**	**0.8372**
	$\mathcal{D}^{point}_{recip}$	∞	0.8205	0.7209
	$\widehat{\mathcal{D}}^{point}_{recip}$	10	0.8527	0.7442
Vanilla BERT	$\mathcal{D}^{pair}_{kde}$	20	**0.8837**	**0.8372**
	$\mathcal{D}^{pair}_{kde}$	∞	↓ 0.7752	↓ 0.6279
	$\widehat{\mathcal{D}}^{pair}_{kde}$	20	0.8314	0.7209

TREC CAR

Ranker	Curriculum	m	R-Prec	MAP
ConvKNRM	$\mathcal{D}^{pair}_{norm}$	50	**0.1258**	0.1572
	$\mathcal{D}^{pair}_{norm}$	∞	0.1250	**0.1579**
	$\widehat{\mathcal{D}}^{pair}_{norm}$	50	↓ 0.1030	↓ 0.1324
Vanilla BERT	$\mathcal{D}^{point}_{recip}$	20	0.2446	0.2864
	$\mathcal{D}^{point}_{recip}$	∞	**0.2475**	**0.2894**
	$\widehat{\mathcal{D}}^{point}_{recip}$	20	↓ 0.2258	↓ 0.2709
Vanilla BERT	$\mathcal{D}^{pair}_{norm}$	10	**0.2921**	**0.3307**
	$\mathcal{D}^{pair}_{norm}$	∞	↓ 0.2669	↓ 0.3103
	$\widehat{\mathcal{D}}^{pair}_{norm}$	10	0.2837	0.3276

ANTIQUE

Ranker	Curriculum	m	MRR	P@1
ConvKNRM	$\mathcal{D}^{pair}_{recip}$	100	**0.5617**	**0.4550**
	$\mathcal{D}^{pair}_{recip}$	∞	0.5368	0.4100
	$\widehat{\mathcal{D}}^{pair}_{recip}$	100	0.5366	0.4200
Vanilla BERT	$\mathcal{D}^{point}_{kde}$	10	**0.6953**	**0.6000**
	$\mathcal{D}^{point}_{kde}$	∞	↓ 0.6139	↓ 0.4750
	$\widehat{\mathcal{D}}^{point}_{kde}$	10	0.6677	0.5500
Vanilla BERT	$\mathcal{D}^{pair}_{recip}$	5	**0.7335**	**0.6450**
	$\mathcal{D}^{pair}_{recip}$	∞	0.7158	0.6150
	$\widehat{\mathcal{D}}^{pair}_{recip}$	5	0.7193	0.6200

5 CONCLUSIONS AND FUTURE WORK

We proposed three weighting heuristics to train neural rankers using curriculum learning that boost performance when ranking answers on three datasets. Our proposed heuristics boost ranking performance through training samples weighting, without changing the sequence in which the data are presented to the model for training. Generally, the reciprocal rank (RECIP) and kernel density estimation (KDE) curricula were the most effective, although when working with inferred relevance labels with TREC CAR, the normalized score (NORM) curriculum was more effective. Although these gains were not always enough to achieve state-of-the-art performance, they were often able to approach the level of performance of larger or more complicated approaches (such as using BERT (large) or re-training BERT with a different pre-training objective). We experimentally showed that the convergence of the curriculum to equal weighting is important when manually-labeled test data are used, otherwise resulting in inferior effectiveness. Finally, we found that focusing on the easiest samples first (rather than the hardest samples) was also an important characteristic of this approach.

Future work could explore alternative difficultly degradation functions or explore how well the method applies to other approaches, such as performing additional domain fine-tuning. It could also combine the weighting strategies with more intelligent sampling approaches for relevant and non-relevant training pairs. We note that our proposed difficulty heuristics may be an effective starting point for sampling strategies. Even with more effective sampling, our weighting approach may be beneficial for ensuring that 'easy' samples are ranked effectively. Another possible direction for future work could explore the use of self-paced learning techniques [19, 20], allowing the model to learn which training samples characteristics make a samples easy or difficult.

ACKNOWLEDGMENTS

Work partially supported by the ARCS Foundation. Work partially supported by the Italian Ministry of Education and Research (MIUR) in the framework of the CrossLab project (Departments of Excellence). Work partially supported by the BIGDATAGRAPES project funded by the EU Horizon 2020 research and innovation programme under grant agreement No. 780751, and by the OK-INSAID project funded by the Italian Ministry of Education and Research (MIUR) under grant agreement No. ARS01_00917.

REFERENCES

[1] Y. Bengio, J. Louradour, R. Collobert, and J. Weston. 2009. Curriculum Learning. In *ICML*. 41–48.
[2] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching Word Vectors with Subword Information. *TACL* 5 (2017), 135–146.
[3] X. Chen and A. Gupta. 2015. Weakly Supervised Learning of Convolutional Networks. In *International Conference on Computer Vision*. 1431–1439.
[4] T. F. Coleman and Z. Wu. 1996. Parallel Continuation-based Global Optimization for Molecular Conformation and Protein Folding. *Journal of Global Optimization* 8, 1 (January 1996), 49–65.
[5] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. 2011. Natural Language Processing (Almost) from Scratch. *The Journal of Machine Learning Research* 12 (August 2011), 2493–2537.
[6] Nick Craswell, Bhaskar Mitra, and Daniel Campos. 2019. Overview of the TREC 2019 Deep Learning Track. In *TREC*.
[7] Zhuyun Dai, Chenyan Xiong, Jamie Callan, and Zhiyuan Liu. 2018. Convolutional Neural Networks for Soft-Matching N-Grams in Ad-hoc Search. In *WSDM*. 126–134.
[8] Mostafa Dehghani, Arash Mehrjou, Stephan Gouws, Jaap Kamps, and Bernhard Schölkopf. 2017. Fidelity-Weighted Learning. In *ICLR*.
[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL*.
[10] Laura Dietz, Ben Gamari, Jeff Dalton, and Nick Craswell. 2017. TREC Complex Answer Retrieval Overview. In *TREC*.
[11] N. Ferro, C. Lucchese, M. Maistro, and R. Perego. 2018. Continuation Methods and Curriculum Learning for Learning to Rank. In *CIKM*. 1523–1526.
[12] Jiafeng Guo, Yixing Fan, Qingyao Ai, and W. Bruce Croft. 2016. A Deep Relevance Matching Model for Ad-hoc Retrieval. In *CIKM*. 55–64. http://arxiv.org/abs/1711.08611 arXiv: 1711.08611.
[13] Helia Hashemi, Mohammad Aliannejadi, Hamed Zamani, and W. Bruce Croft. 2019. ANTIQUE: A Non-Factoid Question Answering Benchmark. *ArXiv* abs/1905.08957 (2019).
[14] Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional Neural Network Architectures for Matching Natural Language Sentences. In *NIPS*.
[15] B. Hu, Z. Lu, H. Li, and Q. Chen. 2014. Convolutional Neural Network Architectures for Matching Natural Language Sentences. In *NIPS*. 2042–2050.
[16] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry P. Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *CIKM*.
[17] Kai Hui, Andrew Yates, Klaus Berberich, and Gerard de Melo. 2018. Co-PACRR: A Context-Aware Neural IR Model for Ad-hoc Retrieval. In *WSDM*. ACM, 279–287.
[18] Shiyu Ji, Jinjin Shao, and Tao Yang. 2019. Efficient Interaction-based Neural Ranking with Locality Sensitive Hashing. In *WWW*.
[19] Lu Jiang, Deyu Meng, Shoou-I Yu, Zhen-Zhong Lan, Shiguang Shan, and Alexander G. Hauptmann. 2014. Self-Paced Learning with Diversity. In *NIPS*.
[20] Lu Jiang, Deyu Meng, Qian Zhao, Shiguang Shan, and Alexander G. Hauptmann. 2015. Self-Paced Curriculum Learning. In *AAAI*.
[21] Jimmy Lin. 2018. The Neural Hype and Comparisons Against Weak Baselines. *SIGIR Forum* 52 (2018), 40–51.
[22] Jimmy Lin and Peilin Yang. 2019. The Impact of Score Ties on Repeatability in Document Ranking. In *SIGIR*. http://arxiv.org/abs/1807.05798 arXiv: 1807.05798.
[23] Sean MacAvaney, Andrew Yates, Arman Cohan, and Nazli Goharian. 2019. CEDR: Contextualized Embeddings for Document Ranking. In *SIGIR 2019*.
[24] Sean MacAvaney, Andrew Yates, Arman Cohan, Luca Soldaini, Kai Hui, Nazli Goharian, and Ophir Frieder. 2018. Overcoming Low-Utility Facets for Complex Answer Retrieval. *Information Retrieval Journal* (2018).
[25] Donald Metzler and W. Bruce Croft. 2005. A Markov random field model for term dependencies. In *SIGIR*.
[26] Bhaskar Mitra and Nick Craswell. 2017. Neural Models for Information Retrieval. *ArXiv* abs/1705.01509 (2017).
[27] Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A Human Generated MAchine Reading COmprehension Dataset. In *CoCo@NIPS*.
[28] Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage Re-ranking with BERT. *arXiv:1901.04085* (2019).
[29] Rodrigo Nogueira, Wei Yang, Jimmy Lin, and Kyunghyun Cho. 2019. Document Expansion by Query Prediction. *ArXiv* abs/1904.08375 (2019).
[30] Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, and Xueqi Cheng. 2016. A Study of MatchPyramid Models on Ad-hoc Retrieval. In *NeuIR @ SIGIR*.
[31] Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Jingfang Xu, and Xueqi Cheng. 2017. DeepRank: A New Deep Architecture for Relevance Ranking in Information Retrieval. In *CIKM*. 257–266.
[32] Gustavo Penha and Claudia Hauff. 2020. Curriculum Learning Strategies for IR: An Empirical Study on Conversation Response Ranking. In *ECIR*.
[33] M. Qu, J. Tang, and J. Han. 2018. Curriculum Learning for Heterogeneous Star Network Embedding via Deep Reinforcement Learning. In *WSDM*. 468–476.
[34] Mrinmaya Sachan and Eric P. Xing. 2016. Easy Questions First? A Case Study on Curriculum Learning for Question Answering. In *ACL*.
[35] David W Scott. 2015. *Multivariate density estimation: theory, practice, and visualization*. John Wiley & Sons.
[36] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014. Learning semantic representations using convolutional neural networks for web search. In *WWW*.
[37] Chenyan Xiong, Zhuyun Dai, Jamie Callan, Zhiyuan Liu, and Russell Power. 2017. End-to-End Neural Ad-hoc Ranking with Kernel Pooling. In *SIGIR*. 55–64. http://arxiv.org/abs/1706.06613 arXiv: 1706.06613.
[38] Peilin Yang, Hui Fang, and Jimmy Lin. 2018. Anserini: Reproducible Ranking Baselines Using Lucene. *J. Data and Information Quality* 10 (2018), 16:1–16:20.
[39] Wei Yang, Kuang Lu, Peilin Yang, and Jimmy Lin. 2019. Critically Examining the "Neural Hype": Weak Baselines and the Additivity of Effectiveness Gains from Neural Ranking Models. In *SIGIR*.