Curriculum Learning Strategies for IR
An Empirical Study on Conversation Response Ranking

Gustavo Penha and Claudia Hauff

TU Delft
{g.penha-1,c.hauff}@tudelft.nl

Abstract. Neural ranking models are traditionally trained on a series of random batches, sampled uniformly from the entire training set. Curriculum learning has recently been shown to improve neural models' effectiveness by sampling batches non-uniformly, going from easy to difficult instances during training. In the context of neural Information Retrieval (IR) curriculum learning has not been explored yet, and so it remains unclear (1) how to *measure the difficulty* of training instances and (2) *how to transition* from easy to difficult instances during training. To address both challenges and determine whether curriculum learning is beneficial for neural ranking models, we need large-scale datasets and a retrieval task that allows us to conduct a wide range of experiments. For this purpose, we resort to the task of *conversation response ranking*: ranking responses given the conversation history. In order to deal with challenge (1), we explore *scoring functions* to measure the difficulty of conversations based on different input spaces. To address challenge (2) we evaluate different *pacing functions*, which determine the velocity in which we go from easy to difficult instances. We find that, overall, by just intelligently sorting the training data (i.e., by performing curriculum learning) we can improve the retrieval effectiveness by up to 2%[1].

Keywords: curriculum learning · conversation response ranking

1 Introduction

Curriculum Learning (CL) is motivated by the way humans teach complex concepts: teachers impose a certain order of the material during students' education. Following this guidance, students can exploit previously learned concepts to more easily learn new ones. This idea was initially applied to machine learning over two decades ago [8] as an attempt to use a similar strategy in the training of a recurrent network by *starting small* and gradually learning more difficult examples. More recently, Bengio et al. [1] provided additional evidence that curriculum strategies can benefit neural network training with experimental results on different tasks such as shape recognition and language modelling. Since then, empirical successes were observed for several computer vision [14,49] and natural language processing (NLP) tasks [36,42,60].

[1] The source code is available at `https://github.com/Guzpenha/transformers_cl`.

In supervised machine learning, a function is learnt by the learning algorithm (the *student*) based on inputs and labels provided by the *teacher*. The teacher typically samples randomly from the entire training set. In contrast, CL imposes a structure on the training set based on a notion of difficulty of instances, presenting to the student easy instances before difficult ones. When defining a CL strategy we face two challenges that are specific to the domain and task at hand [14]: (1) arranging the training instances by a sensible measure of *difficulty*, and, (2) determining the *pace* in which to present instances—going over easy instances too fast or too slow might lead to ineffective learning.

We conduct here an empirical investigation into those two challenges in the context of IR. Estimating relevance—a notion based on human cognitive processes—is a complex and difficult task at the core of IR, and it is still unknown *to what extent CL strategies are beneficial for neural ranking models*. This is the question we aim to answer in our work.

Given a set of queries—for instance user utterances, search queries or questions in natural language—and a set of documents—for instance responses, web documents or passages—neural ranking models learn to distinguish relevant from non-relevant query-document pairs by training on a large number of labeled training pairs. Neural models have for some time struggled to display significant and additive gains in IR [53]. In a short time though, BERT [7] (released in late 2018) and its derivatives (e.g. XLNet [56], RoBERTa [25]) have proven to be remarkably effective for a range of NLP tasks. The recent breakthroughs of these large and heavily pre-trained language models have also benefited IR [55,54,57].

In our work we focus on the challenging IR task of conversation response ranking [50], where the query is the dialogue history and the documents are the candidate responses of the agent. The set of responses are not generated on the go, they must be retrieved from a comprehensive dialogue corpus. A number of deep neural ranking models have recently been proposed for this task [43,52,61,50,62], which is more complex than retrieval for single-turn interactions, as the ranking model has to determine where the important information is in the previous user utterances (dialogue history) and how it is relevant to the current information need of the user. Due to the complexity of the relevance estimation problem displayed in this task, we argue it to be a good test case for curriculum learning in IR.

In order to tackle the first challenge of CL (determine what makes an instance difficult) we study different *scoring functions* that determine the difficulty of query-document pairs based on four different input spaces: conversation history $\{\mathcal{U}\}$, candidate responses $\{\mathcal{R}\}$, both $\{\mathcal{U},\mathcal{R}\}$, and $\{\mathcal{U}, \mathcal{R}, \mathcal{Y}\}$, where $\mathcal{Y}$ are relevance labels for the responses. To address the second challenge (determine the pace to move from easy to difficult instances) we explore different *pacing functions* that serve easy instances to the learner for more or less time during the training procedure. We empirically explore how the curriculum strategies perform for two different response ranking datasets when compared against vanilla (no curriculum) fine-tuning of BERT for the task. Our main findings are that (i) CL improves retrieval effectiveness when we use a difficulty criteria based on

a supervised model that uses all the available information $\{\mathcal{U}, \mathcal{R}, \mathcal{Y}\}$, (ii) it is best to give the model more time to assimilate harder instances during training by introducing difficult instances in earlier iterations, and, (iii) the CL gains over the no curriculum baseline are spread over different conversation domains, lengths of conversations and measures of conversation difficulty.

2 Related Work

Neural Ranking Models Over the past few years, the IR community has seen a great uptake of the many flavours of deep learning for all kinds of IR tasks such as ad-hoc retrieval, question answering and conversation response ranking. Unlike traditional learning to rank (LTR) [24] approaches in which we manually define features for queries, documents and their interaction, neural ranking models learn features directly from the raw textual data. Neural ranking approaches can be roughly categorized into representation-focused [17,38,47] and interaction-focused [13,48]. The former learns query and document representations separately and then computes the similarity between the representations. In the latter approach, first a query-document interaction matrix is built, which is then fed to neural net layers. Estimating relevance directly based on interactions, i.e. interaction-focused models, has shown to outperform representation-based approaches on several tasks [27,16].

Transfer learning via large pre-trained Transformers [46]—the prominent case being BERT [7]—has lead to remarkable empirical successes on a range of NLP problems. The BERT approach to learn textual representations has also significantly improved the performance of neural models for several IR tasks [55,54,37,33,57], that for a long time struggled to outperform classic IR models [53]. In this work we use the no-CL BERT as a strong baseline for the conversation response ranking task.

Curriculum Learning Following a curriculum that dictates the ordering and content of the education material is prevalent in the context of human learning. With such guidance, students can exploit previously learned concepts to ease the learning of new and more complex ones. Inspired by cognitive science research [35], researchers posed the question of whether a machine learning algorithm could benefit, in terms of learning speed and effectiveness, from a similar curriculum strategy [8,1]. Since then, positive evidence for the benefits of curriculum training, i.e. training the model using easy instances first and increasing the difficulty during the training procedure, has been empirically demonstrated in different machine learning problems, e.g. image classification [14,11], machine translation [30,21,60] and answer generation [23].

Processing training instances in a meaningful order is not unique to CL. Another related branch of research focuses on *dynamic* sampling strategies [22,4,39,2], which unlike CL that requires a definition of what is easy and difficult before training starts, estimates the importance of instances during the training procedure. Self-paced learning [22] simultaneously selects easy instances to focus

on and updates the model parameters by solving a biconvex optimization problem. A seemingly contradictory set of approaches give more focus to difficult or more uncertain instances. In active learning [6,44,4], the most uncertain instances with respect to the current classifier are employed for training. Similarly, hard example mining [39] focuses on difficult instances, measured by the model loss or magnitude of gradients for instance. Boosting [2,59] techniques give more weight to difficult instances as training progresses. In this work we focus on CL, which has been more successful in neural models, and leave the study of dynamic sampling strategies in neural IR as future work.

The most critical part of using a CL strategy is defining the difficulty metric to sort instances by. The estimation of instance difficulty is often based on our prior knowledge on what makes each instance difficult for a certain task and thus is domain dependent (cf. Table 1 for curriculum examples). CL strategies have not been studied yet in neural ranking models. To our knowledge, CL has only recently been employed in IR within the LTR framework, using LambdaMart [3], for ad-hoc retrieval by Ferro et al. [9]. However, no effectiveness improvements over randomly sampling training data were observed. The representation of the query, document and their interactions in the traditional LTR framework is dictated by the manually engineered input features. We argue that neural ranking models, which learn how to represent the input, are better suited for applying CL in order to learn increasingly more complex concepts.

Table 1: Difficulty measures used in the curriculum learning literature.

Difficulty criteria	**Tasks**
sentence length	machine translation [30], language generation [42], reading comprehension [58]
word rarity	machine translation [30,60], language modeling [1]
external model confidence	machine translation [60], image classification [49,14], ad-hoc retrieval [9]
supervision signal intensity	facial expression recognition [12], ad-hoc retrieval [9]
noise estimate	speaker identification [34], image classification [5]
human annotation	image classification [45] (through weak supervision)

3 Curriculum Learning

Before introducing our experimental framework (i.e., the scoring functions and the pacing functions we investigate), let us first formally introduce the specific IR task we explore—a choice dictated by the complex nature of the task (compared to e.g. ad-hoc retrieval) as well as the availability of large-scale training resources such as `MSDialog` [32] and UDC [26].

Conversation Response Ranking Given a historical dialogue corpus and a conversation, (i.e., the user's current utterance and the conversation history)

the task of conversation response ranking [50,52,43] is defined as the ranking of the most relevant response available in the corpus. This setup relies on the fact that a large corpus of historical conversation data exists and adequate replies (that are coherent, well-formulated and informative) to user utterances can be found in it [51]. Formally, let $\mathcal{D} = \{(\mathcal{U}_i, \mathcal{R}_i, \mathcal{Y}_i)\}_{i=1}^{N}$ be an information-seeking conversations data set consisting of N triplets: dialogue context, response candidates and response labels. The dialogue context $\mathcal{U}_i$ is composed of the previous utterances $\{u^1, u^2, ..., u^\tau\}$ at the turn τ of the dialogue. The candidate responses $\mathcal{R}_i = \{r^1, r^2, ..., r^k\}$ are either the true response ($u^{\tau+1}$) or negative sampled candidates[2]. The relevance labels $\mathcal{Y}_i = \{y^1, y^2, ..., y^k\}$ indicate the responses' binary relevance scores, 1 if $r = u^{\tau+1}$ and 0 otherwise. The task is then to learn a ranking function $f(.)$ that is able to generate a ranked list for the set of candidate responses $\mathcal{R}_i$ based on their predicted relevance scores $f(\mathcal{U}_i, r)$.

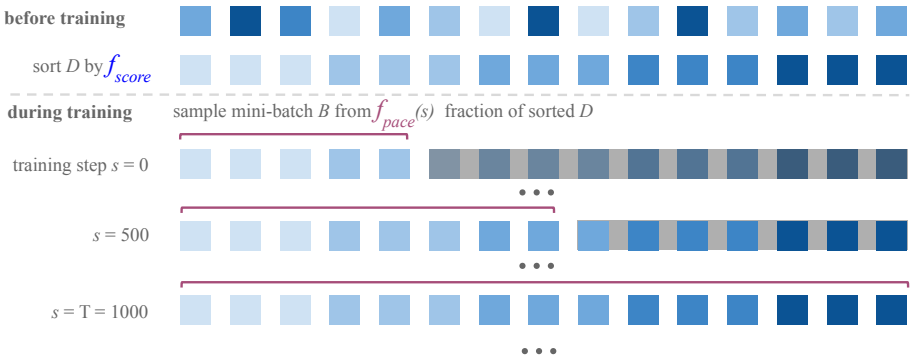

Fig. 1: Our curriculum learning framework is defined by two functions. The scoring function $f_{score}(instance)$ defines the instances' difficulty (darker/lighter blue indicate higher/lower difficulty). The pacing function $f_{pace}(s)$ indicates the percentage of the dataset available for sampling according to the training step s.

Curriculum Framework When training neural networks, the common training procedure is to divide the dataset $\mathcal{D}$ into $\mathcal{D}_{train}, \mathcal{D}_{dev}, \mathcal{D}_{test}$ and randomly (i.e., uniformly—every sample has the same likelihood of being sampled) sample mini-batches $\mathcal{B} = \{(\mathcal{U}_i, \mathcal{R}_i, \mathcal{Y}_i)\}_{i=1}^{k}$ of k instances from $\mathcal{D}_{train}$ where $k \ll N$, and perform an optimization procedure sequentially in $\{\mathcal{B}_1, ..., \mathcal{B}_M\}$. The CL framework employed here is inspired by previous works [49,30]. It is defined by two functions: the *scoring function* which determines the difficulty of instances and the *pacing function* which controls the pace with which to transition from easy to hard instances during training. More specifically, the scoring function $f_{score}(\mathcal{U}_i, \mathcal{R}_i, \mathcal{Y}_i)$, is used to sort the training dataset. The pacing function $f_{pace}(s)$ determines the percentage of the sorted dataset available for sampling

[2] In a production setup the ranker would either retrieve responses from the entire corpus or re-rank the responses retrieved by a recall-oriented retrieval method.

according to the current training step s (one forward pass plus one backward pass of a batch is considered to be one step). The neural ranking model samples uniformly from the initial $f_{pace}(s) * |D_{train}|$ instances sorted by f_{score}, while the rest of the dataset is not available for sampling. During training $f_{pace}(s)$ goes from δ (percentage of initial training data) to 1 when $s = T$. Both δ and T are hyperparameters. We provide an illustration of the training process in Figure 1.

Table 2: Overview of our curriculum learning scoring functions.

Input Space	**Name**	**Definition**	**Difficulty notion**
baseline	*random*	$f_{score} = Uniform(0, 1)$	
$(\mathcal{U})$	$\#_{turns}$	$f_{score}(\mathcal{U}) = \lvert\mathcal{U}\rvert$	information spread
	$\overline{\#_{\mathcal{U}words}}$	$f_{score}(\mathcal{U}) = \frac{\sum_{i=0}^{\lvert\mathcal{U}\rvert} word\_count(u_i)}{\lvert\mathcal{U}\rvert}$	
$(\mathcal{R})$	$\overline{\#_{\mathcal{R}words}}$	$f_{score}(\mathcal{R}) = \frac{\sum_{i=0}^{\lvert\mathcal{R}\rvert} word\_count(r_i)}{\lvert\mathcal{R}\rvert}$	distraction in responses
$(\mathcal{U}, \mathcal{R})$	σ_{SM}	$f_{score}(\mathcal{U}, \mathcal{R}) = \sqrt{\frac{\sum_{i=0}^{\lvert\mathcal{R}\rvert}(SM(\mathcal{U}, r_i) - \overline{SM(\mathcal{U}, \mathcal{R})})^2}{\lvert\mathcal{R}\rvert - 1}}$	responses heterogeneity
	σ_{BM25}	$f_{score}(\mathcal{U}, \mathcal{R}) = \sqrt{\frac{\sum_{i=0}^{\lvert\mathcal{R}\rvert}(BM25(\mathcal{U}, r_i) - \overline{BM25(\mathcal{U}, \mathcal{R})})^2}{\lvert\mathcal{R}\rvert - 1}}$	
$(\mathcal{U}, \mathcal{R}, \mathcal{Y})$	$BERT_{pred}$	$f_{score}(\mathcal{U}, \mathcal{R}, \mathcal{Y}) = -(BERT\_pred(\mathcal{U}, r_i^+) - BERT\_pred(\mathcal{U}, r_i^-))$	model confidence
	$\overline{BERT_{loss}}$	$f_{score}(\mathcal{U}, \mathcal{R}, \mathcal{Y}) = \frac{\sum_{i=0}^{\lvert\mathcal{R}\rvert} BERT\_loss(\mathcal{U}, r_i)}{\lvert\mathcal{R}\rvert}$	

Scoring Functions In order to measure the difficulty of a training triplet composed of $(\mathcal{U}_i, \mathcal{R}_i, \mathcal{Y}_i)$, we define pacing functions that use different parts of the input space: functions that leverage (i) the text in the dialogue history $\{\mathcal{U}\}$ (ii) the text in the response candidates $\{\mathcal{R}\}$ (iii) interactions between them, i.e., $\{\mathcal{U}, \mathcal{R}\}$, and, (iv) all available information including the labels for the training set, i.e., $\{\mathcal{U}, \mathcal{R}, \mathcal{Y}\}$. The seven[3] scoring functions we propose are defined in Table 2; we now provide intuitions of why we believe each function to capture some notion of instance difficulty.

- $\#_{turns}(\mathcal{U})$ and $\overline{\#_{\mathcal{U}words}}(\mathcal{U})$: The important information in the context can be spread over different utterances and words. Bigger dialogue contexts means there are more places where the important part of the user information need can be spread over. $\overline{\#_{\mathcal{R}words}}(\mathcal{R})$: Longer responses can distract the model as to which set of words or sentences are more important for matching. Previous

[3] The function *random* is the baseline—instances are sampled uniformly (no CL).

work shows that it is possible to fool machine reading models by creating longer documents with additional distracting sentences [18].

- $\sigma_{SM}(\mathcal{U}, \mathcal{R})$ and $\sigma_{BM25}(\mathcal{U}, \mathcal{R})$: Inspired by query performance prediction literature [40], we use the variance of retrieval scores to estimate the amount of heterogeneity of information, i.e. diversity, in the response candidate. Homogeneous ranked lists are considered to be easy. We deploy a semantic matching model (SM) and BM25 to capture both semantic correspondences and keyword matching [19]. SM is the average cosine similarity between the first k words from $\mathcal{U}$ (concatenated utterances) with the first k words from r using pre-trained word embeddings.

- $BERT_{pred}(\mathcal{U}, \mathcal{R}, \mathcal{Y})$ and $\overline{BERT_{loss}}(\mathcal{U}, \mathcal{R}, \mathcal{Y})$: Inspired by CL literature [14], we use external model prediction confidence scores as a measure of difficulty[4]. We fine-tune BERT [7] on $\mathcal{D}_{train}$ for the conversation response ranking task. For $BERT_{pred}$ easy dialogue contexts are the ones that the BERT confidence score for the positive response r^+ candidate is higher than the confidence for the negative response candidate r^-. The higher the difference the easier the instance is. For $\overline{BERT_{loss}}$we consider the loss of the model to be an indicator of the difficulty of an instance.

Pacing function	Definition
baseline_training	$f_{pace}(s) = 1$
step	$f_{pace}(s) = \begin{cases} \delta, & \text{if } s \leq T * 0.33 \\ 0.66, & \text{if } s > T * 0.33, s \leq T * 0.66 \\ 1, & \text{if } s > T * 0.66 \end{cases}$
root	$f_{pace}(s, n) = min\left(1, \left(s\frac{1-\delta^n}{T} + \delta^n\right)^{\frac{1}{n}}\right)$
linear	$f_{pace}(s, n) = root(s, 1)$
root_n	$f_{pace}(s, n) = root(s, n)$
geom_progression	$f_{pace}(s) = min\left(1, 2^{\left(s\frac{log_2 1 - log_2 \delta}{T} + log_2 \delta\right)}\right)$

Table 3: Overview of our curriculum learning pacing functions. δ and T are hyperparameters.

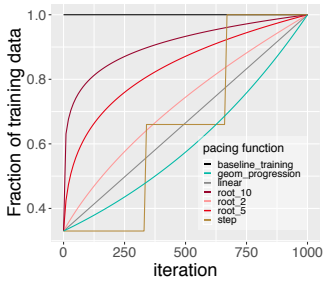

Fig. 2: Example with $\delta = 0.33$ and $T = 1000$.

Pacing functions Assuming that we know the difficulty of each instance in our training set, we still need to define how are we going to transition from easy to hard instances. We use the concept of pacing functions $f_{pace}(s)$; they should each have the following properties [30,49]: (i) start at an initial value of training instances $f_{pace}(0) = \delta$ with $\delta > 0$, so that the model has a number of instances to train in the first iteration, (ii) be non-decreasing, so that harder instances

[4] We note, that using BM25 average precision as a scoring function failed to outperform the baseline.

are added to the training set, and, (iii) eventually all instances are available for sampling when it reaches T iterations, $f_{pace}(T) = 1$.

As intuitively visible in the example in Figure 2, we opted for pacing functions that introduce more difficult instances at different paces—while *root_10* introduces difficult instances very early (after 125 iterations, 80% of all training data is available), *geom_progression* introduces them very late (80% is available after $\sim$ 800 iterations). We consider four different types of pacing functions, formally defined in Table 3. The *step* function [1,14,41] divides the data into S fixed sized groups, and after $\frac{T}{S}$ iterations a new group of instances is added, where S is a hyperparameter. A more gradual transition was proposed by Platanios et. al [30], by adding a percentage of the training dataset linearly with respect to the total of CL iterations T, and thus the slope of the function is $\frac{1-\delta}{T}$ (*linear* function). They also proposed *root_n* functions motivated by the fact that difficult instances will be sampled less as the training data grows in size during training. By making the slope inversely proportional to the current training data size, the model has more time to assimilate difficult instances. Finally, we propose the use of a geometric progression that instead of quickly adding difficult examples, it gives easier instances more training time.

4 Experimental Setup

Datasets We consider two large-scale information-seeking conversation datasets (cf. Table 4) that allow the training of neural ranking models for conversation response ranking. `MSDialog`[5] [32] contain 246K context-response pairs, built from 35.5K information seeking conversations from the Microsoft Answer community, a question-answer forum for several Microsoft products. `MANtIS`[6] [29] was created by us and contains 1.3 million context-response pairs built from conversations of 14 different sites of Stack Exchange. Each `MANtIS` conversation fulfills the following conditions: (i) it takes place between exactly two users (the information *seeker* who starts the conversation and the information *provider*); (ii) it consists of at least 2 utterances per user; (iii) one of the provider's utterances contains a hyperlink, providing grounding; (iv) if the final utterance belongs to the seeker, it contains positive feedback. We created `MANtIS` to consider *diverse* conversations from different domains besides technical ones. We include `MSDialog` [32,52,31] here as a widely used benchmark.

Implementation Details As strong neural ranking model for our experiments, we employ BERT [7] for the conversational response ranking task. We follow recent research in IR that employed fine-tuned BERT for retrieval tasks [28,55] and obtain strong baseline (i.e., no CL) results for our task. The best model by Yang et. al [52], which relies on external knowledge sources for `MSDialog`, achieves a MAP of 0.68 whereas our BERT baselines reaches a MAP of 0.71 (cf.

[5] `MSDialog` is available at `https://ciir.cs.umass.edu/downloads/msdialog/`
[6] `MANtIS` is available at `https://guzpenha.github.io/MANtIS/`

Table 4: Dataset used. $\mathcal{U}$ is the dialogue context, r a response and u an utterance.

	MSDialog			MANtIS		
Number of domains	75			14		
	Train	Valid	Test	Train	Valid	Test
Number of $(\mathcal{U}, r)$ pairs	173k	37k	35k	904k	199k	197k
Number of candidates per $\mathcal{U}$	10	10	10	11	11	11
Average number of turns	5.0	4.8	4.4	4.0	4.1	4.1
Average number of words per u	55.8	55.8	52.7	98.2	107.2	110.4
Average number of words per r	67.3	68.8	67.7	91.0	100.1	94.6

Table 5). We fine-tune BERT[7] for sentence classification, using the CLS token[8]; the input is the concatenation of the dialogue context and the candidate response separated by SEP tokens. When training BERT we employ a balanced number of relevant and non-relevant context and response pairs[9]. We use cross entropy loss and the Adam optimizer [20] with learning rate of $5e-5$ and $\epsilon = 1e-8$.

For σ_{SM}, as word embeddings we use pre-trained fastText[10] embeddings with 300 dimensions and a maximum length of $k = 20$ words of dialogue contexts and responses. For σ_{BM25}, we use default values[11] of $k_1 = 1.5$, $b = 0.75$ and $\epsilon = 0.25$. For CL, we fix T as 90% percent of the total training iterations—this means that we continue training for the final 10% of iterations after introducing all samples—and the initial number of instances δ as 33% of the data to avoid sampling the same instances several times.

Evaluation To compare our strategies with the baseline where no CL is employed, for each approach we fine-tune BERT five times with different random seeds—to rule out that the results are observed only for certain random weight initialization values—and for each run we select the model with best observed effectiveness on the development set. The best model of each run is then applied to the test set. We report the effectiveness with respect to Mean Average Precision (MAP) like prior works [50,52]. We perform paired Student's t-tests between each scoring/pacing-function variant and the baseline run without CL.

5 Results

We first report the results for the pacing functions (Figure 3) followed by the main results (Table 5) comparing different scoring functions. We finish with an error analysis to understand when CL outperforms our no-curriculum baseline.

[7] We use the PyTorch-Transformers implementation `https://github.com/huggingface/pytorch-transformers` and resort to *bert-base-uncased* with default settings.

[8] The BERT authors suggest CLS as a starting point for sentence classification tasks [7].

[9] We observed similar results to training with 1 to 10 ratio in initial experiments.

[10] `https://fasttext.cc/docs/en/crawl-vectors.html`

[11] `https://radimrehurek.com/gensim/summarization/bm25.html`

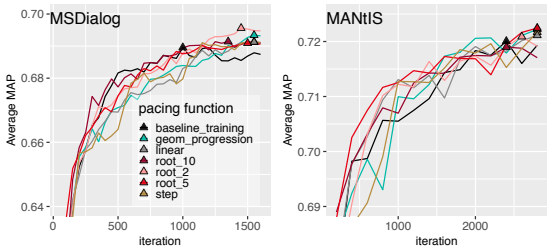

Fig. 3: Average development MAP for 5 different runs, using different curriculum learning pacing functions. △ is the maximum observed MAP.

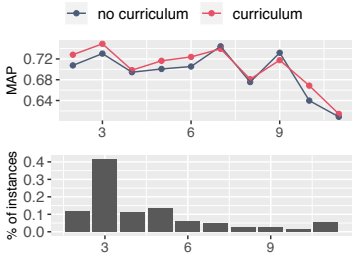

Fig. 4: `MSDialog` test set MAP of curriculum learning and baseline by number of turns.

Pacing Functions In order to understand how CL results are impacted by the pace we go from easy to hard instances, we evaluate the different proposed *pacing functions*. We display the evolution of the development set MAP (average of 5 runs) during training on Figure 3 (we use development MAP to track effectiveness during training). We fix the scoring function as $BERT_{pred}$; this is the best performing scoring function, more details in the next section. We see that the pacing functions with the maximum observed average MAP are *root_2* and *root_5* for `MSDialog` and `MANtIS` respectively[12]. The other pacing functions, *linear*, *geom_progression* and *step*, also outperform the standard training baseline with statistical significance on the test set and yield similar results to the *root_2* and *root_5* functions.

Our results are aligned with previous research on CL [30], that giving more time for the model to assimilate harder instances (by using a root pacing function) is beneficial to the curriculum strategy and is better than no CL with statistical significance on both development and test sets. For the rest of our experiments we fix the pacing function as *root_2*, the best pacing function for `MSDialog`. Let's now turn to the impact of the scoring functions.

Scoring Functions The most critical challenge of CL is defining a measure of difficulty of instances. In order to evaluate the effectiveness of our scoring functions we report the test set results across both datasets in Table 5. We observe that the scoring functions which do not use the relevance labels $\mathcal{Y}$ are not able to outperform the no CL baseline (*random* scoring function). They are based on features of the dialogue context $\mathcal{U}$ and responses $\mathcal{R}$ that we hypothesized make them difficult for a model to learn. Differently, for $\overline{BERT_{loss}}$ and $BERT_{pred}$ we observe statistically significant results on both datasets across different runs. They differ in two ways from the unsuccessful scoring functions: they have access

[12] If we increase the n of the root function to bigger values, e.g. *root_10*, the results drop and get closer to not using CL. This is due to the fact that higher n generate root functions with a similar shape to standard training, giving the same amount of time to easy and hard instances (cf. Figure 2).

Table 5: Test set MAP results of 5 runs using different curriculum learning scoring functions. Superscripts $^{\dagger}/^{\ddagger}$ denote statistically significant improvements over the baseline where no curriculum learning is applied ($f_{score} = random$) at 95%/99% confidence intervals. Bold indicates the highest MAP for each line.

				MSDialog				
run	$random$	$\#_{turns}$	$\overline{\#_{\mathcal{U}words}}$	$\overline{\#_{\mathcal{R}words}}$	σ_{SM}	σ_{BM25}	$BERT_{pred}$	$\overline{BERT_{loss}}$
1	0.7142	0.7220 †	0.7229 †	0.7182	0.7239 †‡	0.7175	**0.7272** †‡	0.7244 †‡
2	0.7044	0.7060	0.7053	0.6968	0.7032	0.7003	**0.7159** †‡	0.7194 †‡
3	0.7126	0.7215 †	0.7163	0.7171	0.7174	0.7159	**0.7296** †‡	0.7225 †‡
4	0.7031	0.7065	0.7043	0.6993	0.7026	0.6949	0.7154 †‡	**0.7204** †‡
5	0.7148	0.7225 †	0.7203	0.7169	0.7171	0.7134	0.7322 †‡	**0.7331** †‡
AVG	0.7098	0.7157	0.7138	0.7097	0.7128	0.7084	**0.7241**	0.7240
SD	0.0056	0.0086	0.0086	0.0106	0.0095	0.0101	0.0079	0.0055
				MANtIS				
1	0.7203	0.7192	0.7198	0.7194	0.7166	0.7200	0.7257 †‡	**0.7268** †‡
2	0.6984	0.6993	0.6989	0.6996	0.6964	0.7009	**0.7067** †‡	0.7051 †‡
3	0.7200	0.7197	0.7134	0.7206	0.7153	0.7153	**0.7282** †‡	0.7221
4	0.7114	0.7117	0.7002	0.6978	0.7140	0.7084	**0.7240** †‡	0.7184 †‡
5	0.7156	0.7174	0.7193 †	0.7162	0.7147	0.7185	**0.7264** †‡	0.7258 †‡
AVG	0.7131	0.7135	0.7103	0.7107	0.7114	0.7126	**0.7222**	0.7196
SD	0.0090	0.0085	0.0102	0.0111	0.0084	0.0079	0.0088	0.0088

to the training labels $\mathcal{Y}$ and the difficulty of an instance is based on what a previously trained model determines to be hard, and thus not our intuition.

Our results bear resemblance to Born Again Networks [10], where a student model which is identical in parameters and architecture to the teacher model outperforms the teacher when trained with knowledge distillation [15], i.e., using the predictions of the teacher model as labels for the student model. The difference here is that instead of transferring the knowledge from the teacher to the student through the labels, we transfer the knowledge by imposing a structure/order on the training set, i.e. curriculum learning.

Error Analysis In order to understand when CL performs better than random training samples, we fix the scoring ($BERT_{pred}$) ad pacing function ($root\_2$) and explore the test set effectiveness along several dimensions (cf. Figures 4 and 5). We report the results only for `MSDialog`, but the trends hold for `MANtIS` as well.

We first consider the number of turns in the conversation in Figure 4. CL outperforms the baseline approach for the types of conversations appearing most frequently (2-5 turns in `MSDialog`). The CL-based and baseline effectiveness drops for conversations with a large number of turns. This can be attributed to two factors: (1) employing pre-trained BERT in practice allows only a certain maximum number of tokens as input, so longer conversations can lose important

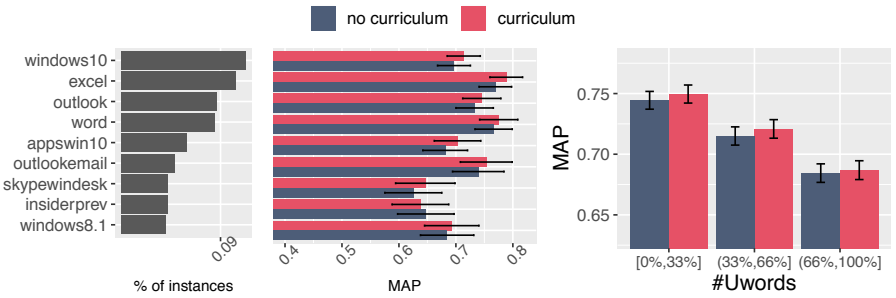

Fig. 5: Test set MAP for `MSDialog` across different domains (left) and instances' difficulty (right) according to $\overline{\#_{\mathcal{R}words}}$ for curriculum learning and the baseline.

information due to truncating; (2) for longer conversations it is harder to identify the important information to match in the history, i.e information spread.

Next, we look at different conversation domains in Figure 5 (left), such as *physics* and *askubuntu*—are the gains in effectiveness limited to particular domains? The error bars indicate the confidence intervals with confidence level of 95%. We list only the most common domains in the test set. The gains of CL are spread over different domains as opposed to concentrated on a single domain.

Lastly, using our scoring functions we sort the test instances and divide them into three buckets: first 33% instances, 33%–66% and 66%–100%. In Figure 5 (right), we see the effectiveness of CL against the baseline for each bucket using $\overline{\#_{\mathcal{U}words}}$ (the same trend holds for the other scoring functions). As we expect, the bucket with the most difficult instances according to the scoring function is the one with lowest MAP values. Finally, the improvements of CL over the baseline are again spread across the buckets, showing that CL is able to improve over the baseline for different levels of difficulty.

6 Conclusions

In this work we studied whether CL strategies are beneficial for neural ranking models. We find supporting evidence for curriculum learning in IR. Simply reordering the instances in the training set using a difficulty criteria leads to effectiveness improvements, requiring no changes to the model architecture—a similar relative improvement in MAP has justified novel neural architectures in the past [50,61,62,43]. Our experimental results on two conversation response ranking datasets reveal (as one might expect) that it is best to use all available information $(\mathcal{U}, \mathcal{R}, \mathcal{Y})$ as evidence for instance difficulty. Future work directions include considering other retrieval tasks, different neural architectures and an investigation of the underlying reasons for CL's workings.

Acknowledgements This research has been supported by NWO projects SearchX (639.022.722) and NWO Aspasia (015.013.027).

References

1. Bengio, Y., Louradour, J., Collobert, R., Weston, J.: Curriculum Learning. In: ICML. pp. 41–48 (2009)
2. Breiman, L.: Arcing classifier. The annals of statistics **26**(3), 801–849 (1998)
3. Burges, C.J.: From ranknet to lambdarank to lambdamart: An overview. Learning **11**(23-581), 81 (2010)
4. Chang, H.S., Learned-Miller, E., McCallum, A.: Active bias: Training more accurate neural networks by emphasizing high variance samples. In: NeurIPS. pp. 1002–1012 (2017)
5. Chen, X., Gupta, A.: Webly supervised learning of convolutional networks. In: ICCV. pp. 1431–1439 (2015)
6. Cohn, D.A., Ghahramani, Z., Jordan, M.I.: Active learning with statistical models. Journal of artificial intelligence research **4**, 129–145 (1996)
7. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In: NAACL. pp. 4171–4186 (2019)
8. Elman, J.L.: Learning and development in neural networks: The importance of starting small. Cognition **48**(1), 71–99 (1993)
9. Ferro, N., Lucchese, C., Maistro, M., Perego, R.: Continuation Methods and Curriculum Learning for Learning to Rank. In: CIKM. pp. 1523–1526 (2018)
10. Furlanello, T., Lipton, Z., Tschannen, M., Itti, L., Anandkumar, A.: Born-Again Neural Networks. In: ICML. pp. 1602–1611 (2018)
11. Gong, C., Tao, D., Maybank, S.J., Liu, W., Kang, G., Yang, J.: Multi-modal curriculum learning for semi-supervised image classification. IEEE Transactions on Image Processing **25**(7), 3249–3260 (2016)
12. Gui, L., Baltrušaitis, T., Morency, L.P.: Curriculum learning for facial expression recognition. In: FG. pp. 505–511 (2017)
13. Guo, J., Fan, Y., Ai, Q., Croft, W.B.: A deep relevance matching model for ad-hoc retrieval. In: CIKM. pp. 55–64 (2016)
14. Hacohen, G., Weinshall, D.: On the power of curriculum learning in training deep networks. arXiv preprint arXiv:1904.03626 (2019)
15. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531 (2015)
16. Hu, B., Lu, Z., Li, H., Chen, Q.: Convolutional neural network architectures for matching natural language sentences. In: NeurIPS. pp. 2042–2050 (2014)
17. Huang, P.S., He, X., Gao, J., Deng, L., Acero, A., Heck, L.: Learning deep structured semantic models for web search using clickthrough data. In: CIKM. pp. 2333–2338 (2013)
18. Jia, R., Liang, P.: Adversarial Examples for Evaluating Reading Comprehension Systems. In: EMNLP. pp. 2021–2031 (2017)
19. Jinfeng Rao, Linqing Liu, Y.T.W.Y.P.S., Lin, J.: Bridging the Gap Between Relevance Matching and Semantic Matching for Short Text Similarity Modeling. EMNLP (2019)
20. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
21. Kocmi, T., Bojar, O.: Curriculum learning and minibatch bucketing in neural machine translation. RANLP pp. 379–386 (2017)
22. Kumar, M.P., Packer, B., Koller, D.: Self-paced learning for latent variable models. In: NeurIPS. pp. 1189–1197 (2010)

23. Liu, C., He, S., Liu, K., Zhao, J.: Curriculum Learning for Natural Answer Generation. In: IJCAI. pp. 4223–4229 (2018)
24. Liu, T.Y., et al.: Learning to rank for information retrieval. Foundations and Trends® in Information Retrieval **3**(3), 225–331 (2009)
25. Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., Stoyanov, V.: Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692 (2019)
26. Lowe, R., Pow, N., Serban, I., Pineau, J.: The Ubuntu Dialogue Corpus: A Large Dataset for Research in Unstructured Multi-Turn Dialogue Systems. In: SIGDIAL. pp. 285–294 (2015)
27. Nie, Y., Li, Y., Nie, J.Y.: Empirical study of multi-level convolution models for ir based on representations and interactions. In: SIGIR. pp. 59–66 (2018)
28. Nogueira, R., Cho, K.: Passage Re-ranking with BERT. arXiv preprint arXiv:1901.04085 (2019)
29. Penha, G., Balan, A., Hauff, C.: Introducing MANtIS: a novel Multi-Domain Information Seeking Dialogues Dataset. arXiv preprint arXiv:1912.04639 (2019)
30. Platanios, E.A., Stretcu, O., Neubig, G., Poczos, B., Mitchell, T.: Competence-based Curriculum Learning for Neural Machine Translation. In: NAACL. pp. 1162–1172 (2019)
31. Qu, C., Yang, L., Croft, W.B., Zhang, Y., Trippas, J., Qiu, M.: User Intent Prediction in Information-seeking Conversations. In: CHIIR (2019)
32. Qu, C., Yang, L., Croft, W.B., Trippas, J.R., Zhang, Y., Qiu, M.: Analyzing and characterizing user intent in information-seeking conversations. In: SIGIR. pp. 989–992 (2018)
33. Qu, C., Yang, L., Qiu, M., Croft, W.B., Zhang, Y., Iyyer, M.: BERT with History Answer Embedding for Conversational Question Answering. pp. 1133–1136. SIGIR (2019)
34. Ranjan, S., Hansen, J.H., Ranjan, S., Hansen, J.H.: Curriculum learning based approaches for noise robust speaker recognition. TASLP **26**(1), 197–210 (2018)
35. Rohde, D.L., Plaut, D.C.: Language acquisition in the absence of explicit negative evidence: How important is starting small? Cognition **72**(1), 67–109 (1999)
36. Sachan, M., Xing, E.: Easy questions first? a case study on curriculum learning for question answering. In: ACL. vol. 1, pp. 453–463 (2016)
37. Sakata, W., Shibata, T., Tanaka, R., Kurohashi, S.: FAQ Retrieval using Query-Question Similarity and BERT-Based Query-Answer Relevance. arXiv preprint arXiv:1905.02851 (2019)
38. Shen, Y., He, X., Gao, J., Deng, L., Mesnil, G.: A latent semantic model with convolutional-pooling structure for information retrieval. In: CIKM. pp. 101–110 (2014)
39. Shrivastava, A., Gupta, A., Girshick, R.: Training region-based object detectors with online hard example mining. In: CVPR. pp. 761–769 (2016)
40. Shtok, A., Kurland, O., Carmel, D.: Predicting query performance by query-drift estimation. In: ICTIR. pp. 305–312 (2009)
41. Soviany, P., Ardei, C., Ionescu, R.T., Leordeanu, M.: Image Difficulty Curriculum for Generative Adversarial Networks (CuGAN). arXiv preprint arXiv:1910.08967 (2019)
42. Subramanian, S., Rajeswar, S., Dutil, F., Pal, C., Courville, A.: Adversarial Generation of Natural Language. In: Rep4NLP. pp. 241–251 (2017)
43. Tao, C., Wu, W., Xu, C., Hu, W., Zhao, D., Yan, R.: One Time of Interaction May Not Be Enough: Go Deep with an Interaction-over-Interaction Network for Response Selection in Dialogues. In: ACL. pp. 1–11 (2019)

44. Tong, S., Koller, D.: Support vector machine active learning with applications to text classification. Journal of machine learning research **2**(Nov), 45–66 (2001)
45. Tudor Ionescu, R., Alexe, B., Leordeanu, M., Popescu, M., Papadopoulos, D.P., Ferrari, V.: How hard can it be? Estimating the difficulty of visual search in an image. In: CVPR. pp. 2157–2166 (2016)
46. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: NeurIPS. pp. 5998–6008 (2017)
47. Wan, S., Lan, Y., Guo, J., Xu, J., Pang, L., Cheng, X.: A deep architecture for semantic matching with multiple positional sentence representations. In: AAAI. pp. 2835–2841 (2016)
48. Wan, S., Lan, Y., Xu, J., Guo, J., Pang, L., Cheng, X.: Match-SRNN: modeling the recursive matching structure with spatial RNN. In: IJCAI. pp. 2922–2928. AAAI Press (2016)
49. Weinshall, D., Cohen, G., Amir, D.: Curriculum Learning by Transfer Learning: Theory and Experiments with Deep Networks. In: ICML. pp. 5235–5243 (2018)
50. Wu, Y., Wu, W., Xing, C., Zhou, M., Li, Z.: Sequential Matching Network: A New Architecture for Multi-turn Response Selection in Retrieval-Based Chatbots. In: ACL. vol. 1, pp. 496–505 (2017)
51. Yang, L., Hu, J., Qiu, M., Qu, C., Gao, J., Croft, W.B., Liu, X., Shen, Y., Liu, J.: A hybrid retrieval-generation neural conversation model. arXiv preprint arXiv:1904.09068 (2019)
52. Yang, L., Qiu, M., Qu, C., Guo, J., Zhang, Y., Croft, W.B., Huang, J., Chen, H.: Response ranking with deep matching networks and external knowledge in information-seeking conversation systems. In: SIGIR. pp. 245–254 (2018)
53. Yang, W., Lu, K., Yang, P., Lin, J.: Critically Examining theNeural Hype: Weak Baselines and the Additivity of Effectiveness Gains from Neural Ranking Models. In: SIGIR. pp. 1129–1132. New York, NY, USA (2019)
54. Yang, W., Xie, Y., Lin, A., Li, X., Tan, L., Xiong, K., Li, M., Lin, J.: End-to-End Open-Domain Question Answering with BERTserini. In: NAACL. pp. 72–77 (2019)
55. Yang, W., Zhang, H., Lin, J.: Simple applications of bert for ad hoc document retrieval. arXiv preprint arXiv:1903.10972 (2019)
56. Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R., Le, Q.V.: XLNet: Generalized Autoregressive Pretraining for Language Understanding. arXiv preprint arXiv:1906.08237 (2019)
57. Yilmaz, Z.A., Yang, W., Zhang, H., Lin, J.: Cross-Domain Modeling of Sentence-Level Evidence for Document Retrieval. pp. 3481–3487. EMNLP (2019)
58. Yu, Y., Zhang, W., Hasan, K., Yu, M., Xiang, B., Zhou, B.: End-to-end answer chunk extraction and ranking for reading comprehension. arXiv preprint arXiv:1610.09996 (2016)
59. Zhang, D., Kim, J., Crego, J., Senellart, J.: Boosting Neural Machine Translation. In: IJCNLP. pp. 271–276 (2017)
60. Zhang, X., Kumar, G., Khayrallah, H., Murray, K., Gwinnup, J., Martindale, M.J., McNamee, P., Duh, K., Carpuat, M.: An empirical exploration of curriculum learning for neural machine translation. arXiv preprint arXiv:1811.00739 (2018)
61. Zhang, Z., Li, J., Zhu, P., Zhao, H., Liu, G.: Modeling Multi-turn Conversation with Deep Utterance Aggregation. In: ACL. pp. 3740–3752 (2018)
62. Zhou, X., Li, L., Dong, D., Liu, Y., Chen, Y., Zhao, W.X., Yu, D., Wu, H.: Multi-turn response selection for chatbots with deep attention matching network. In: ACL. pp. 1118–1127 (2018)