Curriculum Learning for Natural Language Understanding

Benfeng Xu[1]*, Licheng Zhang[1]* , Zhendong Mao[1]†, Quan Wang[2] ,
Hongtao Xie[1] and Yongdong Zhang[1]

[1]School of Information Science and Technology,
University of Science and Technology of China, Hefei, China
[2]Beijing Research Institute,
University of Science and Technology of China, Beijing, China
{benfeng,zlczlc}@mail.ustc.edu.cn, quanwang1012@gmail.com
{zdmao,htxie,zhyd73}@ustc.edu.cn

Abstract

With the great success of pre-trained language models, the pretrain-finetune paradigm now becomes the undoubtedly dominant solution for natural language understanding (NLU) tasks. At the fine-tune stage, target task data is usually introduced in a completely random order and treated equally. However, examples in NLU tasks can vary greatly in difficulty, and similar to human learning procedure, language models can benefit from an easy-to-difficult curriculum. Based on this idea, we propose our Curriculum Learning approach. By reviewing the trainset in a crossed way, we are able to distinguish easy examples from difficult ones, and arrange a curriculum for language models. Without any manual model architecture design or use of external data, our Curriculum Learning approach obtains significant and universal performance improvements on a wide range of NLU tasks.

1 Introduction

Natural Language Understanding (NLU), which requires machines to understand and reason with human language, is a crucial yet challenging problem. Recently, language model (LM) pre-training has achieved remarkable success in NLU. Pre-trained LMs learn universal language representations from large-scale unlabeled data, and can be simply fine-tuned with a few adjustments to adapt to various NLU tasks, showing consistent and significant improvements in these tasks (Radford et al., 2018; Devlin et al., 2018).

While lots of attention has been devoted to designing better pre-training strategies (Yang et al., 2019; Liu et al., 2019; Raffel et al., 2019), it is also valuable to explore how to more effectively solve downstream NLU tasks in the fine-tuning stage.

Easy cases:	
easy, comfortable	positive
most purely enjoyable	positive
most plain, unimaginative	negative
badly edited	negative
Hard cases:	
why didn't Hollywood think of this sooner	positive
I simply can't recommend it enough	positive
supposedly funny movie	negative
occasionally interesting	negative

Table 1: Examples from SST-2 sentiment classification task. Difficulty levels are determined by our review method (detailed later).

Most current approaches perform fine-tuning in a straightforward manner, i.e., all training examples are treated equally and presented in a completely random order during training. However, even in the same NLU task, the training examples could vary significantly in their difficulty levels, with some easily solvable by simple lexical clues while others requiring sophisticated reasoning. Table 1 shows some examples from the SST-2 sentiment classification task (Socher et al., 2013), which identifies sentiment polarities (positive or negative) of movie reviews. The easy cases can be solved directly by identifying sentiment words such as "comfortable" and "unimaginative", while the hard ones further require reasoning with negations or verb qualifiers like "supposedly" and "occasionally". Extensive research suggests that presenting training examples in a meaningful order, starting from easy ones and gradually moving on to hard ones, would benefit the learning process, not only for humans but also for machines (Skinner, 1958; Elman, 1993; Peterson, 2004; Krueger and Dayan, 2009).

Such an organization of learning materials in human learning procedure is usually referred to as ***Curriculum***. In this paper, we draw inspiration from similar ideas, and propose our approach

*Equal contribution.
†Corresponding author.

Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 6095–6104
July 5 - 10, 2020.

for arranging a curriculum when learning NLU tasks. Curriculum Learning (CL) is first proposed by (Bengio et al., 2009) in machine learning area, where the definition of easy examples is established ahead, and an easy-to-difficult curriculum is arranged accordingly for the learning procedure. Recent developments have successfully applied CL in computer vision areas (Jiang et al., 2017; Guo et al., 2018; Hacohen and Weinshall, 2019). It is observed in these works that by excluding the negative impact of difficult or even noisy examples in early training stage, an appropriate CL strategy can guide learning towards a better local minima in parameter space, especially for highly non-convex deep models. We argue that language models like transformer, which is hard to train (Popel and Bojar, 2018), should also benefit from CL in the context of learning NLU tasks, and such idea still remains unexplored.

The key challenge in designing a successful CL strategy lies in how to define easy/difficult examples. One straightforward way is to simply predefine the difficulty in revised rules by observing the particular target task formation or training data structure accordingly (Guo et al., 2018; Platanios et al., 2019; Tay et al., 2019). For example, (Bengio et al., 2009) utilized an easier version of shape recognition trainset which comprised of less varied shapes, before the training of complex one started. More recently, (Tay et al., 2019) considered the paragraph length of a question answering example as its reflection of difficulty. However, such strategies are highly dependent on the target dataset itself and often fails to generalize to different tasks.

To address this challenge, we propose our Cross Review method for evaluating difficulty. Specifically, we define easy examples as those well solved by the exact model that we are to employ in the task. For different tasks, we adopt their corresponding golden metrics to calculate a difficulty score for each example in the trainset. Then based on these difficulty scores, we further design a re-arranging algorithm to construct the learning curriculum in an annealing style, which provides a soft transition from easy to difficult for the model. In general, our CL approach is not constrained to any particular task, and does not rely on human prior heuristics about the task or dataset.

Experimental results show that our CL approach can greatly help language models learn in their finetune stage. Without any task-tailored model architecture design or use of external data, we are able to obtain significant and universal improvements on a wide range of downstream NLU tasks. Our contributions can be concluded as follows:

- We explore and demonstrate the effectiveness of CL in the context of finetuning LM on NLU tasks. To the best of our knowledge, this is one of the first times that CL strategy is proved to be extensively prospective in learning NLU tasks.
- We propose a novel CL framework that consists of a Difficulty Review method and a Curriculum Arrangement algorithm, which requires no human pre-design and is very generalizable to a lot of given tasks.
- We obtain universal performance gain on a wide range of NLU tasks including Machine Reading Comprehension (MRC) and Natural Language Inference. The improvements are especially significant on tasks that are more challenging.

2 Preliminaries

We describe our CL approach using BERT (Devlin et al., 2018), the most influential pre-trained LM that achieved state-of-the-art results on a wide range of NLP tasks. BERT is pretrained using *Masked Language Model* task and *Next sentence Prediction* task via large scale corpora. It consists of a hierarchical stack of l self-attention layers, which takes an input of a sequence with no more than 512 tokens and output the contextual representation of a H-dimension vector for each token in position i, which we denote as $\boldsymbol{h}_i^l \in \mathbb{R}^H$. In natural language understanding tasks, the input sequences usually start with special token $\langle\text{CLS}\rangle$, and end with $\langle\text{SEP}\rangle$, for sequences consisting of two segments like in pairwise sentence tasks, another $\langle\text{SEP}\rangle$ is added in between for separating usage.

For target benchmarks, we employ a wide range of NLU tasks, including machine reading comprehension, sequence classification and pairwise text similarity, etc.. Following (Devlin et al., 2018), we adapt BERT for NLU tasks in the most straightforward way: simply add one necessary linear layer upon the final hidden outputs, then finetune the entire model altogether. Specifically, we brief the configurations and corresponding metrics for different tasks employed in our algorithms as follows:

Machine Reading Comprehension In this work we consider the *extractive* MRC task. Given a passage P and a corresponding question Q, the goal is to extract a continuous span $\langle p_{start}, p_{end} \rangle$ from P as the answer A, where the $start$ and end are its boundaries.

We pass the concatenation of the question and paragraph $[\langle \text{CLS} \rangle, Q, \langle \text{SEP} \rangle, P, \langle \text{SEP} \rangle]$ to the pre-trained LM and use a linear classifier on top of it to predict the answer span boundaries.

For the $i-th$ input token, the probabilities that it is the start or end are calculated as:

$$[logit_i^{start}, logit_i^{end}]^T = \boldsymbol{W}_{MRC}^T \boldsymbol{h}_i^l$$
$$p_i^{start} = \text{softmax}(\{logit_i^{start}\})$$
$$p_i^{end} = \text{softmax}(\{logit_i^{end}\})$$

where $\boldsymbol{W}_{MRC}^T \in \mathbb{R}^{2\times H}$ is a trainable matrix. The training objective is the log-likelihood of the true start and end positions y_{start} and y_{end}:

$$loss = -(\log(p_{y_{start}}^{start}) + \log(p_{y_{end}}^{end}))$$

For unanswerable questions, the probability is calculated as $s_{\text{un}} = p_{cls}^{start} + p_{cls}^{end}$ using $\langle \text{CLS} \rangle$ representation. We classify a question into unanswerable when $s_{\text{un}} > s_{i,j} = \max_{i \le j}(p_i^{start} + p_j^{end})$. **F1** is used as the golden metric.

Sequence Classification We consider the final contextual embedding of $\langle \text{CLS} \rangle$ token $\boldsymbol{h}_0^l$ as the pooled representation of the whole input sequence S. The probability that the input sequence belongs to label c is calculated by a linear output layer with parameter matrix $\boldsymbol{W}_{SC} \in \mathbb{R}^{K\times H}$ following a softmax:

$$P(c|S) = \text{softmax}(\boldsymbol{h}_0^l \boldsymbol{W}_{SC}^T),$$

where K is the number of classes. The log-likelihood is also used as the training objective for this task. **Accuracy** is considered as the golden metric.

Pairwise Text Similarity Similar to sequence classification task, final embedding of $\langle \text{CLS} \rangle$ token $\boldsymbol{h}_0^l$ is used to represent the input text pair (T_1, T_2). A parameter vector $\boldsymbol{W}_{PTS} \in \mathbb{R}^H$ is introduced to compute the similarity score:

$$\text{Similarity}(T_1, T_2) = \boldsymbol{h}_0^l \boldsymbol{W}_{PTS}^T.$$

For this task, we use **Mean Squared Error (MSE)** as the training objective and also the golden metric:

$$MSE = (y - \text{Similarity}(T_1, T_2))^2,$$

where y is the similarity label in continuous score.

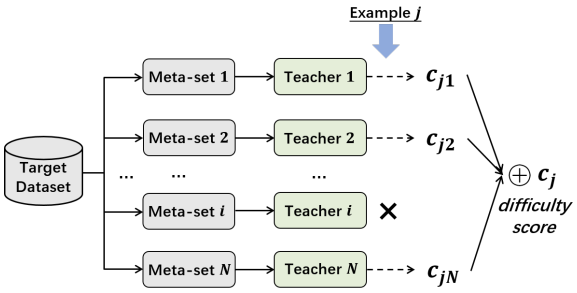

Figure 1: Our Cross Review method: the target dataset is split into N meta-datasets, after the teachers are trained on them, each example will be inferenced by all other teachers (except the one it belongs to), the scores will be summed as the final evaluation results.

3 Our CL Approach

We decompose our CL framework into two stages: **Difficulty Evaluation** and **Curriculum Arrangement**. For any target task, let D be the examples set used for training, and Θ be our language model which is expected to fit D. In the first stage, the goal is to assign each example d_j in D with a score c_j which reflects its difficulty with respect to the model. We denote C as the whole difficulty score set corresponding to trainset D. In the second stage, based on these scores, D is organized into a sequence of ordered learning stages $\{S_i : i = 1, 2, \ldots, N\}$ with an easy-to-difficult fashion, resulting in the final curriculum where the model will be trained on. We will elaborate these two stages in section 3.1 and 3.2 respectively.

3.1 Difficulty Evaluation

The difficulty of a textual example reflects itself in many ways, e.g., the length of the context, the usage of rare words, or the scale of learning target. Although such heuristics seems reasonable to human, the model itself may not see it the same way. So we argue that difficulty score as the intrinsic properties of an example should be decided by the model itself, and the best metric should be the golden metric of the target task, which can be accuracy, F1 score, etc., as we introduced in section 2.

To perform difficulty evaluation, we first scatter our trainset D into N shares uniformly as $\{\widetilde{D}_i : i = 1, 2, \ldots, N\}$, and train N corresponding models $\{\widetilde{\Theta}_i : i = 1, 2, \ldots, N\}$ on them, which are all identical to Θ (note that each model $\widetilde{\Theta}_i$ will only see $1/N$ of the entire trainset). We refer to these N models as ***teachers***, and $\{\widetilde{D}_i\}$ as ***meta-datasets*** for that they are attended only to collect

information (i.e. the extent of difficulty) about the original trainset D. This preparing of teacher can be formulated as:

$$\widetilde{\Theta}_i = \underset{\widetilde{\Theta}_i}{\operatorname{argmin}} \sum_{d_j \in \widetilde{D}_i} L(d_j, \widetilde{\Theta}_i)$$
$$i = 1, 2, \ldots, N$$

where L indicates the loss function. After every teacher is respectively trained on its meta-dataset, the evaluation of trainset D should begin.

For each example d_j, it should be included in one and only one meta-dataset, let's assume it's $\widetilde{D}_k$, then we perform inference of d_j on all teachers except teacher k, because the inference from teacher k is supposed to be isolated with the meta-dataset $\widetilde{D}_k$ it has already seen during training. After all inferences finished, we calculate scores of d_j in the target task's metric, resulting $N-1$ scores from $N-1$ different teachers:

$$c_{ji} = M(\widetilde{\Theta}_i(x_j), y_j)$$

where $\widetilde{\Theta}_i(\bullet)$ represents the inference function, x_j and y_j is the input and label of example d_j respectively, M is the metric calculation formula, which can be either F1, Accuracy or MSE for different tasks as introduced in section 2, and c_{ji} is the score of d_j from teacher $\widetilde{\Theta}_i$. Finally, we define the difficulty score of d_j as the integration of all $N-1$ scores:

$$c_j = \sum_{i \in (1,\ldots,N),\ i \neq k} c_{ji}$$

with all scores calculated, we obtain the final difficulty score set C as desired. We refer to our difficulty evaluation method as **Cross Review**(see Fig. 1)

In the proposed method, the teacher models perform their inferences in a crossed way, which prevents the meta-dataset from contaminating the inference set. Besides, each example gets its score from multi teachers, thus the fluctuation of evaluation results is greatly alleviated. In general, our Cross Review method can address the difficulty evaluation problem in an elegant design.

3.2 Curriculum Arrangement

In this section we describe our method to arrange the training examples D into a learning curriculum according to their difficulty scores C. We design our curriculum in a multi-stage setting $\{S_i : i = 1, 2, \ldots, N\}$. Within each stage S_i, the examples are still shuffled to keep local stochastics, and examples from different stages do not overlap in order to prevent overfitting.

The sampling algorithm is built upon such principle:

> ***The proportion of difficult examples in each stage should start with 0, and gradually increase until it reachs how much it accounts for in the original dataset distribution.***

We first sort all examples by their difficulty score C, and divide them into N buckets: $\{C_i : i = 1, 2, \ldots, N\}$, so the examples are now collected into N different levels of difficulty, ranging from C_1 (the easiest) to C_N (the hardest), with the proportion distribution as:

$$num(C_1) : num(C_2) : \cdots : num(C_N)$$

For tasks with discrete metrics, such distribution is naturally formed by the difficulty score hierarchy, and directly reflects the intrinsic difficulty distribution of the dataset. For other tasks, we manually divide C uniformly[1]. Based on these buckets, we construct the learning curriculum one stage after another. For each learning stage S_i, we sample examples from all antecedent buckets $\{C_j : j = 1, 2, \ldots, i\}$ by the following proportion:

$$\frac{1}{N}num(C_1) : \frac{1}{N}num(C_2) : \cdots : \frac{1}{N}num(C_i)$$

and the final curriculum $\{S_i : i = 1, 2, \ldots, N\}$ is formed as such. We refer to the arrangement algorithm as **Annealing** method for it provides a soft transition through multi learning stages.

At each stage, the model is trained for one epoch. When the training reached S_N, the model should be ready for the original distribution in trainset D, so we finally add another stage S_{N+1} which covers the entire trainset, and the model is trained on it until converges.

4 Experiments

4.1 Datasets

In this section we briefly describe three popular NLU benchmarks on which we evaluate our CL approach: SQuAD 2.0 (Rajpurkar et al., 2018), NewsQA (Trischler et al., 2016) and GLUE (Wang et al., 2018), their scale and metrics are detailed in Table 2.

[1]Please refer to our implementation detail for selected tasks in section 4.2

	SQuAD2.0	NewsQA	MNLI-m	QNLI	QQP	RTE	SST-2	MRPC	CoLA	STS-B
Train	130.3k	92.5k	392.7k	104.7k	363.8k	2.5k	67.3k	3.7k	8.6k	5.7k
Dev	11.9k	5.2k	9.8k	5.5k	40.4k	277	872	408	1.0k	1.5k
Test	8.9k	5.1k	9.8k	5.5k	39.1k	3.0k	1.8k	1.7k	1.0k	1.4k
Metrics	F1/EM	F1/EM	Accuracy	Accuracy	Accuracy	Accuracy	Accuracy	F1	Matthew	Pearson

Table 2: The number of training, development, test examples and metrics of tasks used in this work.

SQuAD The Stanford Question Answering Dataset (SQuAD), constructed using Wikipedia articles, is a well known extractive machine reading comprehension dataset with two tasks: SQuAD1.1 (Rajpurkar et al., 2016) and SQuAD 2.0 (Rajpurkar et al., 2018). The latest 2.0 version also introduced unanswerable questions, making it a more challenging and practical task. In this paper, We take SQuAD 2.0 as our testbed.

NewsQA NewsQA (Trischler et al., 2016) is also a MRC dataset in extractive style but is much more challenging, with human performance at 0.694 F1 score. NewsQA is collected from news articles of CNN with two sets of crowdworkers, the "questioners" is provided with the article's headline only, and "answerers" is supposed to find the answer in full article. We ignore examples flagged to be without annotator agreement for better evaluation following (Fisch et al., 2019).

GLUE The General Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2018) is a collection of nine[2] diverse sentence or sentence pair language understanding tasks including sentiment analysis, textual entailment, sentence similarity, etc. It is considered as a well-designed benchmark that can evaluate the generalization and robustness of NLU algorithms. The labels for GLUE test set is hidden, and users must upload their predictions to obtain evaluation results, the submission is limited to protect test set from overfitting.

4.2 Experimental Setups

We use BERT Large (Devlin et al., 2018) as our pre-trained language model to demonstrate the effectiveness of our CL approach. For MRC, we also test on BERT Base model for more comprehensive results. Besides reported results from literature, we also provide our re-implementation on all datasets, which form a more competitive baseline for comparison. The only difference between our re-implementation and our CL approach is the arrangement of curriculum, i.e., the order of training examples.

Method	SQuAD 2.0		NewsQA	
	EM	F1	EM	F1
BERT Base	73.66	76.30	-	-
BERT Base*	73.66	76.78	47.70	60.10
BERT Base+CL	**74.96**	**77.93**	**47.72**	**60.57**
BERT Large	78.98	81.77	-	-
BERT Large*	79.12	82.09	50.40	64.12
BERT Large+CL	**79.43**	**82.66**	**50.50**	**64.42**

Table 3: Results on SQuAD 2.0 and NewsQA, all on development sets. Baseline on SQuAD 2.0 is obtained from (Yang et al., 2019), * indicates our re-implementation.

To obtain a more comparable and stable difficulty score, we binarize the review results before sum them together if possible. For accuracy as metric, the score c_{ji} is already binary in instance level, for F1 as metric, we count any review result $c_{ji} > 0$ as correct. For other continuous metrics (MSE in this paper), we sum c_{ji} directly. We empirically choose $N = 10$ as the number of meta-datasets for most tasks (also the number of difficulty level and the number of stages), for three datasets with rather limited scale (RTE, MRPC, and STS-B), we change it to $N = 3$. The scale of all datasets employed in this work is provided in Table 2. Intuitively, we shall get better results by searching for the best N, we leave it to future works due to limited computation resource.

We implement our approach based on the PyTorch implementation of BERT (Wolf et al., 2019). We use Adam (Kingma and Ba, 2014) optimizer

[2]The benchmark consists of: Multi-Genre NLI (MNLI) (Williams et al., 2018), Quora Question Pairs (QQP) (Shankar Iyer, 2016), Question NLI (QNLI) (Rajpurkar et al., 2016), Stanford Sentiment Treebank (SST) (Socher et al., 2013), Corpus of Linguistic Acceptability (CoLA) (Warstadt et al., 2019), Semantic Textual Similarity Benchmark (STS-B) (Cer et al., 2017), Microsoft Research Paragraph Corpus (MRPC) (Dolan and Brockett, 2005), Recognizing Textual Entailment (RTE) (Bentivogli et al., 2009), Winograd NLI (WNLI) (Levesque et al., 2012)

	MNLI-m	QNLI	QQP	RTE	SST-2	MRPC	CoLA	STS-B	Avg
results on dev									
BERT Large	**86.6**	92.3	91.3	70.4	93.2	88.0	60.6	90.0	84.1
BERT Large*	**86.6**	92.5	91.5	74.4	93.8	91.7	63.5	90.2	85.5
BERT Large+CL	**86.6**	**92.8**	**91.8**	**76.2**	**94.2**	**91.9**	**66.8**	**90.6**	**86.4**
results on test									
BERT Large	**86.7**	91.1	89.3	70.1	**94.9**	89.3	60.5	87.6	83.7
BERT Large*	86.3	92.2	**89.5**	70.2	94.4	89.3	60.5	87.3	83.7
BERT Large+CL	**86.7**	**92.5**	**89.5**	**70.7**	94.6	**89.6**	**61.5**	**87.8**	**84.1**

Table 4: Results on GLUE benchmark, * indicates our re-implementation, baselines on dev sets are obtained from (Liu et al., 2019), baselines on test sets are obtained from the leaderboard (https://gluebenchmark.com/leaderboard) submitted by (Devlin et al., 2018), they may have taken different hyperparmeters. All results are produced with single task and single model.

with eplison equals to 1e-8. The learning rates warm up over the first 5% steps and then decay linearly to 0 for all experiments. To construct our re-implementation, on both SQuAD 2.0 and NewsQA we perform hyperparameter search with batch size in {16, 32} and learning rate in {1e-5, 2e-5, 3e-5, 4e-5} for Base model, and {32, 48, 64}, {5e-5, 6e-5, 7e-5} for Large model. We reuse the best parameter setting in SQuAD 2.0 on NewsQA. We set the max length of input sequence to 512 for NewsQA task because the paragraph is much more longer. On GLUE, we implement the experiments on Large model with batch size in {16, 32} and learning rate in {1e-5, 2e-5, 3e-5}.

4.3 MRC Results

The results for MRC tasks are presented in Table 3. In all experiments, our CL approach outperforms its baseline with considerable margin. On SQuAD 2.0, we obtain **+1.30 EM/+1.15 F1** improvements using base model and **+0.31 EM/+0.57 F1** using large model compare to our competitive re-implemented baseline. Note that the performance gain is more significant with Base model. On NewsQA, we also get **+0.02 EM/+0.47 F1** and **+0.10 EM/+0.30 F1** improvements for base and large model respectively.

4.4 GLUE Results

We summarize our GLUE results in Table 4. Results on dev sets show that our CL method consistently outperforms their competitive baseline on all 8 tasks, which proves that our CL is not only robustly effective but also generalizable on a wide range of NLU tasks. Because the model architecture and hyper-parameters setting are identical, all the performance gains can be attributed to our CL approach alone.

Specifically, we observe that our CL approach is doing better on more challenging tasks. For CoLA and RTE, the margin is up to **+3.3** and **+1.8** in respective metrics, which is relatively larger than less challenging tasks where the model performance already reached a plateau. Such results are understandable: when learning harder tasks, the model can be overwhelmed by very difficult examples at early stages, and a well-arranged curriculum thus can be more helpful. And for tasks where the baselines are already approaching the human performance like SST-2, our CL approach is still able to provide another **+0.4** improvements, which demonstrates the robustness of our approach. Overall, our CL approach obtains **+0.9 average score** gain on GLUE benchmark compare to our re-implemented baseline.

Results on test sets further demonstrate the effectiveness of our approach. We obtain **+0.4 average score** gain compare to our re-implementation and the baseline on the leaderboard.

4.5 Ablation Study

In this section, we delve into our approach on a series of interesting topics including: (**i**) what is the best CL design strategy for NLU tasks, (**ii**) can Cross Review really distinguish easy examples from difficult ones, (**iii**) the best choice of N. We choose SQuAD 2.0 task in most experiments for generality, and all experiments are performed with BERT Base model.

Comparison with Heuristic CL Methods To demonstrate our advantage over manually designed CL methods, we compare our approach with sev-

Method	SQuAD 2.0 EM	SQuAD 2.0 F1	Δ
No Curriculum	-	76.30	-
No Curriculum*	73.66	76.78	-
Rarity+**Annealing**	73.75	76.90	+0.12
Answer+**Annealing**	74.02	77.15	+0.37
Question+**Annealing**	74.35	77.37	+0.59
Paragraph+**Annealing**	74.45	77.54	+0.76
Cross-Review+*Naive order*	74.31	77.29	+0.51
Cross-Review+Annealing	**74.96**	**77.93**	**+1.15**

Table 5: Comparisions with heuristic CL design (written in *italic*). * indicates our re-implementation, Δ indicates absolute improvements on F1.

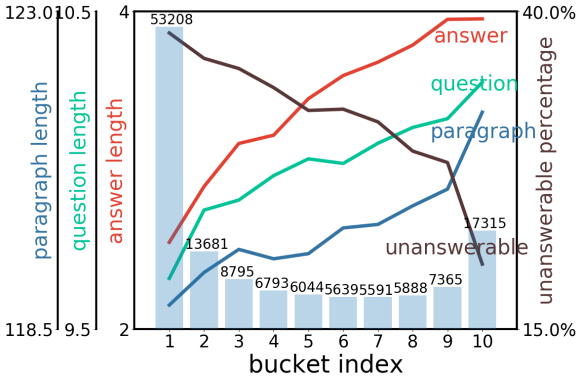

Figure 2: Statistical illustration on different levels of difficulty examples in SQuAD 2.0. Four line respectively indicate average answer length, question length, paragraph length and the proportion of unanswerable examples with respect to level of difficulty. Bar indicates the number of examples in each bucket. Best viewed in color mode.

eral heuristic curriculum design in Table 5. For Difficulty Review methods, we adopt *word rarity*, *answer length*, *question length*, and *paragraph length* as difficulty metrics similar to (Tay et al., 2019; Platanios et al., 2019). We calculate word rarity as the average word frequency of the question, where the frequency is count from all questions in trainset. We define difficult examples as those with lower words frequencies, longer answer, question, and paragraph length. We first sort all examples using these metrics, and divide them evenly to obtain 10 example buckets with a corresponding level of difficulty, and the Curriculum Arrangement strategy remains unchanged as Annealing. For Curriculum Arrangement method, we try *Naive order* for comparison. We directly implement the curriculum as $\{C_i\}$ (instead of $\{S_i\}$) without any sampling algorithm, only that S_{N+1} is still retained for fair comparison. In the meantime, the Difficulty Evaluation method remains unchanged as Cross Review. The results show that these intuitive design indeed works well with various improvements ranging from **+0.12** to **+0.76** on F1 score. But they are all outperformed by our Cross Review + Annealing approach.

Case study: Easy VS Difficult In our Cross-Review method, the dataset was divided into N buckets $\{C_i\}$ with different levels of difficulty. Here we further explore what do these easy/difficult examples in various tasks actually look like. Earlier in the introduction (see Table 1), we have provided a straightforward illustration of easy cases versus hard cases in SST-2 dataset. Among ten different levels of difficulty, these cases are sampled from the most easy bucket (C_1) and the most difficult bucket (C_{10}), respectively. The results are very clear and intuitive.

We further choose SQuAD 2.0 as a more complex task to perform in-depth analysis. Under the $N = 10$ setting, we reveal the statistical distinctions of all buckets $\{C_i\}$ in Fig 2. With three monotonically increasing curve, it is very clear that difficult examples tend to entail longer paragraph, longer questions, and longer answers. Such conclusions conforms to our intuition that longer text usually involves more complex reasoning patterns and context-dependency. And these challenging examples are now successfully excluded in the early stages attributing to our CL approach. Another interesting result is that the percentage of unanswerable examples drops consistently from 40% to 20% along the difficulty axis. We assume that simply doing classification is easier than extracting the exact answer boundaries.

On Different Settings of N One argument that needs to be specified ahead in our approach is N, which decides the number of meta-datasets, learning stages, and also the granularity of our difficulty score. Assume the metric is between 0 and 1, which fits almost all the cases, then the difficulty score c_{ji} should range from 0 (when all teacher models fail) to $N-1$ (when all teacher models succeed), so all examples can be distinguished into N different levels. With N becoming larger, the granularity is also finer.

To examine the impact of different settings, we perform ablation study on SQuAD 2.0 task given a wide range of choices: from 2 to 20 (see Fig 3). It is obvious that under all settings our approach

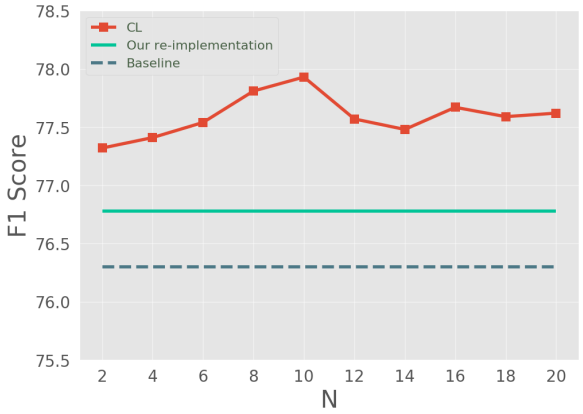

Figure 3: F1 score on SQuAD 2.0 with respect to N. Dotted line is the baseline, solid line is our re-implementation, best viewed in color mode.

outperforms the baseline by at least **+0.5** F1 score (even including $N = 2$, where the difficulty evaluation results may be affected by the fluctuation of single-teacher review). We also experiment with extremely large N value. For $N = 100$, the result is **74.10** on F1 score (**2.68** below our baseline), which is as expected because the meta-dataset is too small to prepare a decent teacher that is capable of evaluating. In general, our approach is very robust with the settings of N.

5 Related Works

The idea of training a neural network in an easy-to-difficult fashion can be traced back to (Elman, 1993). (Krueger and Dayan, 2009) revisited the idea from a cognitive perspective with the *shaping* procedure, in which a teacher decomposes a complete task into sub-components. Based on these works, Curriculum Learning is first proposed in (Bengio et al., 2009). They designed several toy experiments to demonstrate the benefits of curriculum strategy both in image classification and language modeling. They also propose that curriculum can be seen as a sequence of training criteria, and at the end of it, the reweighting of examples should be uniform with the target distribution, which inspired the design of our Curriculum Arrangement algorithm.

Although CL has been successfully applied to many areas in computer vision (Supancic and Ramanan, 2013; Chen and Gupta, 2015; Jiang et al., 2017), it was not introduced to solve NLU tasks until (Sachan and Xing, 2016). By experimenting with several heuristics, they migrated the success of CL (Kumar et al., 2010) to machine reading comprehension tasks. (Sachan and Xing, 2018) further extended this work to question generation. More recently, (Tay et al., 2019) employed CL strategy to solve reading comprehension over long narratives. Apart from them, there aren't very many works that discuss CL in the context of NLU to the best of our knowledge.

On the methodology of designing CL algorithms, our approach is closely related to (Guo et al., 2018; Wang et al., 2019; Platanios et al., 2019; Tay et al., 2019), where a curriculum is formed via two steps: evaluating the difficulty first, then sampling the examples into batches accordingly. For different target tasks, the evaluation methods also vary greatly. (Guo et al., 2018) first examined the examples in their feature space, and define difficulty by the distribution density, which successfully distinguished noisy images. (Wang et al., 2019) incorporated category information into difficulty metric to address imbalanced data classification. In language tasks, (Platanios et al., 2019) and (Tay et al., 2019) propose to consider the length of context as extent of difficulty. Another line of works see curriculum construction as an optimization problem (Kumar et al., 2010; Graves et al., 2017; Fan et al., 2018), which usually involves sophisticated design and is quite different from our approach.

6 Conclusion

In this work we proposed a novel Curriculum Learning approach which does not rely on human heuristics and is simple to implement. With the help of such a curriculum, language models can significantly and universally perform better on a wide range of downstream NLU tasks. In the future, we look forward to extend CL strategy to the pretraining stage, and guide deep models like transformer from a language beginner to a language expert.

Acknowledgments

We thank all anonymous reviewers for their valuable comments. This work is supported by the National Natural Science Foundation of China, Grant No.U19A2057, No.61876223, the National Science Fund for Distinguished Young Scholars No.61525206, the Fundamental Research Funds for the Central Universities, Grant No.WK3480000008, and the grant of Tianjin New Generation Artificial Intelligence Major Program No.19ZXZNGX00110.

References

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48. ACM.

Luisa Bentivogli, Peter Clark, Ido Dagan, and Danilo Giampiccolo. 2009. The fifth pascal recognizing textual entailment challenge. In *TAC*.

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14.

Xinlei Chen and Abhinav Gupta. 2015. Webly supervised learning of convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1431–1439.

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

William B Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

Jeffrey L Elman. 1993. Learning and development in neural networks: The importance of starting small. *Cognition*, 48(1):71–99.

Yang Fan, Fei Tian, Tao Qin, Xiang-Yang Li, and Tie-Yan Liu. 2018. Learning to teach. *arXiv preprint arXiv:1805.03643*.

Adam Fisch, Alon Talmor, Robin Jia, Minjoon Seo, Eunsol Choi, and Danqi Chen. 2019. MRQA 2019 shared task: Evaluating generalization in reading comprehension. In *Proceedings of 2nd Machine Reading for Reading Comprehension (MRQA) Workshop at EMNLP*.

Alex Graves, Marc G Bellemare, Jacob Menick, Remi Munos, and Koray Kavukcuoglu. 2017. Automated curriculum learning for neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1311–1320. JMLR. org.

Sheng Guo, Weilin Huang, Haozhi Zhang, Chenfan Zhuang, Dengke Dong, Matthew R Scott, and Dinglong Huang. 2018. Curriculumnet: Weakly supervised learning from large-scale web images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 135–150.

Guy Hacohen and Daphna Weinshall. 2019. On the power of curriculum learning in training deep networks. *arXiv preprint arXiv:1904.03626*.

Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li, and Li Fei-Fei. 2017. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. *arXiv preprint arXiv:1712.05055*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kai A Krueger and Peter Dayan. 2009. Flexible shaping: How learning in small steps helps. *Cognition*, 110(3):380–394.

M Pawan Kumar, Benjamin Packer, and Daphne Koller. 2010. Self-paced learning for latent variable models. In *Advances in Neural Information Processing Systems*, pages 1189–1197.

Hector Levesque, Ernest Davis, and Leora Morgenstern. 2012. The winograd schema challenge. In *Thirteenth International Conference on the Principles of Knowledge Representation and Reasoning*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Gail B Peterson. 2004. A day of great illumination: Bf skinner's discovery of shaping. *Journal of the experimental analysis of behavior*, 82(3):317–328.

Emmanouil Antonios Platanios, Otilia Stretcu, Graham Neubig, Barnabas Poczos, and Tom M Mitchell. 2019. Competence-based curriculum learning for neural machine translation. *arXiv preprint arXiv:1903.09848*.

Martin Popel and Ondřej Bojar. 2018. Training tips for the transformer model. *The Prague Bulletin of Mathematical Linguistics*, 110(1):43–70.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. *arXiv preprint arXiv:1806.03822*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Mrinmaya Sachan and Eric Xing. 2016. Easy questions first? a case study on curriculum learning for question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 453–463.

Mrinmaya Sachan and Eric Xing. 2018. Self-training for jointly learning to ask and answer questions. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 629–640.

Kornl Csernai Shankar Iyer, Nikhil Dandekar. 2016. Diy corpora: the www and the translator.

Burrhus F Skinner. 1958. Reinforcement today. *American Psychologist*, 13(3):94.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

James S Supancic and Deva Ramanan. 2013. Self-paced learning for long-term tracking. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2379–2386.

Yi Tay, Shuohang Wang, Luu Anh Tuan, Jie Fu, Minh C Phan, Xingdi Yuan, Jinfeng Rao, Siu Cheung Hui, and Aston Zhang. 2019. Simple and effective curriculum pointer-generator networks for reading comprehension over long narratives. *arXiv preprint arXiv:1905.10847*.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2016. Newsqa: A machine comprehension dataset. *arXiv preprint arXiv:1611.09830*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Yiru Wang, Weihao Gan, Jie Yang, Wei Wu, and Junjie Yan. 2019. Dynamic curriculum learning for imbalanced data classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5017–5026.

Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. 2019. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*.