Curriculum Learning in Sentiment Analysis

Proposal

Yuqian Lei & Yunlong Wang

26.01.2022

Abstract

Curriculum learning and anti-curriculum learning investigate the benefits of ordered training, which was inspired by human studying procedures. The examples for training in curriculum learning are ordered by their difficulty levels and are exposed to the model in different ways. In this paper, we will investigate whether curriculum learning could improve the performance of models, more specifically DistilBERT models, compared to standard learning which does not apply curricula. We first calculate the difficulty scores for all the examples in a given dataset based on their loss values. Then, we reorder the dataset with the help of the difficulty scores and quantify the benefits of ordered learning by conducting different experiments on four kinds of learning: curriculum, anti-curriculum, random-curriculum, and standard. Our experiments show that curricula only have marginal benefits when the training time is limited. However, with noisy data added, curricula do improve the performance of a model in general.

1 Introduction

When humans learn, easy principles usually come first. For example, when one learns maths, one first has to learn the meaning of numbers, how to count, then addition, multiplication, and so on. The difficulty level increases as one learns more. Curriculum learning (CL) is inspired by this principle of ordered learning, in which easier examples are exposed to a model first in the training and the difficulty is increased as the model learns more [8]. On the contrary, in anti-curriculum learning, harder examples are fed into a model first and the difficulty gradually decreases. Unlike the previous two methods that apply ordered learning, in random-curriculum learning, the examples are exposed to a model without considering the difficulty levels of the examples, only the size of the dataset gradually grows by time under the pacing function. These three types of learning are also referred to as explicit curricula, in which the learning order can be altered [8].

The difficulty levels of the examples are determined by their loss values, which are calculated by splitting a dataset into four parts, training on three parts and validating on the remaining part, and the difficulty levels are obtained from the validating. The procedure is repeated for four times in order to obtain the loss values for the examples in all four parts of the dataset. The examples are then reordered based on the loss values for either curriculum or anti-curriculum learnings.

1.1 Datasets

The examples that are considered in this paper are 2,500 IMDB film reviews randomly selected from a large IMDB training dataset containing 25,000 reviews. For testing the model, we select 500 reviews randomly from the IMDB testing dataset that also contains 25,000 reviews. Both datasets contain two fields: *text* which contains a string of film review, *label* which contains a value that is either negative(0) or positive(1), showing whether the writer of a review likes a film or not. These datasets are used specifically for binary sentiment analysis, which we will mainly focus on in this paper [5].

1.2 DistilBERT Model

In recent years, a new language representation model called BERT is introduced, which achieved state-of-the-art performance on various natural language processing tasks [2]. Due to the limited computational resources we have, in this paper, we will use DistilBERT model, which is a distilled version of BERT that is 40% smaller but 60% faster than BERT. Despite its reduced size, DistilBERT model can still retain 97% of its language understanding capabilities [6]. More specifically, we will use a DistilBERT base model that is pretrained on raw English texts without any labelling [1], and then fine-tune the pretrained model by training it with our custom IMDB dataset.

1.3 Related Work

The work in this paper is done based on the paper [8], which focuses on investigating the benefits of ordered learning in image classification. The paper shows the results that curriculum learning can improve the performance when the training time is limited or when noise is added to the training. We follow the pacing functions in [8], which determine the number of examples that should be exposed to the model at each training step. In our paper, the pacing functions are applied to curriculum, anti-curriculum and random-curriculum learnings, but not to standard learning.

1.4 Research Question

Since curriculum learning could improve the performance in image classification tasks with limited time or noise [8], we want to explore whether the setup in [8] could also be adopted to a different domain of natural language processing, more specifically to sentiment analysis, and still achieve the same or a similar result. Explicitly, we will investigate the benefits of the three types of learnings in which the order is altered and to which a pacing function is applied, by comparing their results with the results yielded from standard learning that uses neither ordered learning nor any pacing functions.

2 Model Architecture

Knowledge distillation is a technique used to compress a large model (the teacher) into a smaller model (the student). The goal of training the student model is to mimic the behaviour of the teacher model [3]. In our work, the student is DistilBERT model and the teacher is BERT model. BERT stands for Bidirectional Encoder Representations from Transformers, which is built upon twelve transformer encoders in the base case [2]. DistilBERT model has almost the same architecture as BERT, except that in DistilBERT model, the token embeddings and the pooler layer are removed, and the number of layers (transformer encoders) is divided by two, which is six in the base case [6].

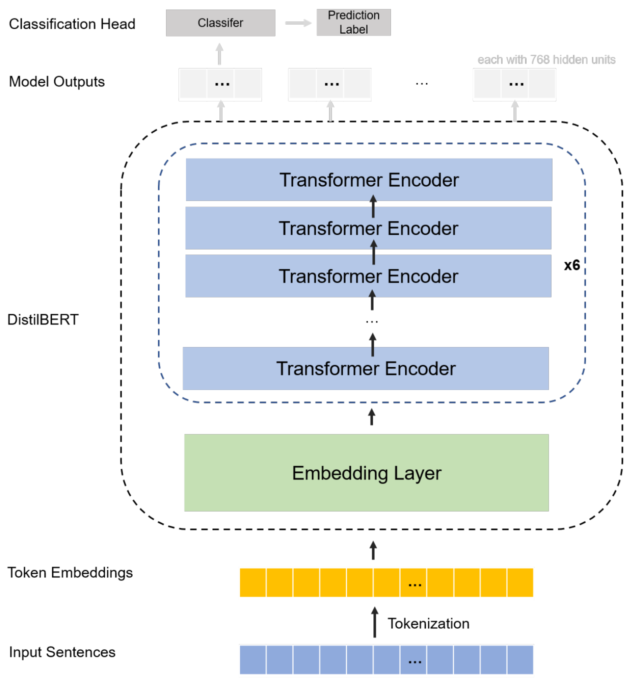

Figure 1: DistilBERT model structure

2.1 Embedding Layers

Similar to BERT, DistilBERT takes input embeddings as inputs, which are the sum of segment embeddings and position embeddings. As shown in Figure 2, Each embedding corresponds to a token, which is obtained by tokenizing the original input text into a sequence of tokens. After tokenization, a special token [CLS] is added to the beginning of each sequence, which stands for the task of classification, and another special token [SEP] is used to separate sentences. Segment embeddings are used to indicate which sentence a token belongs to, while position embeddings are used to locate a token in the whole sequence. [2]

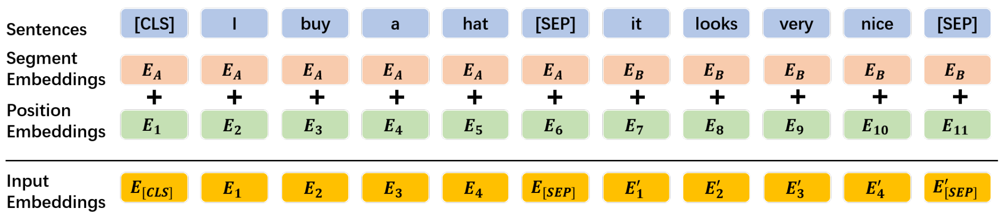
Sentences
[CLS]
I
buy
a
hat
[SEP]
it
looks
very
nice
[SEP]
Segment Embeddings
E_A
E_B
Position Embeddings
E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8 E_9 E_{10} E_{11}
Input Embeddings
$E_{[CLS]}$ E_1 E_2 E_3 E_4 $E_{[SEP]}$ E'_1 E'_2 E'_3 E'_4 $E'_{[SEP]}$

Figure 2: DistilBERT input representation

2.2 Transformer Encoder

As shown in Figure 1, DistilBERT is composed of six transformer encoders in the base case, when the teacher model BERT is consisted of twelve transformer encoders. The architecture of a transformer encoder is mainly based on an attention mechanism used for natural language understanding, which was introduced in [7]. As explained in the paper, an encoder is one part of a transformer that is consisted of both an encoder and a decoder. Since DistilBERT only uses encoders, we will omit the explanation for the decoder. The two main components in the structure of an encoder are the self-attention layer and the feed forward network. In simple terms, self-attention is used to find how much a token is associated with all the tokens in a sequence, including itself.

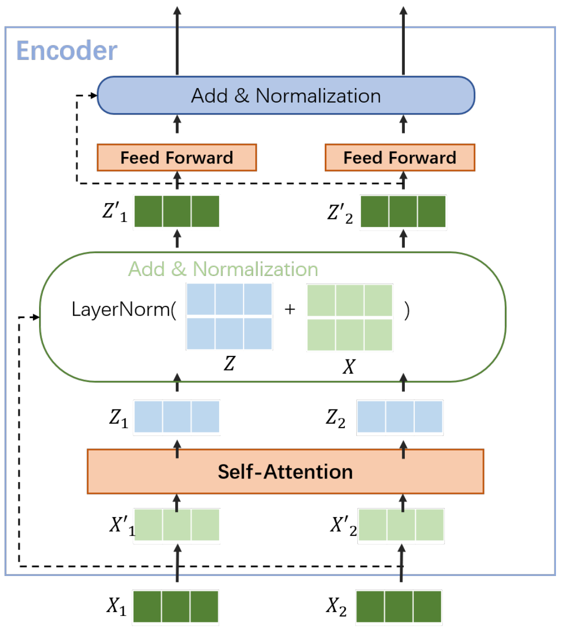

Figure 3: **General architecture for transformer encoder in base DistilBERT.** The main components of an encoder are the self-attention layer and the feed forward network.

To calculate the value of self-attention, three weight matrices W_q, W_k, W_v and an embedding matrix X are used. Each row in X corresponds to a token in the input sequence. Firstly, we calculate the three functional matrices Q, K, V, which stand for queries, keys and values, by multiplying the embedding matrix with its corresponding weight matrix [7]:

$$Q = XW_q, K = XW_k, V = XW_v \tag{1}$$

Then, the following formula is used to calculate self-attention, where d_k denotes the dimensions of W_k [7]:

$$Z = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{2}$$

To calculate the result of the feed forward network, we simply apply the *relu* activation function and linear transformation to the output of the self-attention layer Z [7]:

$$FFN(Z) = max(0, ZW_1 + b_1)W_2 + b_2 \tag{3}$$

2.3 Pre-training DistilBERT: Masked LM

The outputs of DistilBERT are the hidden vectors for each token, which are used for pretraining the model and downstream tasks. In this paper, the hidden vector corresponding to the first token [CLS] is used for sentiment classification. Many traditional models predict a word in a sequence in one direction, which means that they either predict the model based on the context before the word or after the word. This kind of directional approach adds a limitation when the model is learning. To overcome this challenge and to train a bidirectional model, DistilBERT uses a strategy called Masked LM. DistilBERT is pretrained on the datasets *English Wikipedia* and *Toronto Book* [6] by choosing 15% of the tokens in a sequence randomly, replacing the chosen token with a [MASK] token 80% of the time, with a random token 10% of the time, and doing nothing 10% of the time. [2]

3 Approach

In our paper, curricula are implemented very similar to how they are implemented in this paper [8]. A curriculum is defined by three components, which are the scoring function, the pacing function and the order. The scoring function is used to determine the difficulty of an example, the higher the score, the more difficult the example, and vice versa. The pacing function is used to specify the size of the dataset during training at each step. The order can be curriculum, anti-curriculum, or random.

3.1 Scoring Function

A scoring function is defined as $s(x, y) \in \mathbb{R}$, where x is an input example and y is the label of the example. An example x_i is more difficult than an example x_j if $s(x_i, y_i) > s(x_j, y_j)$.

The original work [8] considered three different scoring functions, but in our work we only consider loss based c-score. C-score [4] is designed to determine the consistency of a model by letting the model predict the label of an example after training it on the rest of the dataset that does not contain the example. In our work, the loss based c-score for each example is calculated through the average of 4 runs, and each run contains four set. The training dataset is first split into four subsets, and then the first three subsets are used for training in the first step. After the training, the model predicts the labels of the examples in the last subset and the loss is calculated. In the second step, the next three subsets are selected for training and the first one for validating, the same repeats for the last two steps until all four subsets are validated.

3.2 Pacing Functions

The six pacing functions we choose are the same as the ones used in [8], except that the parameters a and b we use are different. a determines at which percentage of total steps should all the training data be exposed to the model, and b denotes the percentage of training data used at the beginning of the training. The formulas for the pacing functions are shown in Table 1. N denotes the total number of examples, T denotes the total number of steps, and t denotes the current step. We select $a \in \{0.1, 0.4, 0.8, 1.6\}$ and $b \in \{0.1, 0.4, 0.8\}$ for the experiments.

Name	Pacing Function
linear	$Nb + \frac{N(1-b)}{aT}t$
quad	$Nb + \frac{N(1-b)}{(aT)^2}t^2$
root	$Nb + \frac{N(1-b)}{\sqrt{aT}}\sqrt{t}$
step	$Nb + N\left\lceil \frac{t}{aT} \right\rceil$
exponential	$Nb + \frac{N(1-b)}{e^{10}-1}(exp(\frac{10t}{aT} - 1))$
logarithm	$Nb + N(1-b)(1 + 0.1log(\frac{t}{aT} + e^{-10}))$

Table 1: Six pacing functions used throughout our experiments.

3.3 Experimental Setup

For the exploration, we only investigate the benefits of fine-tuning stage of pretrained model [1] in task sentiment analysis. For the sake of comparison, we divided our experiments into two parts. One for standard learning, which does not use any pacing function but does use a random ordering. The other one is for curriculum learning, in which all the six pacing functions and three orderings are applied. To investigate the benefits of curricula in the noisy environment, we add noise with 40% noise(by confusing label randomly) to create new datasets. Table 2 shows the important hyperparameters we used in our experiments.

Hyperparameter	Standard Learning	Curriculum Learning
learning rate	1e-2, 1e-3	1e-2
optimizer	sgd, adam	sgd
scheduler	cosine, exponential	cosine
weight delay	1e-4, 1e-5, 1e-6, 0	1e-4
batch size	5, 10	10
momentum	0.9	0.9
ordering	-	curriculum, anti-curriculum, random-curriculum
pacing function	-	linear, quad, root, step, exponential, logarithm
pacing-a	-	0.1, 0.4, 0.8, 1.6
pacing-b	-	0.1, 0.4, 0.8

Table 2: **Hyperparameter selections for standard learning and curriculum learning.** An optimizer is an optimization algorithm for minimizing of loss value. Learning rate is a parameter in the optimization algorithm, which determines the step size for each iteration of optimization. A scheduler, also known as a learning rate scheduler, adjusts the value of learning rate for each iteration. The hyperparameter Momentum is used to keep the direction of optimization and prevent oscillations. Batch size denotes the size of data for forward computing at each iteration. For faster training with the noise-dataset, we give up pacing-a 1.6, pacing functions quad and root.

4 Results

With standard dataset and standard training Time, we can see that in Figure 4, curriculum learning, anti-curriculum learning and random-curriculum learning do not have any benefits, but rather have a negative impact on the results on the fine-tuning stage.

With limited training time, as shown in Figure 5, only random-curriculum has a marginal benefit when the training time is limited to 402 steps.

Figure 6 shows the results with noisy data and standard training time. Similar to Figure 4, it does not show that curricula have any benefits compared to the standard training. However, Table 3 shows that curriculum learning does have remarkable performance in general, since it has much better average scores and smaller standard deviation than standard learning. Curriculum learning, which uses easy examples first, achieved the best result with the average score 0.8048 and standard deviation 0.029.

5 Discussion

The results in [2] show that curricula could improve the performance of a model in terms of accuracy when the training time is limited or when noisy data is added, more

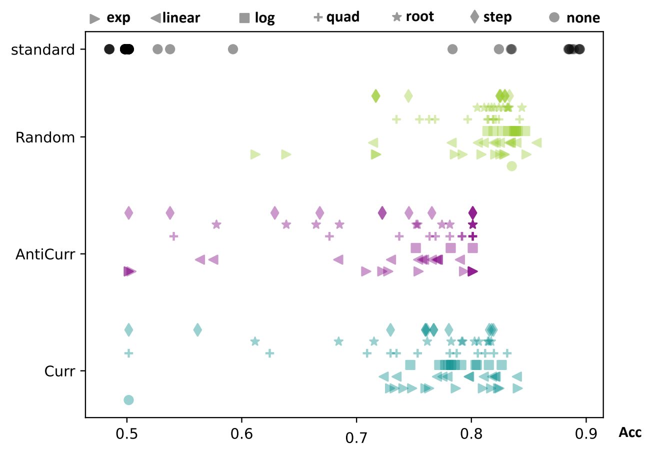

Figure 4: **Test accuracy with standard dataset and standard training time.** Different shapes of points are the results obtained using different pacing functions. Different points with the same shape are the results obtained using different hyperparameters. Y-axis indicates different learning forms, X-axis indicates test accuracy.

Learning	Best	Average	Standard Deviation
Curriculum Learning	0.8320	**0.8048**	**0.0290**
Random-curriculum Learning	0.8300	0.7787	0.1044
Anti-curriculum Learning	0.8340	0.7445	0.0499
Standard Learning	**0.8760**	0.5755	0.1474

Table 3: **Statistical test accuracy of all training with noisy dataset (40% noise).** This statistical result is calculated using the best accuracy of each training.

specifically, when the model is dealing with the image classification task. Our results were obtained after training DistilBERT with the IMDB dataset, which is a completely different task. Nevertheless, the results we got did have some similarity compared to the results shown in the paper. Firstly, both results show that curricula do not have any benefits under the condition of standard training time, since they do not improve the accuracy of the model. Secondly, when the training time is limited, the accuracy for curriculum is largely improved as shown in the paper. We got a different result compared to this result in the paper. Instead of having a huge accuracy improvement, we only got a marginal benefit for random-curriculum. Lastly, the test accuracy is improved when data with noisy labels are added in the paper. Although we do not see a better performance when we only look at the best accuracy scores, we do see that curricula performed much better in terms of the average accuracy scores and standard deviation.

Curriculum learning, in which the examples are ordered by showing easier examples first, does help stabilize the optimization process. One possible reason is that the learn-

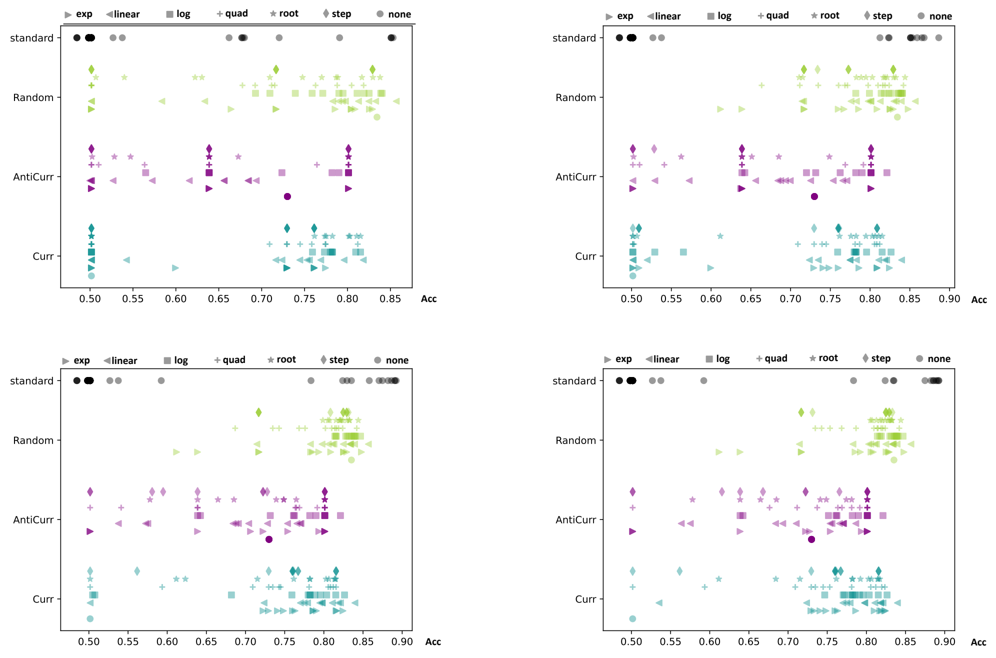

Figure 5: **Test accuracy with standard dataset and limited training time.** Top left with limited step 402, top right with limited step 1005, bottom left with limited step 2010, bottom right with limited step 3015.

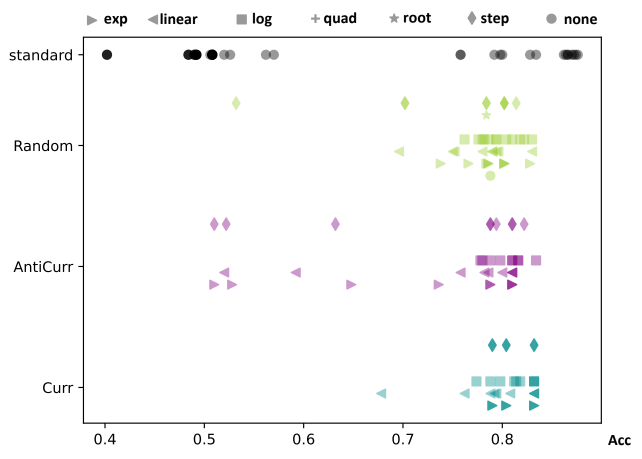

Figure 6: **Test accuracy with noisy data added (40% noise) and standard training time.**

ing rate is the largest at the beginning, and the easier examples make it easier for the model to learn and there is a higher probability that the predicted label would be right. Although in the end, the harder examples do hinder the optimization process of our model, they did not have a big impact on the model since the learning rate is very small.

6 Conclusion

In this paper, we explored the benefits of three types of curricula over standard learning by running an extensive amount of experiments on DistilBERT using our custom IMDB dataset. The results of the experiments show that curricula are not helpful to find the best result when the standard dataset is used and the standard training time is applied. The results do show that curricula have a remarkable performance in terms of the average scores and standard deviation of the accuracy scores. We did obtain some similar results compared to the results in [2] under the settings of standard learning and added noisy data. And we could conclude that we successfully adopted the general setup in the paper to sentiment analysis.

References

[1] Huggingface: DistilBERT base uncased. https://huggingface.co/distilbert-base-uncased.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[3] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015.

[4] Ziheng Jiang, Chiyuan Zhang, Kunal Talwar, and Michael C. Mozer. Exploring the memorization-generalization continuum in deep learning. *CoRR*, abs/2002.03206, 2020.

[5] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[6] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR*, abs/1910.01108, 2019.

[7] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[8] Xiaoxia Wu, Ethan Dyer, and Behnam Neyshabur. When do curricula work? *CoRR*, abs/2012.03107, 2020.