Curriculum-Based Ensembling for Sentiment Classification

Christian Cadisch, Michael Hodel, Johannes Weidenfeller, Lucas Weitzendorf
Group Name: Tranformers go brrrrrr
Department of Computer Science, ETH Zurich, Switzerland

***Abstract*—In recent history, the NLP landscape has been defined by transformer models such as BERT. Fine-tuning these models has led to state-of-the-art results in a wide range of language processing tasks. In this project, we investigate an ensembling approach based on training component models on different learning curricula, specifically, varying training data selection and ordering. We find that ensemble models with different learning curricula achieve marginally better performance than any individual model.**

I. Introduction

Text sentiment classification is a common task in the NLP landscape. Until recently, established methods such as recurrent neural networks (RNNs) [1] or deep convolutional neural networks (CNNs) [2] [3] were the state of the art. However, with the introduction of pre-trained transformer-based masked language models such as BERT [4], the field has shifted. Since the rise of BERT, fine-tuning large models on downstream tasks has consistently outperformed other approaches in many NLP tasks [5]. In this project, the goal was the classification of English tweets according to sentiment. Tweets differ from other texts through their informal grammar, briefness, and other characteristics such as frequent use of emoticons and hashtags. Previous research in this area has led to BERTweet [6], a version of BERT pre-trained on a large corpus of English tweets. Given past research with pre-trained models, it was a natural choice to build on BERTweet and fine-tune it for the task at hand.
Ensembling and curriculum learning are two strategies commonly employed to boost model performance [7]. We hypothesize that combining these two approaches could lead to increased model performance. Specifically, we assume that ensembling can be improved by varying the training dataset and order from component model to component model, as opposed to only for example varying the random seed used for shuffling the training data or different initial model weights. To that end, we compare the performance of an ensemble consisting of models trained with different learning curricula to an ensemble of models varying only in the randomness of the order of their training data.

II. Data

A. Dataset

The given dataset consists of 2.5M tweets labeled either as positive or negative, with 1.25M tweets per sentiment. The tweets were labeled through an automated process using the following heuristic: Tweets containing a happy emoticon were categorized as positive, while tweets with a sad emoticon were categorized as negative. We discuss some shortcomings of this labeling technique in subsection II-B. In addition, user mentions and links are replaced with `<user>` and `<url>` tags, respectively.

B. Exploratory Data Analysis

We perform some exploratory data analysis to obtain a rudimentary understanding of the data and potentially improve performance via explicit prior knowledge. To begin with, we find that roughly 9.25% of tweets are duplicates. The main analysis focuses on computing frequencies of various properties of tweets such as count, length, and occurrences of special tokens such as tags and hashtags. We find that certain tokens are strong predictors for one sentiment class or another and that especially tokens predicting negative sentiment do not necessarily have an inherently negative connotation (e.g. 'paperback' or 'frame', see table III in the appendix).

Additionally, we group tweets based on their length and consider the ratio of positive to negative sentiments within each group. We find that over 85% of tweets with lengths between 120 and 129 characters are labeled as negative. These findings can be explained by considering the method in which the dataset was labeled. Upon further investigation, we find that many tweets labeled as negative include a product title and an incomplete product description, followed by a short link.

These types of tweets typically contain parentheses with some qualifier (e.g. 'paperback' in the case of books), followed by a colon, i.e. the character sequence '):'. They are therefore assigned a negative sentiment, although the occurrence of '):' is not in the context of an emoticon. This very likely indicates a shortcoming in the labeling technique used to create the dataset. Generally, we think the automatic labeling process is flawed, as it is based on simply looking at the occurrence of emoticons that do not necessarily represent true sentiment, as well as the underlying assumption of clear separability into two disjoint sentiments.

In a more thorough analysis, we investigate how often tweets occur that include an open bracket and end in a link, preceded by an incomplete description, that is tweets that end in the character sequence '`... <url>`'. We will refer to these kinds of tweets as spam tweets. About 15% of all tweets in the training set are spam tweets, of which over 99.9% are classified as negative. The share of spam tweets in the holdout set is slightly smaller, at 13%. The large number of tweets of this form also explains the predominantly negative classification of tweets in the length range 120-129, since this corresponds to the maximal tweet length of 140 after replacing a short link with the tag `<url>`. We find that 95.4% of spam tweets lie in the length range of 120-129. Finally, we observe that a model as specified in section III and trained on the full data set already classifies almost all spam tweets in the test set correctly, achieving an accuracy of $> 99.9\%$ on the subset of spam tweets. This indicates that the model can reliably learn the appearance of a spam tweet.

III. Models

A. Architecture

We use the pre-trained language model BERTweet [6] as a base model to obtain tweet embeddings for subsequent classification with a dense classification head. BERTweet is a multi-layer bidirectional transformer, as introduced by Vaswani et al. [8] and shares the same architecture as the model $\text{BERT}_{\text{base}}$ [4]. In particular, it consists of 12 transformer blocks with 12 self-attention heads and a hidden state size of 768 each. It was trained on a dataset consisting of 850M tweets using the RoBERTa [9] pre-training procedure. This procedure mainly differentiates itself from the BERT training by allowing for a longer pre-training with more data, using larger batch sizes with longer sentences and a larger token vocabulary. BERTweet uses FastBPE [10] for tokenization and the vocabulary of the trained tokenizer includes 64K tokens. The classification head consists of a single linear layer on top of the pooled output of the base model.
We follow the training procedure for fine-tuning as outlined in [5] and [4], specifically, training on batches of size 16, applying dropout with a probability of 10%, and using the Adam optimizer [11] with decoupled weight decay [12], a learning rate schedule linearly increasing the learning rate from 0 to 2×10^{-5} in the first 10% of iterations and then linearly decreasing it to 0 in the remaining iterations. One adaptation we make is training the model for only a single epoch, a decision largely motivated by the size of the dataset and additional computational cost and empirically confirmed by not obtaining better results when training for more epochs.

B. Sensitivity Analysis

To investigate the effect of hyperparameter choice on the model performance, we conduct a sensitivity analysis, varying each hyperparameter individually and monitoring the change in classification accuracy. The parameters investigated in this manner are the 5 numerical parameters deemed most relevant, namely the warm-up ratio, maximal learning rate, weight decay, dropout probability, and batch size. Starting from the hyperparameters described above, we multiply each parameter with factors of 2 and 0.5, respectively, while keeping all other parameters fixed. For each of the resulting $2 \times 5 + 1 = 11$ parametrizations, we train the model on 7.6% of the training data and evaluate it on 10K test examples. The results are summarized in table I. We take the results as an indication that the model performance is not particularly sensitive to the precise choice of hyperparameters, since, among the 11 models, the difference in classification accuracy between the best (using a warm-up ratio of 0.2) and the worst-performing model (using a batch size of 8) is only 0.37%. In addition, the default hyperparameters are already somewhat optimal, since they yield a score less than 0.1% below the best one. For comparison, a change in the seed for the data order randomization leads to a maximal difference in accuracy of 0.32% over 18 trained models.

Table I
Test accuracies of models trained with different hyperparameters. Values that exceed the accuracy of the default parametrization of 90.69% are marked in bold.

	Lower		Higher	
Hyperparameter	Value	Accuracy	Value	Accuracy
batch size	8	90.40%	**32**	**90.71%**
dropout	0.05	90.69%	0.2	90.69%
max. learning rate	1e-05	90.59%	4e-05	90.50%
warm-up ratio	0.05	90.54%	**0.2**	**90.77%**
weight decay	**0.005**	**90.76%**	0.02	90.62%

It is important to note that this analysis has at least three shortcomings. First, parameters were only investigated separately from one another, effectively ignoring effects of their interdependence. Second, the hyperparameter analysis could potentially yield significantly different results when using more of the training data, thus the findings do not necessarily translate to the setting of using the full training data. Third, we trained a model only once for each configuration, making it difficult to separate the variance of the model performance arising from the stochasticity of the training process from the effects of hyperparameter choice. Nevertheless, since the initial results described above did not indicate strong improvements based on varying the hyperparameters, and also due to the required computational resources, a more extensive analysis, for instance, a grid search based approach, was not pursued. For the training of the final models, the default hyperparameter values recommended in [5] and [4] as listed above were therefore used.

C. Preprocessing

The authors of [6] perform the following preprocessing steps

1) Replace user mentions and URLs with corresponding special tokens `@USER` and `HTTPURL`, respectively,

2) Convert emoticon tokens into descriptive text using the `emoji` package.
3) Normalize punctuation marks like ellipsis points and apostrophes.

In addition to these preprocessing steps, we also drop duplicate tweets, as they contain no additional information, as well as all spam tweets, since we assume that they pollute the dataset. We classify all spam tweets appearing in the test set as negative.

D. Baseline Models

In addition to the model described above, we also implement two baseline models for comparison.
As a first baseline, we implement a classifier based on the frequencies of n-tuples of consecutive words. During training, the model computes the per-sentiment frequency of each n-tuple of consecutive words, where the length of tuples n is a parameter. Let T be a tweet and denote by $\mathcal{X}(T)$ the set of consecutive n-tuples contained in T. At inference time, the model computes scores s_+ and s_- for the two sentiment classes as follows

$$s_\pm(T) = \sum_{x \in \mathcal{X}(T)} n_\pm(x)^p$$

where $n_\pm(x)$ denotes the frequency of the n-tuple x in the set of negative/positive training examples, respectively, and $p \in [0, 1]$ is a parameter intended to discount the contribution of overall very frequent tuples to the score. The model predicts the sentiment with a higher score and a negative sentiment if the scores are tied or in cases where $s_+ + s_- = 0$. Using a grid search approach, we find that values of $n = 2$ and $p = 0.25$ seem to work well. Interestingly, using tuples instead of single words worked better. This model trained on 190K tweets with $n = 2$ and $p = 0.25$ and evaluated on 10K tweets resulted in a classification accuracy of 80.98%, slightly exceeding the grade 4 baseline of 80.42%.
As a second baseline, we use the `catboost` module [13] to train a gradient boosting model on the embeddings of BERTweet. Using the same training and testing data as with the first baseline and the default model parametrization as provided by the library for this approach - except for using early stopping based on the performance on 10% of the data used for validation - results in an accuracy of 87.31%.

IV. Ensembling Candidate Search

A. Ensembling

We try to further boost the performance of our predictions using an ensemble of classifiers. A straightforward way of creating different models is to vary the seeds that determine the data order and the model weight initialization. However, we hypothesize that we can increase the effect of ensembling by training component models not only using varying seeds but also carefully crafting curricula with varying data and data order. We theorize that each model has different weaknesses and that combining them in an ensemble could reduce the intersection of their misclassifications, hence improving on the benchmark ensemble of only varying the seed. To determine suitable ensemble component models, we decide to train a set of candidate models and save their respective predictions for a subsequent ensemble candidate search.

B. Candidate selection

Curriculum learning is a strategy based on the idea to train a model on examples in a more meaningful order, that is, with increasing difficulty. It has been proven to increase accuracy in various natural language understanding (NLU) tasks [7] as well as speed up the convergence of BERT-based models [14].

We select 18 ensemble candidates, only varying in the data selection and order:

- 1 curriculum generated by the default seed that determines the random data order.
- 6 curricula generated by duplicating the $\{$easiest, hardest$\}$ $\times$ $\{10\%, 25\%, 50\%\}$ of the data and randomly inserting it in the original data.
- 6 curricula generated by using the $\{$easiest, hardest$\}$ $\times$ $\{80\%, 90\%, 95\%\}$ of the data exclusively.
- 5 curricula generated by ordering the data by increasing difficulty and subsequently randomly swapping $\{0\%, 10\%, 20\%, 30\%, 40\%\}$ of the ordered examples.

As a measure of the difficulty of a tweet, we use the following approximation: First, we train a proxy model using the BERTweet configuration described in section III-D on a subset of only 10K tweets. Using this model, we then infer the difficulty as the absolute deviation of the predicted class probability from the true label (0 or 1).

For the curricula based on a strict ordering of the examples by increasing hardness, a clustering of classes occurs. That is, among the 20% of the most difficult data, most belong to the same class, and among the easiest examples, the other class dominates. This was deemed an issue, as a model trained on such a curriculum is expected to not only move in the wrong direction at the start of the training but also unlearn some of what it would have learned towards the end. To circumvent this problem, we reorder the data such that within-class order is preserved, but the overall relative class frequencies are preserved as much as possible in any continuous subset of the curriculum. Finally, we reintroduce a small amount of randomness to the data by swapping a certain percentage of samples to prevent too strict of an order or too regular of a pattern in the labels sequence.

C. Candidate search

We train each of the 18 models using the training curricula generation schemes described above on the same 180K tweets (before duplication or filtering). To keep computational costs at reasonable levels, we decide against training more candidate models or using larger data. For each of those models, the predicted probabilities on a set of 10K tweets were saved for the candidate search. The ensemble candidate search considers each of the ensemble sizes $k \in \{2, 3, 4, 5\}$ (limited to 5 due to diminishing returns of ensembling and exponential computational complexity of exhaustive search procedure) and each of the following three aggregation methods. Let p_E denote the probability of positive sentiment as predicted by the ensemble and p_i the probability of positive sentiment as predicted by the i-th ensemble component model.

- Mode of Classes: $p_E = \lfloor \frac{1}{k} \sum_{i=1}^{k} \lfloor p_i \rceil \rceil$
- Arithmetic Mean of Probability: $p_E = \frac{1}{k} \sum_{i=1}^{k} p_i$
- Geometric Mean of Odds: $p_E = \frac{\left(\prod_{i=1}^{k} \frac{p_i}{1-p_i}\right)^{\frac{1}{k}}}{1+\left(\prod_{i=1}^{k} \frac{p_i}{1-p_i}\right)^{\frac{1}{k}}}$

For each ensemble size k and each of the three aggregation schemes described above, we find an ensemble candidate as follows.

1) Rank all possible ensembles by their classification accuracy on the validation set of 10K tweets.
2) Add the $k-1$ candidate models appearing most often in the top 16% of ensembles.
3) Out of the remaining candidate models, add the model appearing most often alongside any $k-2$ of the already added candidate models in the top 16% of ensembles as the last model.

In this manner, we obtain an ensemble of size k for each combination of values for k and possible aggregation methods. We use this more involved approach to determine the top ensemble candidates, as opposed to simply picking the ensemble ranking highest in classification accuracy, to alleviate the possibility of selecting an ensemble that ranks highly due to stochasticity (i.e. to make it less prone to overfitting on the ensemble selection). Note that the 16% cutoff was chosen intuitively as we do not believe the exact percentage to be a significant factor in the final ensemble performance. Finally, the ensemble with the highest classification accuracy is chosen from the 12 ensemble candidates for all possible values of k and all possible aggregation methods as the overall best model.

D. Results

As a benchmark, we also train the same number of models on the same data where only the seed determining the data order has been varied. We find that the performance of the curriculum and the baseline ensembling methods deviates at most 0.24% in classification accuracy from the best ensemble models for any value of k and aggregation method. A comparison of the classification accuracies for the best ensembles sorted by the value of k and aggregation method can be found in the appendix.

Based on the candidate search described in subsection IV-C, we choose the following ensemble, with the component models (A) ordered by hardness with 20% shuffling, (B) duplicate easiest 50%, (C) duplicate hardest 10%, (D) filter out easiest 20%, (E) duplicate hardest 25%. We opt for the arithmetic mean as the aggregation scheme.

For each of those five best components, we train a model on the entire dataset of $2.48M$ tweets, again using 10K tweets for the proxy model and 10K tweets for testing. Repeating the candidate search procedure on these 5 models trained on the full dataset yields the ensemble composed of models (A), (B), and (D) to be the best one, scoring an accuracy of 92.22% on the internal test set. This is also the ensemble we choose as our final submission, despite it only ranking second on the leaderboard (behind (C), which achieves a roughly 0.2% higher accuracy), mainly because it performs best on the larger and thus more representative internal test set. A summary of the individual component model performances, the best ensemble, and the full ensemble (i.e. the one using all 5 models) is given in table II.

Table II
CLASSIFICATION ACCURACY OF FINAL ENSEMBLE MODELS AND THEIR INDIVIDUAL COMPONENT MODELS

	Internal	Leaderboard
(A)	91.88%	90.92%
(B)	91.85%	91.30%
(C)	91.67%	**91.68**%
(D)	91.78%	91.16%
(E)	91.66%	91.06%
Best-3	**92.22**%	91.48%
Full-5	92.02%	91.40%

V. SUMMARY

In this report, we described a curriculum-based BERT [6] fine-tuning approach for English tweet sentiment classification. Perhaps unsurprisingly, we were able to show that a carefully constructed ensemble model can achieve a binary classification accuracy of up to 92.22% on the testing data, outperforming any of the individual models. While we aimed to be as thorough as possible in our experiments, some areas remain unexplored. Future work could further refine the training and ensembling process. In particular, a more thorough hyperparameter search might yield superior classification performance. Additionally, the present approach does not provide any guarantees against overfitting, both in training individual models as well as ensemble component selection. Early stopping constitutes a potential improvement in this area. Further marginal improvements might be made by varying the seed for the initialization of the classification head weights in the search procedure.

APPENDIX

Table III shows the tokens with more than 25K occurrences and the highest ratio of occurrences in negative tweets to overall occurrences.

Table III
TOP 5 TOKENS WITH MORE THAN 25K OCCURRENCES BY RATIO OF OCCURRENCES IN NEGATIVE TWEETS TO OVERALL OCCURRENCES

token	number of occurrences	pos. sentiment ratio
paperback	35501	0.0003
frame	85816	0.0012
wide	23122	0.0145
pack	27669	0.0236
complete	27098	0.0311
(	492390	0.0656

Table IV shows the classification accuracy of all 18 individual component models described in subsection IV-B after training on 180K tweets and evaluating the performance on a test set of 10K tweets. Models included in the final ensemble are marked in bold. The individual models achieve an average classification accuracy of 90.384%. This is about 0.24% lower than the average classification accuracy of the 18 baseline models that only differed in the choice of the seed determining the data order and achieved an average classification accuracy of 90.625%.

Table IV
CLASSIFICATION ACCURACY OF INDIVIDUAL COMPONENT MODELS

Model	Classification accuracy
Increasing difficulty, 20% random swapping	90.742%
Increasing difficulty, no random swapping	90.669%
Easiest 25% duplicated	90.597%
Baseline model (no adaptations)	90.577%
Hardest 80 %	90.577%
Increasing difficulty, 30% random swapping	90.577%
Hardest 50% duplicated	90.494%
Easiest 10% duplicated	90.484%
Hardest 25% duplicated	90.484%
Hardest 95 %	90.484%
Hardest 90 %	90.484%
Increasing difficulty, 10% random swapping	90.433%
Increasing difficulty, 40% random swapping	90.422%
Easiest 50% duplicated	90.412%
Hardest 10% duplicated	90.299%
Easiest 95 %	90.237%
Easiest 90 %	89.825%
Easiest 80 %	89.114%

Tables V and VI show the best candidate of ensembles for each choice of ensemble size (row) and aggregation method (column) as described in subsection IV-C.

Table V
BEST CURRICULUM ENSEMBLES

	Mode	Arithmetic Mean	Geometric Mean
2	90.74%	90.98%	90.98%
3	90.90%	90.88%	90.85%
4	90.84%	90.88%	90.99%
5	90.90%	90.99%	90.90%

Table VI
BEST BASELINE ENSEMBLES

	Mode	Arithmetic Mean	Geometric Mean
2	90.70%	90.99%	90.99%
3	90.93%	91.05%	90.98%
4	91.02%	90.94%	90.98%
5	91.13%	91.01%	91.02%

REFERENCES

[1] K. Filippos and P. Alexandros, "Structural attention neural networks for improved sentiment analysis," *Proc. 15th Conf. Eur. Chapter Assoc. Comput. Linguistics*.

[2] C. A. S. Holge, B. Loic and L. Yann, "Very deep convolutional networks for text classification," *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*.

[3] R.JohnsonandT.Zhang, "Deep pyramid convolutional neural networks for text categorization," *Proc. 55th Annu. Meeting Assoc. Comput. Linguistics*.

[4] J. Devlin, M.-W. Chang, K. Lee, and K. N. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2018. [Online]. Available: https://arxiv.org/abs/1810.04805

[5] M. Mosbach, M. Andriushchenko, and D. Klakow, "On the stability of fine-tuning {bert}: Misconceptions, explanations, and strong baselines," in *International Conference on Learning Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=nzpLWnVAyah

[6] D. Q. Nguyen, T. Vu, and A. Tuan Nguyen, "BERTweet: A pre-trained language model for English tweets," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online: Association for Computational Linguistics, 2020, pp. 9–14. [Online]. Available: https://aclanthology.org/2020.emnlp-demos.2

[7] B. Xu, L. Zhang, Z. Mao, Q. Wang, H. Xie, and Y. Zhang, "Curriculum learning for natural language understanding," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, Jul. 2020, pp. 6095–6104. [Online]. Available: https://aclanthology.org/2020.acl-main.542

[8] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[9] N. G. J. D. M. J. D. C. O. L. M. L. L. Z. V. S. Yinhan Liu, Myle Ott, "Roberta: A robustly optimized bert pretraining approach."

[10] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 1715–1725. [Online]. Available: https://aclanthology.org/P16-1162

[11] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *ICLR (Poster)*, 2015. [Online]. Available: http://arxiv.org/abs/1412.6980

[12] I. Loshchilov and F. Hutter, “Decoupled weight decay regularization,” in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=Bkg6RiCqY7

[13] “Catboost: gradient boosting with categorical features support.”

[14] K. Nagatsuka, C. Broni-Bediako, and M. Atsumi, “Pre-training a BERT with curriculum learning by increasing block-size of input text,” in *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2021)*. Held Online: INCOMA Ltd., Sep. 2021, pp. 989–996. [Online]. Available: https://aclanthology.org/2021.ranlp-1.112