@BERTweet and friends analyzing sentiments #investigation

Sinan Demirci, Claudio Ferrari, Jérémy Scheurer, Vanessa Tschichold
Team: FrontRowCrew
Department of Computer Science, ETH Zurich, Switzerland

***Abstract*—In this work we perform sentiment analysis on Twitter data. This has become especially relevant due to the growing popularity and importance of social media. Our model leverages BERTweet [1], a transformer-based language model pre-trained on Twitter data, an additional dataset, as well as ensemble learning to achieve 91.9% accuracy on test data. Furthermore, we draw attention to a possible source of information leakage in the provided dataset.**

I. Introduction

Over the last decade, the popularity of social media platforms, in particular microblogging, has increased substantially [2]. With more people expressing themselves on such platforms, the interest in extracting information from this data has grown as well. This has led sentiment analysis to be one of the fastest growing research fields in computer science [3]. Generally, text sentiment analysis tackles the problem of classifying a piece of text into predefined sentiment classes. In this work, we specifically focus on classifying tweets into the classes positive and negative.

Our contributions are summarized as follows:

- We show that BERTweet [1] improves the accuracy by about 1.5% compared to BERT [4] and conduct an ablation study to pinpoint the exact cause of this gain.
- We improve the accuracy of BERTweet by 0.4% by using an additional dataset for training.
- We further improve the accuracy by another 0.4% by ensemble learning.
- We shed light on a potential source of information leakage in the provided data and propose a solution for it.

II. Related Works

Early approaches for sentiment analysis have mostly focused on extracting hand-crafted features from text and feeding those to a classifier. In the interest of brevity, we refer the reader to [5] for a general overview and to [6] for a summary of sentiment analysis specifically on Twitter data.

With growing success of deep learning in natural language processing, the focus moved away from manual feature engineering, to learning word embeddings from data in an automated fashion. In addition, convolutional and recurrent neural networks were used as classifiers on top of the learned word embeddings. We refer to [7] for an overview.

After the introduction of the transformer architecture [8], several large scale language representation models derived from this architecture have been proposed [4, 9, 10]. They have brought substantial improvements in terms of effectiveness and accuracy in various NLP tasks with sentiment analysis among them. Such models are pre-trained in an unsupervised fashion on large-scale datasets, using massive amounts of compute power in order to be able to capture the semantics of written text. To perform sentiment analysis, a simple neural network classifier is appended to the pre-trained model. It is then fine-tuned for a few epochs on the task-specific classification data.

III. Data

A. Original Data

Our dataset consist of 2’500’000 labeled and 10’000 unlabeled tweets. The tweets were preprocessed as described in exercise 6.5 of the course[1]. The labels were obtained by finding positive `:)` and negative smileys `:(` in the tweets. The smileys were then removed from the data. Nevertheless we still found 1 positive and 23 negative smileys in the negative tweets, as well as 7 positive and 3 negative smileys in the positive tweets. Due to discovering this fact only late in the process, we unfortunately have not been able to remove those tweets from the data.

We remove duplicate tweets from the labeled data which leaves us with 1’142’838 negative and 1’127’644 positive tweets. Subsequently, we split the labeled data into 90% training and 10% validation data.

B. Additional Data

The idea of using smileys to automatically label tweets has been around for some time was originally introduced in [11]. Go et al. [12] have made another such dataset publicly available[2]. It consists of 800’000 negative and 800’000 positive tweets. We preprocessed them to match the format of the original dataset, which leaves us with 771’169 negative and 761’082 positive tweets. We use this additional dataset to investigate whether training on more data leads to an improvement in accuracy. The additional data is added only to the training split of the original data.

[1]http://www.da.inf.ethz.ch/teaching/2020/CIL/files/exercises/solutions06.pdf

[2]https://github.com/imoea/twitterSentimentClassifier

IV. Methods

A. Baselines

1) Google Natural Language API: In the last several years, many of the large cloud computing companies have started to offer machine learning as a service. The benefit is that a customer does not need to have any machine learning expertise or even a computing infrastructure.

We believe that a performance comparison of such a black-box service to solutions that were specifically designed for and trained on our task-specific data provides an interesting contrast. We choose the sentiment analysis functionality of the Google Natural Language API as an example for such a service.

2) GloVe: GloVe [13] is an unsupervised algorithm that generates word embeddings where semantically similar words are mapped to similar regions in the embedding space. We use the solution code from exercise 6.5[3] to perform the data preprocessing and the training of the embeddings. We create feature vectors for each tweet by averaging the embedding vectors corresponding to each word occurring in the tweet. These feature vectors can then be used to train a classifier.

3) BERT: BERT is a state of the art language representation model that uses bidirectional attention, meaning that it is able to represent a word in a sentence using both previous and following segments of the sentence. In our experiments, we utilize a pre-trained $\text{BERT}_{\text{Base}}$ model as described in [4] without any modifications as our baseline. For more details on BERT, we refer the reader to [4].

B. BERTweet

BERTweet [1] is a derivative of BERT. The differences can be summarized in the following three points:

1) BERTweet is based on the RoBERTa model [14]
2) BERTweet is pre-trained on Twitter-specific data
3) BERTweet represents user mentions and URLs as separate tokens

RoBERTa is also a descendant of BERT. It uses the same architecture, but revises BERT's pre-training procedure. Modifications include, but are not limited to: longer pre-training with substantially more data, training on longer sequences with larger batch sizes and a larger token vocabulary. For more details on the specific modifications, we refer to [14].

As outlined previously in Section II, these transformer-based language models are pre-trained on large language corpuses. In contrast to BERT and RoBERTa, BERTweet uses a corpus of tweets for pre-training. The characteristics of tweets differ from the text used for pre-training BERT and RoBERTa in their typical short length, as well as the use of informal grammar.

The third difference listed is best illustrated using an example: RoBERTa splits the URL tag `<url>` in our original dataset into 3 tokens `"Ġ<"`, `"url"` and `">"`, whereas BERTweet is able to keep it as a single token.

C. Ensemble learning

We employ two variants of ensemble learning: Simple model averaging where we trained the same model with different random seeds and bagging [15], where each model is trained on a bootstrap sample of the data. For both variants the ensemble predictions are obtained by averaging the prediction probabilities of all members in the ensemble.

V. Experiments

All hyperparameter choices, as well as instructions to reproduce our results can be found in our GitHub repository [4]. All reported scores, unless stated otherwise, are on the public part of the test set, which constitutes about 50% of the 10'000 unlabeled tweets[5].

A. Baselines

For the GloVe baseline we have tried several different classifiers, hyperparameters and number of embedding training epochs. The full analysis can be found in Appendix A. The best score is obtained with 400 epochs of embedding training and a random forest classifier.

Baselines	**accuracy** (in %)
Google Natural Language API	71.2*
GloVe	71.8
BERT	89.4

Table I: Results obtained for our baselines.
*Approximated value since the model has not produced a prediction for all tweets. For more details refer to Appendix B

In Table I we see that the Google Natural Language API and GloVe are very similar in performance, achieving around 70% accuracy. Unsurprisingly, BERT shows immense improvement over both.

B. BERTweet

To carefully analyze the performance of BERTweet in comparison with BERT, we conduct an ablation study where we compare BERT to a progression of models. They each differ in exactly one of the 3 points, outlined in Section IV-B, to its predecessor, i.e. RoBERTa, then a version of BERTweet that does not make use of the special tokens, and lastly the regular BERTweet model. The results are presented in Table II.

We see that BERTweet achieves 1.5% higher accuracy than BERT on average. About 0.6% of the gain is due

[3]https://github.com/dalab/lecture_cil_public/tree/master/exercises/2019/ex6

[4]https://github.com/ferraric/Computational-Intelligence-Lab-2020

[5]https://www.kaggle.com/c/cil-text-classification-2020/leaderboard

	original data		**with additional data**	
	accuracy (in %)	**sd**	**accuracy** (in %)	**sd**
BERT	89.6	0.2	90.2	0.4
RoBERTa	90.2	0.2	90.1	0.1
BERTweet (no special tokens)	91.2	0.2	91.3	0.1
BERTweet	91.1	0.2	91.5	0.3

Table II: Mean accuracy and standard deviation for 5 runs per model with different random seeds.

to using RoBERTa's pre-training procedure. This difference aligns with the findings in [14]. The remaining gain of about 1% are owed to pre-training the model on Twitter-specific data.

The fact that BERTweet outperforms BERT and RoBERTa has already been established in [1]. Here however, we discover that this is largely by reason of the domain specific pre-training data.

C. *Additional data*

Using additional data helps BERT to achieve a 0.6% increase in accuracy. Due to the high variance across model runs this difference should however not be overstated. For RoBERTa, as well as BERTweet trained without special tokens, the effect of additional data appears to be negligible. Lastly for BERTweet, we see an increase of 0.4%. Considering the measured standard deviation, the impact of additional data is rather small.

In conclusion we observe that training with additional data generally seems to help increase the accuracy score, although the effect depends on the model. Experiments to determine the reason for the varying impact are left to future work.

D. *Ensemble learning*

We have applied ensemble learning for BERTweet using 5 models with different random seeds.

Our results in Table III show that simple model averaging brings improvements in terms of accuracy, especially when additional data is not used. Bagging on the other hand seems to help without additional data, but perform marginally worse with additional data. Overall, it appears that bagging performs slightly worse than simple model averaging.

We would like to note that due to the prohibitive computational cost of training multiple ensembles, we are not able to provide confidence measures for the ensemble scores. Considering the rather small size of the test set, the variability of the scores might be substantial. Therefore the interpretation of individual results should not be assumed to be perfectly conclusive. To get a clearer picture, it helps to look at the same experiments on the validation data which can be found in Appendix C.

Driven by the competitive nature of this project, we further include the score of creating an ensemble of all 5 runs of BERTweet, BERTweet with additional data, BERTweet bagging and BERTweet bagging with additional data in Table III. This model, which we refer to as BERTweet total ensemble, is therefore an ensemble of 20 models. It achieves the highest accuracy score of all our models with 91.9%.

	accuracy (in %)		
BERTweet	single	model averaging	bagging
original dataset	91.1	91.5	91.4
+ additional data	91.5	91.6	91.3
		accuracy (in %)	
BERTweet total ensemble		91.9	

Table III: Accuracy score of a 5 member ensemble versus a single model (mean score).

We further investigate how the effect of ensemble learning behaves with increasing number of members.

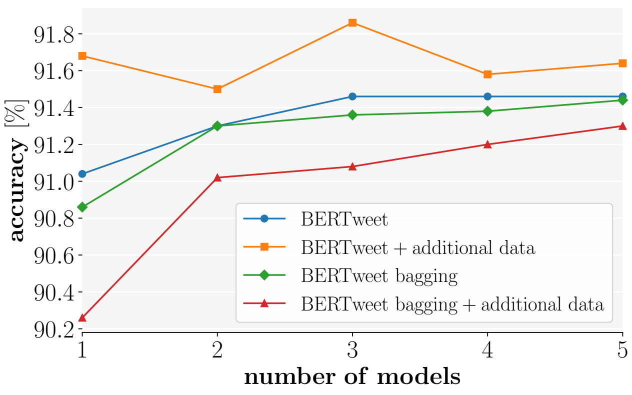

Figure 1: Accuracy for ensembles by number of members.

In Figure 1, we observe a trend of increasing performance with an increasing number of models combined for all models, except BERTweet with additional data. A look at the same experiment on the validation data in Appendix C shows a positive correlation of increasing ensemble members and accuracy for all models. This suggests that ensemble learning in general achieves better results with an increasing amount of models, and the effect is already visible with as few as 2 models.

VI. On Data Quality and Model Robustness

By manual inspection of our dataset, we notice that there are numerous unmatched parentheses in the tweets. Unmatched parentheses are opening parentheses that are never closed or closing parentheses without matching open parentheses. See the following tweet for an example:

`<user> watching now hehe ) )`

We hypothesize that the unmatched parentheses were once part of an exaggerated smiley, i.e. in this case `:))))`. If this were in fact true, we would have information leakage from the labels to our tweets. The most straightforward way to validate this hypothesis would be to compare the preprocessed data to the raw data. Unfortunately, our request to have access to the raw data was not granted[6]. Instead we follow a different approach: we build a rule-based system, called Parenthesis Rule, that classifies based on the unmatched parentheses in tweets in the following manner:

- Positive if the tweet contains an unmatched `")"`
- Negative if the tweet contains an unmatched `"("`
- Unknown if the tweet meets both conditions or has no unmatched parentheses

	accuracy (in %)	
	BERT	BERTweet
Model trained on original tweets		
Full validation set	89.4	91.1
Rule Subset	*98.2*	*98.6*
Full validation set modified	81.6	87.3
Rule Subset	*59.9*	*79.2*
Model trained on modified tweets		
Full validation set	88.9	90.9
Rule Subset	*95.4*	*97.9*
Full validation set modified	88.9	90.6
Rule Subset	*95.7*	*96.5*

Table IV: Results for our parentheses analysis. The modified tweets are the original data with unmatched parentheses removed wherever the Parenthesis Rule does not predict "unknown".

Since access to subsets of labels is needed, all results in this section are calculated on the validation set. The Parenthesis Rule is able to classify 19.94% of the validation set as negative or positive. We shall refer to this subset of the validation set as "Rule Subset". On the Rule Subset, the Parenthesis Rule is able to achieve an accuracy of 95.3%, indicating that there is some validity to our hypothesis.

To see how that potential information leakage affects our models, we evaluate one BERT and one BERTweet model on the Rule Subset. As we can see in Table IV both BERT and BERTweet achieve over 98% accuracy on those tweets, which is almost 10% higher than the accuracy on the full validation set. To confirm that the superior performance on the Rule Subset is indeed related to the unmatched parentheses, we remove them from the Rule Subset and evaluate the same models on it again. The accuracy of the same BERT and BERTweet models on the Rule Subset drops by about 20% for BERTweet and a staggering 40% for BERT. This implies that both models rely heavily on unmatched parentheses. We see this as strong evidence that our initial hypothesis holds, i.e. that unmatched parentheses are remainders of labels within our tweets.

This data leakage is undesired. As we have just shown, it can lead to models that rely on these patterns which originated as an artefact of the data collection process. What we instead would prefer are models that are able to infer the semantic information of a tweet and use that to determine the sentiment.

To this end, we investigate if we can train models that are robust against this type of data leakage. We train the same BERT and BERTweet models on tweets where the unmatched parentheses are removed wherever the Parenthesis Rule does not predict "unknown". In Table IV we see that the validation accuracy drops slightly for both models compared to when they were trained on the unmodified data. However, when evaluated on the modified validation data, in particular on the Rule Subset, we observe very similar performance as on the unmodified validation data. We infer that the models trained on the modified tweets do not rely on this parentheses pattern anymore. We would argue that such models are of more value, yet the scores do not reflect this preference.

We admit that the evidence presented here is not definitive, but we argue that it is strong enough to warrant further investigation. If our doubts about the data would turn out to be true, the issues could be fixed and future Computational Intelligence Lab students would profit from a more clean dataset and a leaderboard that is more aligned with what we believe is the goal of sentiment analysis: finding the sentiment based on the semantic content of text.

VII. Conclusion

In this work we have shown that transformer-based architectures significantly outperform other considered baselines. We have further confirmed the claim of [1] that their model, BERTweet, improves significantly over other transformer-based models. By performing an ablation study we have found that the improvement is for the most part owed to pre-training on Twitter-specific data.

Additionally we have used both an additional dataset and ensemble learning to increase the accuracy of BERTweet by a total of 0.8%.

Lastly, we draw attention to a potential source of information leakage in the provided dataset. We also suggest a solution that in our opinion would improve the project for future generations of CIL students.

[6]https://piazza.com/class/k6hqt5l6hyd46t?cid=154

References

[1] D. Q. Nguyen, T. Vu, and A. T. Nguyen, “Bertweet: A pre-trained language model for english tweets,” 2020.

[2] E. Kouloumpis, T. Wilson, and J. Moore, “Twitter sentiment analysis: The good the bad and the omg!” in *Fifth International AAAI conference on weblogs and social media*, 2011.

[3] M. V. Mäntylä, D. Graziotin, and M. Kuutila, “The evolution of sentiment analysis—A review of research topics, venues, and top cited papers,” pp. 16–32, feb 2018.

[4] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” *arXiv preprint arXiv:1810.04805*, 2018.

[5] B. Liu and L. Zhang, “A survey of opinion mining and sentiment analysis,” *Mining Text Data*, pp. 415–463, 2013.

[6] A. Giachanou and F. Crestani, “Like it or not: A survey of twitter sentiment analysis methods,” *ACM Computing Surveys*, vol. 49, pp. 1–41, 06 2016.

[7] L. Zhang, S. Wang, and B. Liu, “Deep learning for sentiment analysis : A survey,” 2018.

[8] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” *CoRR*, vol. abs/1706.03762, 2017. [Online]. Available: http://arxiv.org/abs/1706.03762

[9] A. Radford, “Improving language understanding by generative pre-training,” 2018.

[10] Z. Yang, Z. Dai, Y. Yang, J. G. Carbonell, R. Salakhutdinov, and Q. V. Le, “Xlnet: Generalized autoregressive pretraining for language understanding,” *CoRR*, vol. abs/1906.08237, 2019. [Online]. Available: http://arxiv.org/abs/1906.08237

[11] J. Read, “Using emoticons to reduce dependency in machine learning techniques for sentiment classification,” in *Proceedings of the ACL Student Research Workshop*, ser. ACLstudent ’05. USA: Association for Computational Linguistics, 2005, p. 43–48.

[12] A. Go, R. Bhayani, and L. Huang, “Twitter sentiment classification using distant supervision,” *Processing*, vol. 150, 01 2009.

[13] J. Pennington, R. Socher, and C. D. Manning, “Glove: Global vectors for word representation,” in *Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543. [Online]. Available: http://www.aclweb.org/anthology/D14-1162

[14] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, “Roberta: A robustly optimized bert pretraining approach,” 2019.

[15] L. Breiman, “Bagging predictors,” *Mach. Learn.*, vol. 24, no. 2, p. 123–140, Aug. 1996. [Online]. Available: https://doi.org/10.1023/A:1018054314350

Appendix A. GloVe Experiment Details

Classifier		embedding training epochs 20	100	200	400
Log. Regression	val. accuracy	58.2	61.9	63.6	65.8
	test accuracy	59.7	58.7	60.1	62.5
Decision Tree	val. accuracy	60.7	63.4	64.8	66.2
	test accuracy	57.2	61.5	61.4	63.9
Random Forest	val. accuracy	66.8	69.2	71.3	73.2
	test accuracy	65.5	66.8	69.1	71.8

Table V: Results using GloVe with different classifiers.

From our extensive experiments with GloVe, we see that we are able to push the accuracy score substantially regardless of the classifier used, by increasing the number of epochs, as is evident in Table V. This means that with more epochs, we obtain word embeddings which can be more clearly separated in the embedding space.

Appendix B. Google Natural Language API Results

Google's Natural Language API is not able to produce a prediction for all the tweets. Specifically, it is able to make a prediction for 8'682 out of the 10'000 unlabeled test tweets. For all others we submit an invalid prediction of 0, which is always counted as a misclassification. To make the score comparable to other models, we need to find the score conditioned on the model being able to make a prediction. Since we do not know how many of the 8'682 were in the public part and how many tweets the public part contains overall, we divide the score by 0.8682. This would be correct if the fraction of valid predictions were the same in the public part as in the whole test set, an assumption we feel is reasonable.

Appendix C. Ensemble Analysis on the Validation Set

	accuracy (in %)		
BERTweet	single	model averaging	bagging
original dataset	91.1	91.5	91.3
+ additional data	91.2	91.5	91.3
	accuracy (in %)		
BERTweet total ensemble	91.6		

Table VI: Accuracy score comparison when ensemble learning is applied versus a single model (mean score) on the validation set.

In Table VI, we see similar outcomes when comparing results on the test set with results on the validation set. A notable improvement is evident when using simple model averaging, while bagging is less effective.

The only notable difference between both results is the score for the single model of BERTweet using additional data. On the validation set, the score is lower by 0.3% compared to the score on the test set, however, as we discuss in Section V-D, these measures should not be considered as definitive due to the variability of the obtained scores.

In Figure 2 we observe a general trend of increasing accuracy with increasing number of ensemble members on the validation set. For all combinations of BERTweet, we see the accuracy rise by 0.3 to 0.7%. BERTweet with bagging and additonal data sees the largest gain in accuracy.

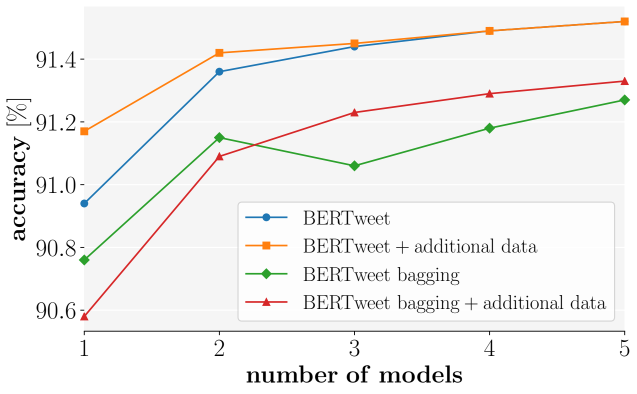

Figure 2: Accuracy score on our validation set for ensembles of k models.