Proceedings of the 15th Conference on Natural Language Processing (KONVENS 2019)

bertZH at GermEval 2019: Fine-Grained Classification of German Offensive Language using Fine-Tuned BERT

Tim Graf
University of Zurich
Institute for Computational Linguistics
Andreasstrasse 15, 8050 Zrich
tim.graf@uzh.ch

Luca Salini
University of Zurich
Institute for Computational Linguistics
Andreasstrasse 15, 8050 Zrich
lucaelio.salini@uzh.ch

Abstract

The bertZH system for abusive tweet detection in the GermEval 2019 competition is a neural classifier based on BERT (Devlin et al., 2018). We describe our submission runs for subtask 2 on fine-grained classification of tweets. We used the pretrained German language model from deepset.ai[1] implemented in `pytorch` and fine-tuned it to the data of the task, before then using it to train on the classification task. We also experimented with the pretrained multilingual BERT model from Google Research implemented in `keras`, but it resulted in a worse score than with the German model. We have found that a language-specific BERT model outperforms a multilingual model and that fine-tuning a BERT model to the tasks domain achieves a small gain in performance.

1 Introduction

It can be very useful for the user experience of a social media platform to sort out abusive content. But first one has to know what content can be declared as abusive in order to avoid false-positives. The goal of our deep neural network is to find this abusive content.

We developed our models as a part of a Text Mining course at the University of Zurich as a final work. We are two Bachelor students in Computational Linguistics.

This paper is organized as follows: in section 2 we will explain the details of the competition, especially task 2. Then, in section 3 we provide some details about the preprocessing of of our pipeline. In the 4th section we present the architecture of our deep neural network and the background of BERT. In section 5 we describe the configuration of each submitted run in detail and we finally present our results in section 6.

[1]`https://deepset.ai/german-bert`

2 Competition Tasks

The GermEval 2019 Shared Task on the Identification of Offensive Language[2] focused on classification of German tweets with respect to their offensiveness. With the overwhelming amount of social media posts everyday, systems that can reliably detect profane language or harassment grow more important in assisting human moderators.

- Subtask 1: Coarse grained classification. This dataset was labelled with only two labels, namely OFFENSIVE and OTHER, where OTHER represents non-offensive tweets.
- Subtask 2: Fine grained classification. For this task, each sample of the dataset (which is the same as in subtask 1) is labelled with four labels: INSULT, PROFANITY, ABUSE and OTHER.

We only participate in subtask 2. The task is a multi-class classification problem, which means that each tweet is only labelled with a single label (e.g. an abusive tweet that uses profane language is only labelled ABUSE). The data is not uniformly distributed as the class OTHER has a frequency of 67.8%, while the others are quite under-represented (INSULT: 15.6%, ABUSE: 12.7%, PROFANITY: 3.8%). This usually makes it very difficult to learn automatically how to predict the under-represented classes - especially the class PROFANITY.

The evaluation metric is F1-score, hence it is important to have a good classification rate for every single class.

3 Preprocessing

Since tweets contain a lot of colloquial language and also hashtags or usernames or similar, we

[2]`https://projects.fzai.h-da.de/iggsa`

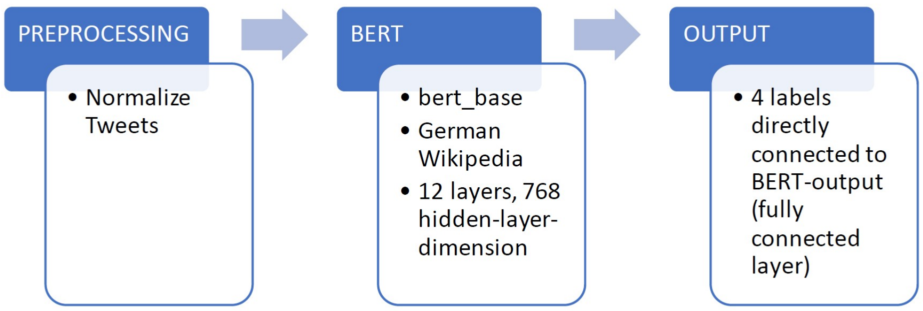
PREPROCESSING
• Normalize Tweets
BERT
• bert_base
• German Wikipedia
• 12 layers, 768 hidden-layer-dimension
OUTPUT
• 4 labels directly connected to BERT-output (fully connected layer)

Figure 1: Top-level overview of our `pytorch` architecture.

needed to filter or normalize such occurrences. For that reason we used the German tokenizer `SoMaJo`[3] and added some extra cleaning steps:

- Normalizing character repetitions: We replace characters which occur more than twice in a row with two of them (”coooooool” → ”cool”).
- Substituting usernames: Every username gets replaced with ”@USER”. We didn’t cut out the whole username because it could be important for the classification if someone is mentioned.
- Removing special characters: We remove characters such as hashtags, newlines, linebreaks or underscores.

We also used `scikit-learn` (Pedregosa and others, 2011) for the train-test split during development.

4 Architecture

BERT (Devlin et al., 2018) has proven to be exceptionally effective in many downstream NLP-tasks including sentence classification. It has improved the state-of-the-art in several applications, and hence our goal was to implement BERT for the fine-grained classification task. However, training a BERT model from scratch is computationally very expensive and impossible to train on a single consumer-grade GPU, so we had to rely on the publicly available models. When we first started the project, the only available BERT model that was trained on German data was `bert_multi_cased_L-12_H-768_A-12`, a BERT model released by Google Research on GitHub[4] that was trained on Wikipedia dumps in 104 different languages, of which one was German. In order to use the model in `keras` (Chollet and others, 2015), we followed Jacob Zweig’s blogpost *BERT in Keras with Tensorflow hub*[5]. With this implementation, we could fine-tune the last *n* layers of the BERT transformer while connecting a 256-units Feed-Forward layer with dropout to the first generated token by BERT. This `[CLS]` token is a representation of the whole sequence and is the only component of BERT’s output we use to perform the classification task (Devlin et al., 2018).

Later on in the project, we found that deepset released a BERT model to the public that was trained on German data exclusively (`bert-base-german-cased`)[6], which was promising better results on several German tasks than the multilingual model by Google Research, including the GermEval 2018 Shared Task on the Identification of Offensive Language (Wiegand et al., 2018). The implementation in `keras` we described in the preceding paragraph relied on the model being available as a module on *TensorFlow Hub*, which was not the case for this model. Hence, we used the well-known `pytorch` (Paszke et al., 2017) implementation of BERT by the HuggingFace team[7] and followed the blogpost *A Simple Guide On Using BERT for Binary Text Classification*[8] by Thilina Rajapakse to be able to use the German model, and modified the code to suit the multiclass classification task. With the implementation by HuggingFace we were able to fine-tune the German model on to the tasks dataset using BERT’s original language modeling tasks MLM

[3]`https://github.com/tsproisl/SoMaJo`
[4]`https://github.com/google-research/bert`

[5]`https://towardsdatascience.com/bert-in-keras-with-tensorflow-hub-76bcbc9417b`
[6]`https://deepset.ai/german-bert`
[7]`https://github.com/huggingface/pytorch-transformers`
[8]`https://medium.com/swlh/a-simple-guide-on-using-bert-for-text-classification-bbf041ac8d04`

(masked language modeling) and next sentence prediction before we trained the model to actually perform classification (Devlin et al., 2018). For classification, we then used the already provided `BertForSequenceClassification` model architecture without modifying it at all. We conducted some informal experiments on the following hyperparameters:

- Number of layers to fine-tune BERT (`keras`-implementation)
- rate of dropout (`keras`-implementation)
- Learning rate (both implementations)
- Number of epochs

5 Submitted Runs

5.1 Run 1

Run 1 was a submission that was trained on 11536 samples and evaluated on 1000 samples. It used the `pytorch`-implementation with the German BERT model and the following Hyperparameters:

Hyperparameter	**Size**
Learning rate	0.00002
# of epochs	5

Table 1: Set Hyperparameters of run 1.

5.2 Run 2

Run 2 was a blind submission, which we trained on all of the available 12536 samples, which means we did not know how well the system would actually perform. It was a `pytorch`-implementation using the German BERT model as well and used the following Hyperparameters:

Hyperparameter	**Size**
Learning rate	0.00002
# of epochs	5

Table 2: Set Hyperparameters of run 2.

5.3 Run 3

Our third submission was made with the `keras`-implementation and Google-Research's multilingual model. Even though we were observing significantly worse performance using this model, we were interested in how well this model would perform. This submission was trained with the following Hyperparameters:

Hyperparameter	**Size**
Learning rate	0.00002
# of epochs	3
# of fine-tuned layers	3
Dropout	0.5

Table 3: Set Hyperparameters of run 3.

5.4 Training Times

All of our experiments were conducted on a single RTX 2080ti GPU. The fine-tuning of the German BERT model took around 60 minutes for 3 epochs, and the training of the classification tasks for runs 1 and 2 took around 15 minutes. The *keras*-implementation of run 3 finished in 6 minutes. It is very impressive that using such powerful and large models is possible within very reasonable time-frames on consumer grade GPUs, and the practice of open-sourcing these large pretrained models should be applauded.

6 Results

We pre-calculated the F1-score for our different systems:

Features	**F1**	**Diff**
run 2: with all data	-	-
run 1: German BERT + fine-tuning	0.65	-
(no submission): German BERT	0.63	-0.02
run 3: multilingual BERT	0.53	-0.12

Table 4: **Pre-calculated** F1-scores of the models.

Features	**F1**	**Diff**
run 2: with all data	0.53	-
run 1: German BERT + fine-tuning	0.52	-0.01
(no submission): German BERT	0.50	-0.03
run 3: multilingual BERT	0.43	-0.1

Table 5: Final F1-scores of the models.

From these results it is obvious that a language specific BERT model improves the performance of a system. This should hold for any language. We also assume that our models are not yet saturated and that more training data would help achieve

Class	F1
OTHER	0.87
ABUSE	0.59
INSULT	0.54
PROFANITY	0.56

Table 6: F1-score distribution of the different classes for run 1: *German BERT with fine-tuning.*

an even higher score without any modification to the model, especially because more samples from the underrepresented classes should help the BERT model to get a better grasp of what for example makes a tweet an INSULT and not an abusive tweet. Another observation to point out is that fine-tuning a BERT model to task-specific data seems to improve the score even further. Hence, given enough training examples, BERT might be all you need.

7 Conclusion

After BERT has revolutionized the NLP-Community, we have applied it to the task of German offensive language detection. A common problem with neural approaches is that they usually require a larger amount of training data than more traditional machine learning approaches. However, large, pre-trained language models seem to model a language well enough so that even with a rather small dataset of 12536 they can be used to achieve impressive results. It was also very impressive to see that, even though PROFANITY made up only 3.8% of the training data, without any further data augmentation or oversampling, the BERT model did not face the problem of not predicting PROFANITY at all. Hence, our submission shows that relatively good results can be achieved without spending many resources on feature engineering or training large models, as fine-tuning existing released models does not take a lot of time.

Acknowledgments

We would like to thank deepset.ai and Google for making their BERT models available to the public. Furthermore, we thank our instructor Simon Clematide for his support and inputs.

References

François Chollet et al. 2015. Keras. `https://keras.io`.

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*.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch. In *NIPS Autodiff Workshop*.

Fabian Pedregosa et al. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Michael Wiegand, Melanie Siegel, and Josef Ruppenhofer. 2018. Overview of the germeval 2018 shared task on the identification of offensive language.