OffensEval - Offensive Language Identification

Peter Svenningsson
petsvenn@student.chalmers.se
920114-1257

Olle Månsson
ollema@student.chalmers.se
950623-3833

Introduction

Offensive language is pervasive in social media. Individuals frequently take advantage of the perceived anonymity of computer-mediated communication, using this to engage in behavior that many of them would not consider in real life.

As manual filtering of posts in social media is time consuming, and as it can cause post-traumatic stress disorder-like symptoms to content moderators (Arsht and Etcovitch 2018), there have been many research efforts aiming at automating the process.

The 2019 SemEval task with the name OffenseEval is one of those efforts. Here, the task is divided into three different sub-tasks. For this project, we have focused on sub-task A: Offensive Language Identification. Using NLP, our aim to classify a number of social media posts as either offensive or not offensive. To our help, we have a dataset of over 13 000 posts from the social media platform Twitter. We will refer to these posts as *tweets*. The tweets in this dataset have been labeled by at least two independent parties and if there is a disagreement, a third party has acted as the tie-breaker. The dataset is named OLID and was presented in the paper (Zampieri et al. 2019).

Definition of offensive During the production of the dataset OLID the following definition of a offensive tweet was used

> Posts containing any form of non-acceptable language (profanity) or targeted offense, which can be veiled or direct. This includes insults, threats, and posts containing profane language or swear words. (Zampieri et al. 2019).

This definition is subjective since no set of non-acceptable language is provided. Also, it can be challenging to identify a veiled offense when no wider context of the conversion is provided. It follows that it is likely that there exists mislabeled data even though the above described consensus based annotation process is used.

Methods

BERT - Bidirectional Encoder Representations from Transformers

BERT is a deeply bidirectional unsupervised language representation pretrained using a large label-free corpus. When BERT was introduced it was referred to as a milestone for NLP achieving new state of the art results in many common NLP benchmarks, as seen in Devlin et al. (2018) Devlin et al. (2018).

In the BERT model words are embedded based on their tokens as well as their positions in the sequence. The architecture is then based on the Transformer architecture presented in the paper Attention is All You Need (Vaswani et al. 2017). The Transformer architecture is a encoder-decoder architecture. The encoder consists of chained encoder blocks. A encoder block is comprised of a multi-head attention which feeds into a linear layer. Multi-head attentions consists of multiple scaled dot-product calculations which are concencated. BERT consists only of the encoder architecture from the Transformer making it a versatile encoder for any number of NLP-tasks.

Tokenization A unique feature of the BERT model is the use of the pretrained tokenizer WordPiece. Unlike other tokenization schemes WordPiece does not replace unknown words with a ”unknown”-token. Rather it splits up the unknown into smaller word segments which are in the vocabulary. In this way WordPiece avoids the out-of-vocabulary problem.

Model Distilation This study used the DistilBERT model which is a distilled version of a pretrained BERT model. The DistrilBERT model is able to retain more than 97% of the performance of the BERT model while having 40% fewer parameters. Therefore the DistilBERT model requires less working memory and less compute to run.

The process of knowledge distillation is to compress a large model like BERT, the teacher model, into a smaller model, the student model. The student model is trained to reproduce the results of the larger model. The reason why this is useful is because commonly the ground truth for any given observation is one label. However, models commonly output a probability distribution of all discrete labels. These distributions may contain insight in how the model generalizes information. In fact, for tasks like masked word prediction one can recognize that the ground truth should be a distribution of words rather than one label since several words may be semantically valid and approximately as likely. For example ”I like blue cars” and ”I like red cars” are approximately equally correct predictions for the masked word prediction ” I like [MASK] cars”.

Therefore by using the probability distribution output of the teacher network as the ground truth for the training of the student network one can achieve a loss function which is

more rich in information, namely one can use the Kullback-Leibler loss $L_{KL} = \sum_i p_i * \log(p_i) - \sum_i p_i * \log(q_i)$ for the student prediction distribution p_i and the teacher prediction distribution q_i. In this formulation the Kullback-Leibler loss, often called divergence, represents the distance between two discrete probability distributions given the assumption that they are in the exponential family of distributions.

Unsupervised fine-tuning The BERT model is able to train on label-free data by solving two unsupervised tasks, mask language modelling, and next sentence prediction. The MLM task is to predict one or several masked words in a sequence based on the context of the remaining words. The NSP task is a binary classification task which consist of determining if two sentences follow eachother in the text. This allows the model to learn the relationship between sentences.

This study uses models which are pretrained on the MLM and NSP task on a very large corpus. However it can be beneficial to do additional unsupervised fine-tuning of the network on a corpus from the same domain as the intended task. However, this was not done for this study.

Supervised fine-tuning For the binary classification task of categorizing a tweet into the categories $\{\text{Offensive}, \text{Not Offensive}\}$ the supervised fine-tuning of the model consisted of adding a binary linear classifier to the output of the BERT encoder and training the network from the cross-entropy loss of the label and the prediction.

Model parameters and training

All models used the same parameters for DistilBERT, regardless of pre-processing. We use the model called "distilbert-base-uncased" which translates to the distilled base BERT model for uncased tokens. We also used an extension called DistilBertForSequenceClassification, which includes a linear layer on top of the pooled output. Default values were used for all parameters, including the number of labels.

As for the optimizer, a modified version of the popular optimizer Adam is used. This version is called BertAdam and the optimizer is bundled with the BERT package. In addition, a scheduler gradually reduced the learning rate after each training step. For learning rate, experiments and previous work showed the an initial learning rate of 2e-5 was a good choice for fine-tuning the last layer.

For all models, any additional training after 2 epochs proved to be detrimental to the macro F1-score so training was limited to 2 epochs. This is also reflected in previous work with BERT.

Pre-processing

The OLID dataset is noisy in several regards. The dataset is exclusively extracted from the social media platform Twitter. The text contains much language use specific for the domain as well as misspellings. A example of a offensive tweet is:

"@USER Are you an ostrichs? ☺#AnimalFarm"

We can note that the frequent use of special UTF-8 character like emojis can be problematic since the WordPiece tokenizer will set these to a unknown-token. Also, keywords are preceded by the # symbol. Several procedures were implemented to make the text data more similar to the corpus BERT has been pretrained on.

Spelling correction The popular library SymSpell was used correct misspelled words. The library uses a dictionary lookup method based on the Damerau–Levenshtein distance. Damerau–Levenshtein distance is a string metric for measuring the edit distance between two strings. For the implementations words were only replaced if they only contained alpha characters and had a Damerau–Levenshtein distance of 1 to the closest in-dictionary word. The reason for this is to not replace words which are correctly spelled but missing from the dictionary.

Hashtag segmentation Keywords preceded by the hashtag symbol were identified and segmented using a dictionary based approach. The python library WordSegment was used to segment the words using a 3-trillion word dictionary.

Text representation of emoji The python library Emoji is used to replace utf-8 emoji characters by a shorthand text representation as defined by the unicode consortium. E.g. the character ☺is replaced by :smile:. Any semicolons are then dropped and underscores are replaced by spaces.

Adding in-domain specific tokens It may be beneficial to add out-of-vocabulary words which are common in the training data to the WordPiece tokenizer. These embeddings are initialized randomly but will hopefully be meaningfully tuned during the supervised fine-tuning. The most common out of vocabulary words in set of offensive tweets was found to be:

{maga, antifa, lol, kavanaugh, gop, tweet, nra, dems}

The word maga occurred at least 500 times in the training set and all of these words occurred at least 25 times in the training set.

Macro F-score

The statstic used to benchmark the models is the Macro F-score as specified by the OffensEval comptetition. The Macro F-score is the Arithmetic average of F1-scores from several classes. For this study considering binary classification two F1-scores are calculated by first considering the setting of retrieving the Offensive class and then the setting of retrieving the Not Offensive class. No advantage of using the Macro F-score over the F1 score for the class offensive has been found by the writers but the two statistics do provide different results for binary classification.

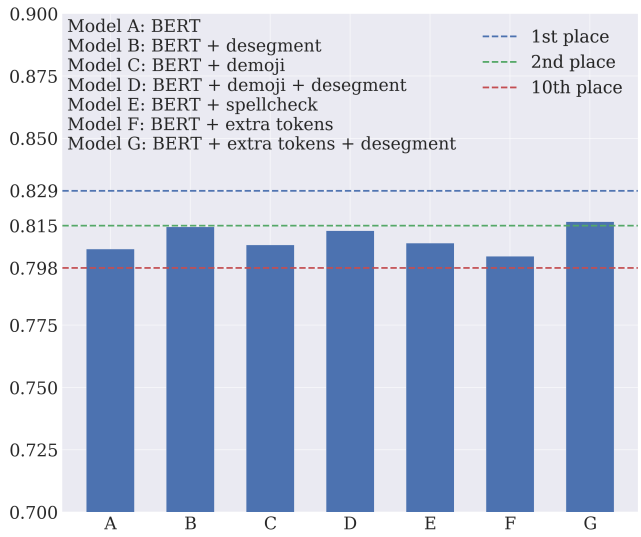

Figure 1: The average macro F-score over 5 training instances for 7 models differentiated by the pre-processing operations.

Results

Using the prepossessing techniques described above a number of models were generated. The macro F-scores for these models can be found in figure 1. The macro F-scores were presented is the average macro F-scores of five training instances of the models. Three benchmarks from the OffensEval 2019 competition are also shown in the figure. Keep in mind that five training instances are not enough to get any statistical significance but computational resources limited the scope of this study.

The best model used hashtag segmentation and extra in-domain tokens as pre-processing steps and achieved slightly higher results than the second place in the competition. Moreover, the model outperforms the baseline model A which has no pre-processing stage by roughly two percentiles. However, the model performs similarly well when the extra in-domain tokens are removed as seen in the results for model B in figure 1.

It is noteworthy that correcting misspelled words does not impact the models effectiveness meaningfully. Similar results are found text representations of emoiji characters.

Figure 2 displays the confusion matrix averaged over five training instances for the best performing model, model G. We can note that there exists class imbalance in the test set 72% of the sequences are Not Offensive. Roughly the same class imbalance is found in the training set. For the class Offensive a recall of 71% and a precision of 76% is achieved.

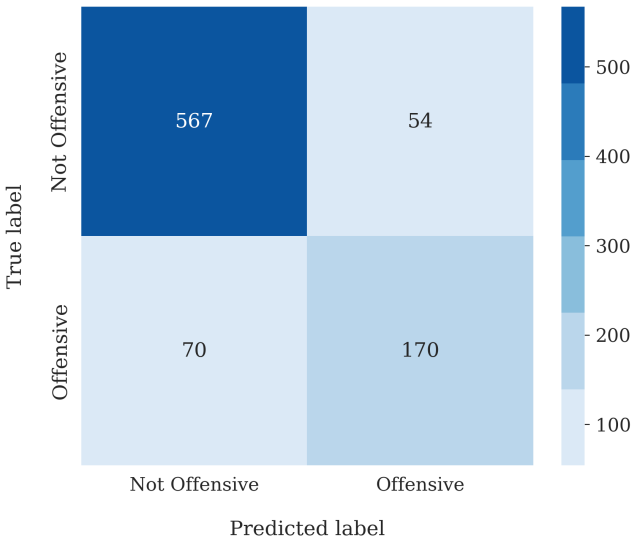

Figure 2: Confusion matrix for model G averaged over five training instances.

Model	**Macro F1-score**
1st place OffensEval 2019	0.829
BERT + extra tokens + desegment (ours)	0.817
2nd place OffensEval 2019	0.815
BERT + desegment	0.814
BERT + demoji + desegment	0.813
BERT + spellcheck	0.808
BERT + demoji	0.807
BERT	0.806
BERT + extra tokens	0.803
10th place OffensEval 2019	0.798

Table 1: The average macro F-score over 5 training instances for 7 models differentiated by the pre-processing operations.

Conclusion

Model choice As seen in the results in table 1 the best model using keyword segmentation and extra tokens achieved a average macro F-score of 81.7% and the second best model using only keyword segmentation achieved a average macro F-score of 81.4%. These values are very similar. So one should in accordance with the principal of parsimony chose the second best model using only keyword segmentation. Since the extra in-domain tokens adds complexity not only to the pre-processing step of finding commonly out-of-vocabulary words but also adds parameteric complexity to the model as embeddings are initialized for the new tokens.

Addition of in-domain tokens The model was not improved by adding tokens for common in-domain words. The likely reason for this is because the dataset is too small to learn the newly initialized embeddings for these words. Something that could have mitigated this problem would have been to complete additional unsupervised learning on a dataset in the same domain using the MLM and NSP tasks. Because of time constraints this was not implemented however this would be a area of future study.

Correcting misspelled words The model was also not improved by correcting misspelled words. Initially this is surprising, exemplified by the following offensive tweet in the dataset

"@USER Good!! This will surely show us how croocked he is. ☺☺☺"

in which the only words which indicate that the tweet is offensive is misspelled. The expectation is that if this word was corrected to crooked then it is easier for the model to correctly classify the tweet. However the spell-correction algorithm used is a simple dictionary lookup algorithm which replaces any out-of-dictionary words if it can find a word within a edit distance of 1. In practice this resulted in more unintended behaviour than misspellings corrected. For instance the common abbreviation lol was replaced with the word low, the word phony was replaced with phone. Since this model similarly to BERT without pre-processing we have a indication that this process created as much noise as it created informative features.

The authors have concluded that using spell-correction algorithms based on edit distance in a automated pipeline is ill-advised for any application.

Emoji text replacement The emoji text replacement preprocessing made no improvements to the model. A reason for this may be that the emojis are simply not informative. Another reason may be that the description of the emojis are not representative of the meaning the user is attempting to convey. A example of this would be the 👊 emoji which is replaced with the text incoming fist which described the image well but likely does not capture the meaning. A third reason is that replacing emojis with text creates incorrect grammatical structures which is not similar to the corpus BERT was pretrained on.

Comparison with entries from the official competition Comparing the results to the official competition last year one could see that the results achieved in this study would do quite well. In fact, all five tested seeds with the best performing model outperforms every other model in the competition except the winning one. With that said, when developing these models, we have had the luxury to be able to run tests against the test set after each training session. This does not affect the training per say, since all training is done exclusively on the training data, but it does allow for quick and easy verification that any potential changes are useful.

While other teams from last year competition also trained their models using exactly the same dataset, they did not have the opportunity to test their models on the test set. Instead, they used an additional set called the trial set. This set also raises some questions, because the teams that performed the best on the training and the trial set did not perform the best on the test set. What is even more interesting is that the top performing team performed quite poorly on the trial set. Maybe the trial set and the data set captured different types of offensive posts and teams who focused on a high F1-score on the available training and trial set ended up with models that were too specialized.

Overall, it seems like many of the approaches investigated in this report and strategies used by other teams does require more data to be successful. This might explain why the newly released training set for the 2020 edition of the competition contains several magnitudes more training data.

Closing remarks on pretrained BERT models When working with pretrained BERT models it is now our experience that one should strive to pre-process the dataset so that it is as similar as possible to the corpus the model was pre-trained on. One should be vary of naive attempts to engineer features which one expects to be informative. Since BERT has displayed capabilities to encode a highly nuanced data representation one should rather employ a more holistic approach focused on leveraging the BERT encodings appropriately.

References

[Arsht and Etcovitch 2018] Arsht, A., and Etcovitch, D. 2018. The human cost of online content moderation.

[Devlin et al. 2018] Devlin, J.; Chang, M.; Lee, K.; and Toutanova, K. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR* abs/1810.04805.

[Vaswani et al. 2017] Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is all you need. *CoRR* abs/1706.03762.

[Zampieri et al. 2019] Zampieri, M.; Malmasi, S.; Nakov, P.; Rosenthal, S.; Farra, N.; and Kumar, R. 2019. Predicting the Type and Target of Offensive Posts in Social Media. In *Proceedings of NAACL*.

Implementations

Notebook:
https://colab.research.google.com/drive/1op1Cc0WvvalHMSrpG8jzclhXRg0K7utn