Changes in Baselines of Hatespeech Detection with pretraining and Transformers

Anonymous ACL-IJCNLP submission

Abstract

Since the foundational papers [Vaswani et al., 2017] and [Devlin et al., 2018] attentive systems, especially transformer models have taken the world of natural language processing by storm. Using the ability to inject context into embeddings and finetune them based on a wide array of natural language tasks, these have yielded very strong and rigid results. Now the question is if baselines, like in [Davidson et al., 2017], are still valid or if the out-of-the-box performance of transformers validates a readjustment of baselines. We finetune a Distilbert model on the dataset of Davidson et al., analyse the performance of the embeddings and propose a new baseline on the aforementioned dataset.

1 Introduction

In their paper [Davidson et al., 2017] propose a tweet dataset of about 25.000 handannotated tweets and use some basic methods to establish baselines. Especially interesting is that BoW approaches seem to have major problems distinguishing between hatespeech and offensive language since the distributions are so similar. This might indicate that context sensitive methods should be used. For example transformer/BERT-models yield such a functionality, so our focus will lie on those methods.

The Research Question: We aim to provide sufficient evidence that with transformer-models we need to adjust the baselines proposed by Davidson et al. up, to accommodate their stronger out-of-the-box performance.

As a convention we will use **bold font** to refer to codebase related entities, for example: "...in the notebook **data_pipeline** one finds the function **preprocess_tweet** which handles preprocessing".

Now to remind the reader of the central points of Davidson's paper we will review a few aspects of it here.

1.1 The Model

They use Tidf-vectorizers on the tokens and on the POS-tags of the tweet to get a vectorized version of the tweet on the syntactic and on the word-level. Then calculate a few more features based on reading-ease levels, counts of hashtags,mentions,urls and sentiment. All these get fed into a feature selecter via logistic regression with L1-regularization (using the liblinear solver) for dimensionality reduction and then predictions are generated using a **LinearSVC** with L2-penalty and squared hinge loss.

1.2 The Results

Davidson et al. claim that their final model has an overall precision of 0.91, an overall recall of 0.9 and a f1-score of 0.9. They also claim a precision of 0.44 and recall of 0.61 (which implies a f1-score of about 0.525) on the hate-class. These claims come with an asterisk, because these numbers do not represent the performance on a test set but rather using the entire data to train and evaluating effectively on the training-set. It is impossible to really judge how much overfitting inflates the numbers here, but to keep everything comparable we will use this methodology as well. We were able to replicate a model which achieves an overall precision of 0.87, recall of 0.85 and f1-score of 0.85 trained and evaluated on the entire dataset. And a precision of 0.48, recall of 0.47 and an f1-score of 0.44 on the hate-class. Which is quite a bit less than what was reported, this could be due to version differences in **sklearn** messing up the hyperparameters.

Table 1: classification report for our reproduction of Davidson's model with features

class	precision	recall	f1-score	support
hateful	0.5	0.4	0.44	1430.0
offensive	0.95	0.87	0.91	19190.0
neither	0.63	0.91	0.75	4163.0
metric	precision	recall	f1-score	support
accuracy	nan	nan	0.85	24783.0
macro	0.69	0.73	0.7	24783.0
weighted	0.87	0.85	0.85	24783.0

Table 2: classification report for our reproduction of Davidson's model without features

class	precision	recall	f1-score	support
hateful	0.4	0.53	0.46	1430.0
offensive	0.94	0.88	0.91	19190.0
neither	0.69	0.84	0.76	4163.0
metric	precision	recall	f1-score	support
accuracy	nan	nan	0.85	24783.0
macro	0.68	0.75	0.71	24783.0
weighted	0.87	0.85	0.86	24783.0

Since we aim to compare the embeddings only we reran the source code of Davidson with just the vectorizers on the tweet and the pos-tags. This yielded a model with overall precision of 0.87, recall of 0.85, f1-score of 0.86 and a precision of 0.4, a recall of 0.53 and a f1-score of 0.46 on the hate-class.

The confusion matricies can be seen in Figure 1 and Figure 2.

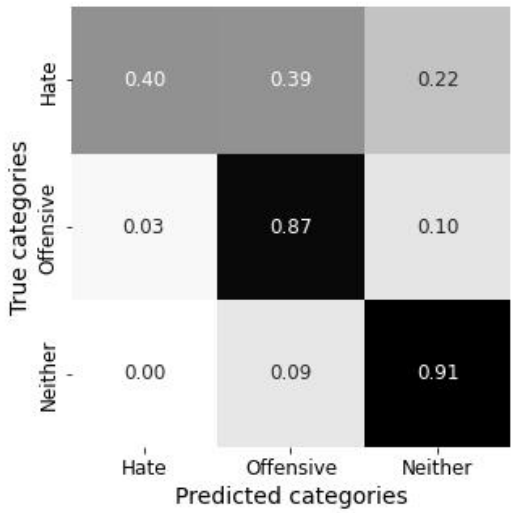

Figure 1: Confusion of our version of Davidson's model with features

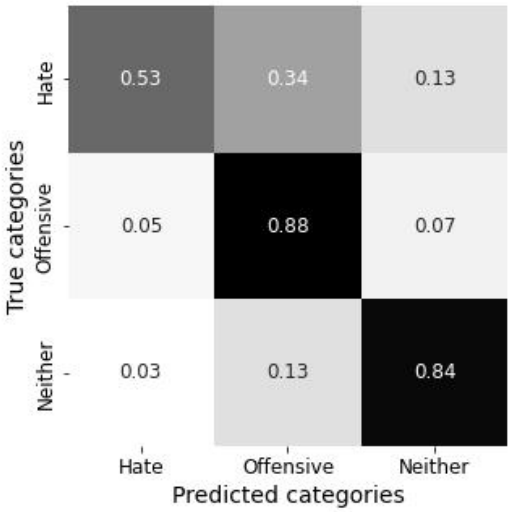

Figure 2: Confusion of our version of Davidson's model with no features

2 The Transformer Model and Implementation

2.1 The preprocessing

For preprocessing of the Tweets we used the **spaCy en_core_web_sm** pipeline with an **EntityRuler** component added. This uses identical RegEx to the preprocessing in [Davidson et al., 2017] to identify urls, mentions and excess whitespace. Then the entire tweet is lemmatized via the **Lemmatizer** component in the **en_core_web_sm** pipeline and urls are replaced with a "[URL]" token, as well as mentions being replaced by a "[MENTION]" token. The entire procedure is handeled in the function **preprocess_tweet** in the **data_pipeline** notebook. Here some possible improvements for the future could be made by also having a dedicated "[HASHTAG]" token as well as dealing better with elongated or slightly modified versions of the same tokens. For example: turning "Shuuuuuuuuut upppp" or "$hut up" into "shut up".

2.2 The Model

For the model we used a **distil-bert-uncased** pretrained Distilbert model with a classification head for finetuning using the **Huggingface AutoModelForSequenceClassification** class. In addition we used the appropriate BERT-WordPiece tokenizer adding two custom tokens ([URL],[MENTION]) to the tokenizer. The model gets called by the **Trainer** class via a **model_init** method, since otherwise the classification head weights are not connected to the

seed parameter of the **Trainer** class. This turns out to be a great problem to get reproducable results. Especially in our experiments, this turned out to be a problem. Since the best model we trained was not able to be replicated, with the set seed. We simply got lucky with the initial classification head weights. Which is exactly why we now use a **model_init** method. This is also why it shouldn't be expected to get the exact weights we got once the training procedure is followed. The Pytorch model is provided completely in the repository to make the results reproducable.

2.3 Finetuning

We used a custom lossfunction to mediate class-imbalances. For this we subclassed the **Trainer** class and overwrote the **compute_loss** function with a custom function. We use **Categorical-CrossEntropy** with weights for each class given by:

$$w_i = 1 - \frac{count(class_i)}{count(data)}$$

For training we used the following Hyperparameters: First we finetuned with:

- learning_rate = 1e-5
- per_device_train_batch_size = 16
- per_device_test_batch_size = 16
- num_train_epochs = 5
- weight_decay=0.01
- evalution_strategy = "steps"
- eval_steps = 500
- fp16 = True
- load_best_model_at_end = True
- metric_for_best_model = eval_hatef1
- seed = 42

Then we finetuned another pass with changes only in:

- learning_rate = 5e-6
- num_train_epochs = 2

the rest of the unspecified hyperparameters were the default value or are paths/logging settings which aren't training relevant and can be looked up in the **train_model** notebook.

Table 3: Classification report for our finetuned model without external features

class	precision	recall	f1-score	support
hateful	0.56	0.71	0.63	1430.0
offensive	0.97	0.95	0.96	19190.0
neither	0.94	0.97	0.95	4163.0
metric	precision	recall	f1-score	support
accuracy	nan	nan	0.94	24783.0
macro	0.83	0.88	0.85	24783.0
weighted	0.94	0.94	0.94	24783.0

2.4 Results

This yielded the following results using **sklearn**'s **classification_report**.

Table 3 shows that one can expect a relatively basic transformer model to outperform the baseline proposed by [Davidson et al., 2017]. The most significant improvement on the hate-class, that is pretty stable, under multiple training runs (different seeds and random classifier head weights) is the precision. It is not uncommon to see with finetuning for a few epochs a precision of 0.5 and a recall of 0.5 on the hate-class of the testset. This still is a 0.06 improvement in precision over Davidson's reported numbers. Here one starts to see a form of precision/recall trade-off emerge when training. Where one needs some trial and error to get a model for which $recall + precision > 1$ holds since one seems to trade recall for precision in the hate-class very evenly.

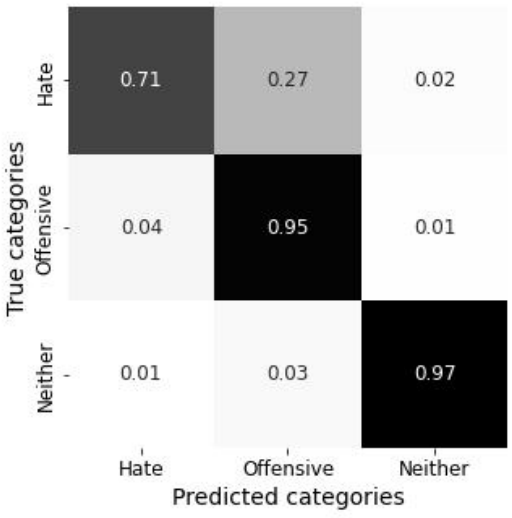

Figure 3: Confusion of the finetuned model

3 Evaluation

3.1 Comparison to [Davidson et al., 2017]

Comparing Table 3 to the reported numbers in Davidson's paper we can clearly see the overall performance increase.

metric	our model	Davidson
precision-avg	0.95	0.91
recall-avg	0.94	0.90
f1-score-avg	0.95	0.90

Particularly interesting is the improvement on the hate-class of the transformer model with tuned hyperparameters. The finetuned transformer model outperforms the baseline by Davidson reliably.

metric	our model	Davidson
hate-precision	0.58	0.44
hate-recall	0.75	0.61
hate-f1-score	0.66	0.53

3.2 Confusion and Maximum loss

We first begin to investigate some confusions of the model, for this purpose we use the **get_top_loss** function and investigate the examples which caused the biggest contributions to the loss. A typical error of the model is to overfit on the frequency of slurs like "bitch","hoe" and similar. These get classified almost universally as offensive independend of the context they were used. A few examples of a tweets which were classified as offensive but are actually part of the neither class would be

Examples of confusing neither with offensive class:

"I don't know what you been told but I love them project hoes"
"No girls no hoes just me myself and I"
"Up early then a bitch driving to denton omg can I move already"
"Frustration can be a bitch !"

This example illustrates the overfitting well:

"School Schoo Scho Sch Sc S St Sta Stay Stay h Stay ho Stay hom Stay home"

in this tweet the subword "ho" is so close to "hoe" that the model immediately thinks the tweet is offensive. This can be seen in the saliency scores in Figure 4. The same overfitting on some special token frequency also leads to confusion between the hateful and offensive class. While confusion between the neither and hateful class occurs often in two types. The first type of error is not recognizing directed offensiveness versus undirected offensiveness. While the second type of error is not seeing masked offensiveness. Which often happens when user use slurs that don't have a high enough frequency to be recognized as such or where everyday objects get used creatively to create offensive context.

Examples of confusions in the hateful-class The tweets below were classified as belonging to the neither-class but belong to the hateful-class.

"Twice divorced teabagger Florida Attorney General says marriage is reserved for 'stable, enduring family units' http://t.co/9ZWr3O33iu #wtf"
"@Flow935 jus wanted to let yáll know hope the hurricane kills your soca sunday trash #LORDWILLIN"

Both don't feature obvious offensive words and really become hateful by directing offensiveness towards someone personally, these are good examples of the first type of error. While these Tweets, which also belong to the hateful-class and were classified as part of the neither-class, showcase the second type of error.

"stuffed her like an Oreo http://t.co/mPAjkWJbZp"
"Porch monkey 4 Life"
"You are a jiggaboo...!"

The first type of error also tends to mix with the overfitting on sexists-slurs as seen in the following tweets which were classified as offensive but which belong to the hateful-class since they direct their offensiveness towards someone.

" @Fit4LifeMike @chanelisabeth hoe don't make me put up screenshots of your texts to me hoe "
"@GlitteredInPink hoe don't tweet me"
"RT @Slanguage: Stop flattering yourself, bitch. The only fan you have is on your ceiling."

Every one of these is a use of "hoe" or "bitch" directed at someone. These types of errors make up a big portion of the 100 examples with the highest loss in the entire dataset.

3.3 What did the model learn?

The investigation of errors yields also some evidence of the model actually learning, instead of only fitting some frequency correlations. This evidence is seen in some of the maximum loss examples, which are just plain mislabeled, but get labeled correctly by the model.

Examples of tweets that belong in the neither-class:

"Bae made me deep fried Oreos 😋"
"@VinceSchilling Vince, what is the general feeling of the natives that you know in regards to the term "redskins"? Offensive to them or not?"
"wishing I could high five whoever named a species of birds "great tits""
"RT @BALTsneakerShow: Water is wet. RT @202SOLE: Nike DC is trash."
"Chocolate cake is so trash."

Examples of tweets that belong in the offensive-class:

"You wanna find hoes on here? Just follow the chicks who reply with heart eyes under sex gifs"
"@vh1ADENECA @260chocolate @4Sunshinejones1 aye, I respect ya hustle and all but dont tweet me anymore trash ass videos...thank u kindly"

Examples of tweets that belong in the hateful-class:

"Looks like a tool but is useful as a clay hammer #fag @hOPPondis http://t.co/K7wqlAWRwM"

This is not yet complete evidence, but already provides some confidence that the model actually learned some reasonable structure from the ground up. The model definitely learned about racist remarks a few of the highest confidence examples in the hate-class revolve around homophobic or racist language. As one can see in the saliencey scores using the **multiclass_explainer** in the **evaluate_model** notebook. example outputs with high confidence are shown in Figure 5, as one can see in these examples the model derives almost all confidence out of racist-slurs.

4 Conclusion

A Reasonable Baseline for Hatespeech: So we are now able to confidently answer our research question. Yes we should put forward a different baseline performance for transformer models on this dataset. Based on our model and the possibilities for improving the performance we propose a baseline to be a recall of 0.65 and precision of 0.55 on the hate-class and at least an f1-score of 0.93 overall. This is a reasonable performance to expect of a transformer model with finetuning and hyperparameter tuning. But a better optimized architecture, a better classification head that takes features, better hyperparameter tuning or other smart tricks should be able to beat those scores reliably. Which is why it should provide a good baseline,filtering ideas that just get beaten by spending more GPU hours on parameter tuning and finding good initial weights. We didn't implement a real hyperparameter optimization. On a faster GPU, it is possible to get a much deeper search into the parameter space and possibly push these results by a good bit. A good goal to strive towards would be a recall of 0.75 and a precision of 0.6, but it is unclear if the training process for such a model, would be simple enough to really call it a baseline anymore.

Relevance for Industry purposes: This is obviously a rather important improvement for industrial purposes since moderators for social media are expensive. With social media really only getting bigger at the moment it will become harder and harder to reinforce rules. An 0.14 increase in precision over the old baseline is a big deal, when one realizes that every day $500 \cdot 10^6$ Tweets are sent. Even with 1 in every 10000 Tweets being hateful, that would be 50000 Tweets per day that need to be correctly detected and removed. The improvement in precision means that 7000 tweets more get classified correctly and do not need human intervention. To put this in perspective even if a very skilled moderator would need 3 seconds per tweet, that is $\frac{3 \cdot 7000}{60^2} \approx 5.83$ hours of work every day without a pause or slowing down from fatigue. At a salary of 20$ per hour that is 42559$ every year in labor (without calculating all the cost of human capital). That is a descent buffer in capital to pay for the computation hours needed to evaluate 500 million tweets.

A Figures

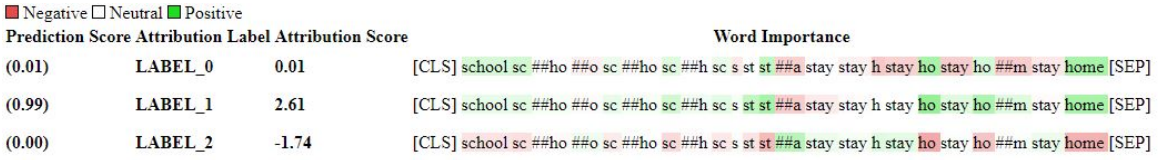

Figure 4: saliencey scores for the ”school stay home” example

Figure 5: The model gets strong signal from racist remarks

References

T. Davidson, D. Warmsley, M. W. Macy, and I. Weber. Automated hate speech detection and the problem of offensive language. *CoRR*, abs/1703.04009, 2017. URL http://arxiv.org/abs/1703.04009.

J. Devlin, M. Chang, K. Lee, and K. Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018. URL http://arxiv.org/abs/1810.04805.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017. URL http://arxiv.org/abs/1706.03762.