Bitter on Twitter: An Offensive Tweet Classification Task

Mareta Masaeva
University of Antwerp

Mohammadreza Barati
University of Antwerp

Shima Rahimi
University of Antwerp

Abstract

In this paper, we present two possible classification systems for the identification of offensive language on the Internet. In particular, we define one non-neural and one neural model that identify offensive language in four datasets, one of which is the official OffensEval tweet dataset of the SemEval 2019 competition. Compared to the baselines, only our classic model reaches reasonable accuracy and macro-averaged F1 scores. Our deep-learning model has trouble competing with the baselines. This difficulty is clarified by the precision and recall scores, which show that both our classic and deep-learning models are unable to correctly deal with the class imbalance in the dataset of tweets.

1 Introduction

On the Internet, and more specifically on social media, one of the largest problems is the amount of offensive language posted every day, by people all around the world. Avoiding severely offensive language when scrolling through websites like Twitter is, unfortunately, almost impossible. Manually filtering out these posts can be quite stress inducing, and even cause post-traumatic stress disorders. Automating the process, then, is the ideal solution for this type of text classification problem. In this paper, we attempt to find suitable methods of text classification in order to automatically identify offensive language on the Internet.

One way computational linguists have attempted to find solutions for this problem is by means of the SemEval 2019 competition. In this competition, the goal was to identify and categorise offensive language on social media, particularly on Twitter. One specific task of this competition was task 6, which consisted of three subtasks: to create a text classification algorithm that finds out a) whether content is offensive or not, b) if the content, if offensive, is targeted or not and c) if it is targeted, what kind of group does it aim at? More specifically, we were assigned to find a solution only the first subtask of this competition. To find a solution for this task, we were asked to use the OLID dataset, or the Offensive Language Identification Dataset (Zampieri et al., 2019). Additionally, we tested our models on two out-domain datasets, as well as a dataset from the Textgain startup company. The data will be described in more detail in Section 3.2.

For this task, we were to build two different models; the first model was to be classic one, using no deep-learning techniques. Traditional NLP tools that rely on deep-learning techniques under the hood, however, were allowed. We will describe our classic model in more detail in Section 3.1.1.

Our second model was to be a model that relies on deep-learning techniques. During the building of our final model, we experimented with many different architectures and techniques, which led us to submitting a different model for the class competition than our final model reported on in this paper. Both of these models will be described in detail in Section 3.1.2. The results that were output by our models will be shared and discussed in Section 3.4.2 and Section 4. The code of this project can be found at github.com/m-barati/NLP_Shared_Task.

2 Literature

A commonly used traditional technique in NLP text classification is SpaCy Bag of Words, which is a classification in the conventional machine learning sense. Examples that SpaCy BoW is used for

include spam detection, sentiment analysis, and tagging customer queries. SpaCy handles the bag of words conversion and builds a simple linear model with the TextCategorizer class. [1]

Oberstrass and his colleagues (Oberstrass et al., 2019) in their research presented that context embeddings were important features for the three different subtasks in connection with classical machine learning models and with deep learning, using SpaCy BoW. They transformed the tweet texts into a TF-IDF weighted bag of words. For the Named Entities recognition part in a tweet (indicating whether the tweet was person-related or related to, for example, religious or political groups or organisations) they used SpaCy's named entity recognition. By evaluating their model with OffensEval datasets, they achieved a macro-averaged F1-score of .70, while most frequent traditional classification methods resulted in macro-averaged F1-scores of .28 with these mentioned datasets.

The deep-learning model architecture presented by Devlin et al. (2019) is a system based on pre-trained language models, and have reached new state-of-the-art results. This BERT-based bidirectional transformer model (Devlin et al., 2019) is designed to bidirectionally pre-train deep representations from unlabeled text by jointly conditioning on both left and right context in all layers. As a result, this pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as text labelling, without substantial task specific architecture modifications. One issue with these pre-trained models is that the training language variety makes them well suited for general-purpose language understanding tasks, and it highlights their limits with more domain-specific language varieties. Another state-of-the art model for this type of text classification is HateBERT (Caselli et al., 2021). Caselli et al. showed, with their portability experiment, that the behaviour of HateBERT can be explained by accounting for the differences in specificity across the abusive language phenomena. Their key contributions were: a) additional evidence that further pre-training is a viable strategy to obtain domain-specific or language variety-oriented models in a fast and cheap way, b) the release of HateBERT, a pre-trained BERT for abusive language phenomena, intended to boost research in this area and lastly c) the release of a large-scale dataset of social media posts in English from communities banned for being offensive, abusive, or hateful. They evaluated both the robustness and the portability of HateBERT by imbalancing all datasets between positive and negative classes, and also by targeting phenomena that vary along the specificity dimension. By evaluating their approach with OffensEval datasets, their results showed that HateBERT, with a macro-averaged F1-score of .82, ensures slightly better portability than a generic BERT model with a macro-averaged F1-score of .81, especially when going from generic abusive language phenomena (i.e., offensive language) towards more specific one (i.e., abusive language or hate speech). This behaviour was due to the differences across the annotated phenomena. They also claim that HateBERT consistently obtains better representations of the targeted phenomena.

[1] https://spacy.io/api/textcategorizer.

3 Models

In this section, we will provide system descriptions of both our classic and our deep learning model. Further on, we will describe the data used in this research and the reasons why we applied certain methods, as well as briefly report on the baseline models for this task.

3.1 System description

3.1.1 Model without deep-learning techniques

The model that we built without deep-learning techniques is an ensemble model based on a VotingClassifier, which determines the optimal combination of classification methods for our binary classification problem. The considered methods for the classification were Logistic Regression, Bernouilli Naïve Bayes, Multinomial Naïve Bayes, and two types of XGB Classifiers (one with a linear and another with a tree booster). The model deemed optimal for our problem was the Multinomial Naïve Bayes, however, we decided to use the VotingClassifier anyways since it is a more advanced method. Because we used soft voting for the Voting Classifier, weights were needed as well; the optimal weight for this problem needed to be found manually, as using a specific weight could very quickly overfit on the inoffensive class. The ideal weight for the soft voting turned out to be X.

The eventual pipeline consisted of a FeatureUnion of the CountVectorizer and TfidfVectorizer and the VotingClassifier. The two vectorizers were applied to the lemmas and POS tags by means of a ColumnTransformer, and then the results of those were concatenated using the FeatureUnion. Lastly, a GridSearch was implemented with a cross validation of 5. As input for the model, we used both the lemmas output by the preprocessor and POS-tags of these lemmas. Reasons why we used these techniques will be explained in Section 3.3.2.

3.1.2 Deep-learning model

For our first deep-learning implementation of the offensive tweet classification, which we submitted to the class competition, a Keras Sequential model was used as a basis. The stack of layer consisted of a simple Keras embedding layer, SimpleRNN layer, a global max pooling layer, a regular Dense layer with relu activation, a dropout layer and a Dense output layer.

For the loss function, the Adam optimizer and sparse categorical cross entropy were used as we worked with integer labels. We fitted the model for 10 epochs, and the output file with results was saved in a separate results folder. For these results, the labels were re-converted to strings NOT and OFF.

Our final model architecture reported on in this paper, which did not make it to the class competition, is a Tensorflow Sequential model that consists of GloVe word vectors as a first embedding layer. The GloVe word vectors used in this system were pre-trained on Twitter, more specifically on a database of two billion tweets, consisting of 27 billion tokens. The vocabulary for this database consists of 1.2 million words. In this system, we used the GloVe 100-dimensional word vectors to capture as much information as possible while keeping computational complexity at a manageable level. The next part in our stack of layers is a bidirectional LSTM layer; the reason behind using this type of layer will be discussed in the Methods section. The output of this bidirectional LSTM layer then flows to a global max pooling layer and BatchNormalization layer, which normalized the inputs. Three pairs of Dropout and Dense layers follow, with two times a relu and one time a sigmoid activation function for the Dense layers.

For this model, we used a Keras Tuner to find the optimal hyperparameters. The Keras Tuner determined the optimal number of units for the first densely-connected layer to be 160, and the optimal learning rate for the Adam optimizer to be .001. Reasons why we used the Keras Tuner will be given in Section 3.3.3.

3.2 Data

The models were tested on a number of different datasets. The first dataset is the official OffensEval data, which is an in-domain test set. Two other, out-domain datasets were used as well, the contents of which we were kept unaware of until after completion of the class competition. These two datasets turned out to contain text data from Reddit as well as from Wikipedia. Lastly, a 'mystery' dataset was used, which originated from the language technology startup company Textgain. Here, we will discuss mainly the in-domain dataset from OffensEval, and briefly the other three datasets.

3.2.1 In-domain dataset

The dataset used in task 6 of SemEval-2019 is OLID, which is a collection of about 14,000 English tweets. These tweets are annotated for offensive language use, based on three levels of offensiveness. These levels are encompassed by the annotation model used for the tweets, the first of which is the focus of our work.

SemEval-2019 task 6 focused on identifying and categorising offensive language on social media, more particularly on the Twitter platform. This task involved three different sub-tasks. In our work, we only cover sub-task A, but all three will be briefly summarised here.

- **Sub-task A** is offensive language identification. For this sub-task, tweets were labelled as either offensive OFF or not offensive NOT. This is the sub-task we focussed on in our work. Table 1 shows examples of this identification. A tweet is classified as offensive when it contains profane language or targeted offences, which may or may not be direct.

- **Sub-task B** is the categorization of offensive tweets as either targeted TIN or untargeted UNT tweets.

- **Sub-task C** is the identification of targets in offensive, targeted tweets. These tweets are labelled as either individual IND, group GRP or others OTH.

The data used consists of 14,100 annotated tweets that are labelled for these three subtasks. This data was collected using the Twitter API; Zampieri et al. (2019) searched for keywords that are often prevalent in offensive language on social media. These keywords contained political words like 'gun control' and 'MAGA', as well as other keywords determined by six experts. The proportion of offensive tweets in the dataset was around 30 %, so that the dataset accurately represents the number of actual offensive tweets on Twitter.

The data was annotated using crowdsourcing. For each tweet, two annotations by two annotators were obtained and when there was annotator disagreement, a third annotation was requested. This was done for all three of the sub-tasks. For about 40 % of the tweets, a third annotation was needed. One thing that should be noted is the fact that getting sufficient tweets annotated for each class was challenging, especially for sub-tasks B and C. These crowdsourced annotations were not corrected afterwards. In the original tweets themselves, the mentions of Twitter users were substituted by '@USER' and URLs were substituted by 'URL'.

3.2.2 Out-domain and Textgain datasets

The first out-domain dataset is a collection of offensive English language from the online forum Reddit, and is named Ruddit. It contains 6000 IDs from comments that are scored on an offensiveness spectrum between -1 (not offensive) and 1 (very offensive). It does not give the body of the comments, as this is against GDPR regulations, however, the comments can be found via the Reddit API using the IDs provided.

The second out-domain dataset contains Wikipedia text data. It contains linguistic data that is either labeled as offensive or not offensive. Originally, this dataset contained a wider spectrum of offensive language (non-offensive, moderately offensive, of severely offensive language), however, the moderately offensive data was removed to widen the distinction between these datapoints.

Lastly, the Textgain dataset contains English tweets about football, and were labelled manually either as offensive or not offensive.

3.3 Methods

In this section of the paper, we will describe our pre-processing pipeline and the reasoning behind why we used specific techniques in our model architectures.

3.3.1 Pre-processing

Pre-processing of the data consisted of, firstly, removing certain noise in the tweets: URLs, '@USER'- and '&amp'-tokens were deleted, as well as most punctuation. All punctuation except apostrophes (as in contracted verb forms like 'you're' and 'he'll'), hashes and double or more exclamation marks, question marks and periods, were extracted and deleted. Hashes remain in the data because they are very often used in tweets and can be tokens that convey certain relationships between tweets that otherwise might seem unrelated. Hashtags are especially prevalent in tweets about political matters, which appear to often contain offensive language (Zampieri et al., 2019). Double or more exclamation or question marks, as well as double or more periods, were kept in the data as well because these are frequently used to express sarcasm or different kinds of offence.

Furthermore, all digits and stopwords found in the default NLTK stopwords list were removed from the tweets. To convert classic emoticons to tokens that can be processed, regular expressions were used, while the more commonly used emojis were converted to text tokens using Python's emoji module. Lastly, all tweets were tokenized and lemmatized. For the deep-learning model, the tweets were pre-padded as well.

Other techniques that were experimented with for the pre-processing pipeline, but eventually omitted, include a spelling correction function and a determiner of offensiveness level using a list of offensive words. The spelling corrector used the SymSpell algorithm, but it was omitted because it wrongly edited words that were correct, or misinterpreted abbreviations as wrongly spelled words. It also by default removed hashes. For these reasons we decided not to include a spelling corrector in the pre-processing pipeline. It was also decided not to use the offensive word list in the pre-processing because the results using this feature were no different from the results using a simple TF-IDF vectorizer.

For both models, with and without deep-learning techniques, lemmas were used to perform the classification on. We decided to use lemmatization over stemming because it often results in better accuracy in NLP tasks. Lemmatization is a slower and more computationally expensive

method, but as it produces real dictionary words, we considered it more trustworthy than its alternative of stemming. Also, as lemmatization requires a word's POS tag to determine the lemma, we automatically obtained another feature to make use of and feed to our classical classifier. This, we believed, could possibly improve our results.

3.3.2 Model without deep-learning techniques

Instead of using the CountVectorizer and TfidfVectorizer separately, which would have been very time consuming when tuning hyperparameters, we decided to use a FeatureUnion. This FeatureUnion concatenates the results of our two vectorizers and makes using GridSearch later on more efficient, since we do not have to run it on two transformers separately. The FeatureUnion takes a list of transformer objects and applies them in parallel, combining them into a new transformer that concatenates the two outputs. In the end, this results in larger vectors, and more elegant code. After the concatenation, the data is fed to the classifier.

We decided to include the XGBoost classifiers in our Voting Classifier for the reason that this is a currently very popular algorithm in the Machine Learning Community, and regularly outperforms other algorithms in NLP and other machine learning tasks. The XGBClassifier Tree solver appears to be the best solver for text classification, which is our reason for implementing it along with the Linear solver.

3.3.3 Deep-learning model

For our deep-learning model, many different kinds of model architectures were experimented with; a RoBERTa transformer model, a Recurrent Neural Network based model combined with Keras embeddings, and a Pytorch Long Short-Term Memory network. Eventually, these models were discarded, as we were unable to integrate these models with GloVe embeddings, which we believed would be crucial to achieve high scores. Eventually, a final deep-learning model was designed. This model, however, was not used for our submission for the class competition, as it was designed after the competition deadline.

The deep-learning model that we submitted for the class competition was built with the following architecture. A Recurrent Neural Network was used as the basis, because it is a widely used architecture in NLP and provides quite accurate classification results in language tasks. Combining the RNN with global max pooling, we achieve a model that is able to classify offensive tweets without having to regard the position of these possibly offensive words; matches are aggregated globally. This results in a model that is able to interpret contexts of words and offensive language due to its recurrent nature and find offensive phrases and patterns in each input. Together, they form a robust algorithm for offensive language identification.

Our eventual deep-learning model, which outputs our final results, is a Tensorflow Sequential model. This model was built a model builder function. Via this model builder function, we were able to use the Keras Tuner, which allowed us to tune hyperparameters while initially training the model, instead of afterwards. We used this Tuner in our system because it makes finding the optimal hyperparameters more accessible and simple, compared to a basic GridSearch. The limits of GPU usage in Google Colab are quite easy to reach, especially when running a GridSearch for many epochs. Using the Keras Tuner made this process much quicker and, we believe, more efficient.

Different word embeddings can play a large role in correctly representing words and their meanings in the right context, since context often carries rich semantic information about the words used. Using GloVe word vectors pre-trained on a large corpus of Tweets specifically, we hoped to represent the meanings of the tweets in our dataset as accurately as possible. GloVe also assigns lower weights to highly frequent words such as function words, which is especially useful in the case of Tweets, where function words are often omitted because of the limited tweet length of 280 characters. Even though function words might have become highly infrequent in tweets because of the character limits, the fact that GloVe still assigns a lower weight to those infrequent words can still be advantageous when working with Twitter data.

A bidirectional LSTM layer was used because, in tasks like sentiment analysis, this type of layer appears to perform better than a regular unidirectional LSTM layer. This is because, unlike in a standard LSTM, the input floats both ways, which allows the system to use the information from both

sides of the pipeline. The outputs of these two layers can then be combined. Because of this combination of outputs, bidirectional LSTMs will return different outputs for every component of every sequence. This makes them useful in many NLP tasks, which is why they were used in the current task as well.

3.4 Experiments

In this section, we will briefly discuss the baseline models for this text classification task, as well as share the results output by our final models.

3.4.1 Evaluation

In this OffensEval project, the most frequent class model and Spacy bag-of-words model are considered baseline non-neural models. Neural model baselines are Regular Bert and HateBERT (Caselli et al. 2020). Macro-averaged F1-scores for all four of these models can be found in Table 1 and Table 2.

	Most frequent class	Spacy BOW
OffensEval	.28	.70
Reddit	.31	.66
Wikipedia	.33	.86
Textgain	.46	.54

Table 1: Non-neural model baselines

	Regular Bert	HateBERT
OffensEval	.81	.82
Reddit	.70	.68
Wikipedia	.90	.91
Textgain	.47	.48

Table 2: Neural model baselines

When evaluating our two models, the same evaluation metric was used, as well as the accuracy of the model. Along with these, we looked at the confusion matrices for all different datasets. The results of both our models on all the datasets will be discussed below.

3.4.2 Results

Model without deep-learning techniques

All results from the classic model can be found in Table 3 below, as well as confusion matrices for the four datasets.

When it comes to accuracy and the macro-averaged F1-scores, the second out-domain dataset, Wikipedia data, returned the best results; a .82 score for both of these metrics. This probably has to do with the fact that the divide between offensive and non-offensive language in this dataset was made very large by the instructors of the task. The accuracy and macro-averaged F1-scores for the Textgain dataset are the lowest of all, barely reaching a .50 F1. The dataset of tweets from OffensEval got .77 accuracy and .70 macro-averaged F1, which, to the casual eye, seem to be decent scores.

		OFE	RED	WIKI	TG
Precision	NOT	.82	.64	.82	.89
	OFF	.60	.60	.82	.20
Recall	NOT	.87	.73	.82	.63
	OFF	.52	.50	.82	.54
Macro F1		.70	.61	.82	.51
Accuracy		.77	.63	.82	.61

Table 3: Results classic model

For all but one of the datasets, both precision and recall are quite off-balance. The classifier returns a higher precision and recall for the inoffensive tweets, and even has trouble reaching a higher than .50 recall for the OffensEval, Reddit and Textgain datasets. The OFF precision for the Textgain dataset reaches only .20, which is especially low. In general, precision and recall are consistently higher for inoffensive language in all but the Wikipedia data, where all scores align at 82 %.

For these four datasets, confusion matrices output by the non-neural model can be found below.

		Predicted	
		NOT	OFF
True	NOT	538	82
	OFF	115	125

Table 4: Confusion matrix OffensEval dataset (non-neural model)

		Predicted	
		NOT	OFF
True	NOT	487	117
	OFF	274	269

Table 5: Confusion matrix Reddit dataset (non-neural model)

		Predicted	
		NOT	OFF
True	NOT	491	109
	OFF	106	494

Table 6: Confusion matrix Wikipedia dataset (non-neural model)

		Predicted	
		NOT	OFF
True	NOT	681	407
	OFF	87	101

Table 7: Confusion matrix Textgain dataset (non-neural model)

Deep-learning model

All results from our deep-learning model can be found in Table 8 below, as well as confusion matrices for the four datasets.

		OFE	RED	WIKI	TG
Precision	NOT	.73	.57	.43	.84
	OFF	.31	.49	.37	.11
Recall	NOT	.75	.76	.55	.72
	OFF	.30	.29	.26	.21
Macro F1		.52	.51	.39	.46
Accuracy		.62	.55	.41	.65

Table 8: Results neural model

When looking at the macro-averaged F1-scores and accuracy scores output by the deep-learning model, we can see that the OffensEval dataset reaches the highest F1-score, however, the Textgain dataset reaches the highest accuracy. With an accuracy of 65 %, it appears that the Textgain tweets about football were classified as either offensive or inoffensive in the best way. However, again, the precision and recall scores reveal that there is a large imbalance between tweets correctly classified as offensive and tweets correctly classified as inoffensive. Only 21 % of offensive tweets were correctly identified by the neural model, and merely 11 % of tweets classified as offensive were correct.

When we look at the scores for the in-domain and out-domain datasets, the same pattern can be found. Again, the precision and recall for the binary labels are quite off-balance. Only for the Wikipedia dataset, the imbalance is not that large compared to the others; a .06 difference in precision and .29 difference in recall. Possible reasoning why we believe these results were found will be discussed below.

Confusion matrices output by our deep-learning model can be found in the following four tables.

		Predicted	
		NOT	OFF
True	NOT	464	156
	OFF	169	71

Table 9: Confusion matrix OffensEval dataset (deep-learning model)

		Predicted	
		NOT	OFF
True	NOT	503	101
	OFF	386	157

Table 10: Confusion matrix Reddit dataset (deep-learning model)

		Predicted	
		NOT	OFF
True	NOT	331	269
	OFF	442	158

Table 11: Confusion matrix Wikipedia dataset (deep-learning model)

		Predicted	
		NOT	OFF
True	NOT	785	303
	OFF	149	39

Table 12: Confusion matrix Textgain dataset (deep-learning model)

4 Discussion

In this section, we will examine and discuss the classification results output by our non-neural and neural model. We will attempt to identify possible faults that were made during the constructing of the two models, and investigate in which parts of the models possible improvements could could be made, in order to achieve a more accurate classification.

4.1 Model without deep-learning techniques

As the results of our classic model in Table 3 show, there is somewhat of an unbalance between

the amount of correctly classified offensive and the correctly classified inoffensive tweets. This shows itself especially in the recall of the OffensEval dataset, the one most important to this type of task; as it appears, our classic model was able to correctly identify only 52 % of offensive tweets in the dataset. This imbalance most likely has to do with the fact that the OffensEval dataset contains unbalanced data; only around 30 % of the tweets are classified as offensive (Zampieri et al., 2019). Our classic model, as it appears, is unable to accurately manage imbalanced data. The satisfactory accuracy that it returns, 77 %, is thus a misleading score.

Lowest results were returned for the Textgain dataset. The precision scores in Table 3 show that, again, our model misclassified many of the non-offensive tweets as offensive; here, the OFF label obtains only 20 % precision, while almost 90 % of tweets classified as inoffensive by our classic model were correct. Of the total of inoffensive tweets, however, only 63 % were correctly identified, as the recall for this dataset shows. This again probably is a result of imbalance of the tweets. Another matter that might have had negative impact on the classification is the fact that a Naïve Bayes classifier is, as the name implies, naïve; a large disadvantage of this otherwise excellent classification system is that it assumes independency of features. In real life, as well as on the Internet when it comes to tweets, this independence assumption does not hold. For linguistic data especially, the semantics and pragmatics terms and words are very much dependent of one another, which is why integrating a Naïve Bayes classifier might not have been the most ideal choice for such a task. Along with the class imbalance, these these two things might have been shortcomings of our non-neural classification system.

Compared to the baseline models for this task, our classic model performs reasonably well. It only slightly underperforms for the out-domain datasets, and reaches the same macro-averaged F1-score of 70 % for the in-domain dataset, which is pleasing.

4.2 Deep-learning model

As was the case with our non-neural model, our neural model as well shows symptoms of not being able to handle imbalance in the class distribution. As the confusion matrices for all four datasets show, the number of true positives for the NOT label is in every case a lot larger than the number of true positives for the OFF label, which confirms this reasoning. The smaller difference in precision and recall for the Wikipedia dataset results, which can be found in Table 8, is again probably caused by the large distinction between offensive and inoffensive language in this data. The large distinction, however, does not completely eradicate the imbalanced class distribution.

Compared to the baseline models for this task, Regular Bert and HateBERT, the macro-averaged F1-scores output by our deep-learning model are considerably worse. Where the baseline models reach 90 % and 91 % for the Wikipedia dataset, our neural model reaches only 39 %. Only for the Textgain dataset, the difference is but slight; the baselines reach 47 % and 48 %, while ours reach 46 %. When we compare our model to the baselines then, or even analyze it on itself, we see that there are many improvements that should be made in possible future versions. Using transformers such as Bert or RoBERTa and combining them with GloVe word vectors might be more suitable for this task.

5 Conclusion

By interpreting the classification results output by the non-neural and neural models we created, a number of conclusions can be drawn. First of all, approaching and discussing the problem of class imbalance in our dataset might have positively impacted the results of both our models. There were no real precautions taken against this issue, which in retrospect appears to us a grave mistake. Nevertheless, we did achieve classification results that are fit against the non-neural baseline models for this task. Unfortunately, we were unable to compete with the deep-learning baseline models. however, we believe that experimenting with these advanced systems and learning new things is more valuable to students such as ourselves. We hope that, in the future, we will be able to design model architectures that are up to par with the baselines in this field, and learn even more about these complex structures.

6 References

Caselli, T., Basile, V., Mitrović, J., and Granitzer, M. 2021. HateBERT: Retraining BERT for Abusive Language Detection in English.

In Proceedings of the 5th Workshop on Online Abuse and Harms (WOAH 2021). Association for Computational Linguistics.

Devlin, J., Chang M. W., Lee, K., and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies.* Association for Computational Linguistics.

Oberstrass, A., Romberg, J., Stoll, A., and Conrad, S. 2019. HHU at SemEval-2019 Task 6: Context Does Matter - Tackling Offensive Language Identification and Categorization with ELMo. *Proceedings of the 13th International Workshop on Semantic Evaluation.* Association for Computational Linguistics.

Zampieri, M., Malmasi, S., Nakov, P., Rosenthal, S., Farra, N., and Kumar, R. 2019. SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media (OffensEval). *Proceedings of the 13th International Workshop on Semantic Evaluation.* Association for Computational Linguistics.