SSL & NLP - Final Milestone

Ionescu Diana & Andrei Dumitrescu

1. Introduction

For the final milestone for the SSL project, we extended our solutions for the Abusive Language Detection in Social Media task. As a short recapitulation, during the previous milestones we decided on what datasets to use and also we implemented multiple baselines and one State-of-the-Art Model based on Bi-directional LSTMs with context attention. The datasets used were: Abusive Language on YouTube (*ALYT),* Toxic Comment Datasets (*TCC*) and Conv Abuse. Each of the datasets was representative of a concrete type of social media interaction context such as: YouTube comments, tweets and online conversations. During this last milestone we achieved using another State-of-the-Art Model, based on Transformers, called HateBERT, Moreover, we also combined this approach with the main idea behind the first State-of-the-Art and finally, we also experimented with transfer learning on the final model using 2 different datasets to see the impact that it has.

2. HateBERT Vanilla

The first experiments we have run for this milestone were using the classic *HateBERT* in combination with some simple Linear Layers. As a short reminder, *HateBERT* was a new flavour of *BERT* specially retrained to be biassed to abusive language. This was achieved by retraining the classic model on the *Masked Language Model* objective on a custom built dataset called *Reddit Abusive Language English dataset (RAL-E).*

For the previous milestones, most of the training loop was made using the TensorFlow API. During this milestone, we used PyTorch and the HuggingFace library. This decision was made because the only available *HateBERT* in the HuggingFace library was built for PyTorch. The input for the model in our experiments will be represented by the sequence of tokens obtained from the text from the sample. To obtain this sequence of tokens we used the tokenizer exposed through the HuggingFace library. So the input for the model will be of shape *(Batch Size, Number Of Tokens).* During the previous experiments runned in the last milestone, we have allowed a maximum number of tokens equal to 120. This time around, due to memory limitations of the GPU used to run the experiments, we have decided to use only 100 tokens. The embeddings produced by *HateBERT* will be of size 768. In our first approach, the output of *HateBERT* is flattened keeping the batch axis and then passed through 3 Linear Layers, using a dropout after every layer. The *HateBERT* model was fine-tuned during the experiments. The optimizer used was Adam, and the loss function was CrossEntropyLoss. The models were trained for 10 epochs, with a training batch size of 32. The learning rate was set to 1e-5. These hyper-parameters were kept along all the training experiments for all the 3 datasets. Additionally, in the previous milestone, we used a split 80-20 split of the training set in order to build a validation set. We kept the same strategy of splitting during these experiments for the datasets that didn't offer a validation set. To test the performance of the obtained models, we computed, as previously, the macro F1-Score and Accuracy score on the test sets of each dataset. The best results obtained are presented in the following table, and will also be presented in an overall comparison table between all models.

Table 1 - HateBERT Vanilla Results

	HateBERT Vanilla	
	Macro F1-Score	***Accuracy***
ALYT	0.3212	0.8447
ConvAbuse	0.8672	0.9366
TCC	0.5995	0.7662

3. HateBERT with LSTMs and custom attention mechanism

During the previous milestone, we presented a State-of-the-Art based on Bi-directional LSTMs which also introduced a custom attention mechanism called *context-attention*. Our experiments showed that this State-of-the-Art Model achieves better results than every baseline we trained on every dataset. Taking this into consideration, we decided that it would be interesting if we would combine the two models together to see what results would achieve the new model. In comparison with the previous experiment, we build on top of *HateBERT* the *Bi-directional LSTM layers,* followed by the attention mechanism and 2 different Linear Layers. Inside this model, different dropouts are introduced. The hyperparameters used for training this model are the ones mentioned in the previous paragraphs. Similarly, we will present the score values in the following table and also in another one alongside all results.

Table 2 - HateBERT + Bi-LSTMs + Custom Attention Results

	HateBERT + Bi-LSTMs + Custom Attention	
	Macro F1-Score	***Accuracy***
ALYT	0.4173	0.8548
ConvAbuse	0.8668	0.9331
TCC	0.6094	0.7638

4. Overall results

In this paragraph we would like to present the final overall results obtained during our project development. In *Table 3* there are presented all the relevant results regarding the *Macro F1-Score* and in *Table 4* there are all the results regarding the *Accuracy Score*. The bolded results are the best ones obtained.

Table 3 - Macro F1-Score

	SVM Baseline	***Neural Network Baseline***	***Bi-directional LSTMs with self attention***	***HateBERT Vanilla***	***HateBERT + Bi-LSTMs + Custom Attention***
ALYT	0.41	0.47	**0.48**	0.32	0.41
ConvAbuse	0.67	0.78	0.82	**0.86**	**0.86**
TCC	**0.63**	**0.63**	0.60	0.59	0.60

Table 4 - Accuracy Score

	SVM Baseline	***Neural Network Baseline***	***Bi-directional LSTMs with self attention***	***HateBERT Vanilla***	***HateBERT + Bi-LSTMs + Custom Attention***
ALYT	0.87	0.88	**0.88**	0.84	0.85
ConvAbuse	0.88	0.89	0.90	**0.93**	**0.93**
TCC	0.71	0.70	**0.77**	0.76	0.76

As it can be seen from the results, neither of the 3 *State-Of-The-Art Models* can outperform the baselines on the *Macro F1-Score* on the *TCC* dataset. In terms of accuracy, all 3 of them achieve better results than the baselines. On the *ConvAbuse* dataset, the *HateBERT* based models perform similarly and outperform the rest of the models both in terms of *Macro F1-Score* and *Accuracy Score.* On the *ALYT* dataset, the *HateBERT* with *Bi-LSTMs* model performs better than the vanilla one, but they perform worse than the last State-of-the-Art. As it can be seen from the experiments, the 2 based *HateBERT* models don't perform much differently, the only significant difference being at the *ALYT* dataset on *F1-Score Metric.*

5. Transfer Learning

As a last experiment, we have decided to experiment a little bit with *transfer learning*. We wanted to see if a model already trained on a dataset could learn better on another one. In order to achieve this milestone, we decided to take the best trained *HateBERT Vanilla* on the *TCC* dataset and train it on the *ConvAbuse* dataset. We will test this model on how it performs on both testsets and also we will compare it with the performance of the model just trained on the *TCC* dataset and of a model just trained on *ConvAbuse*. These results can be seen in the following table.

Table 5 - Transfer Learning experiments

	HateBERT on ConvAbuse		***HateBERT on TCC***		***HateBERT on TCC fine tuned on ConvAbuse***	
	Macro F1-Score	***Accuracy***	***Macro F1-Score***	***Accuracy***	***Macro F1-Score***	***Accuracy***
TCC	0.43	0.43	0.60	0.76	0.57	0.66
ConvAbuse	0.86	0.93	0.76	0.86	0.85	0.92

As it can be seen from the above results, the *HateBERT vanilla model* outperforms the other 2 models on the opposite datasets, but performs a little bit worse on the datasets that the models were trained on. We consider that it is a good thing that the model trained on both datasets keeps performing well on all contexts, even if it doesn't perform better on the sets on which the models were trained on. We consider this as a good result because when developing a pipeline that would be used in production, it would be more cost-efficient in terms of memory and logic complexity to use a single model for more contexts than two or multiple ones.

6. *Conclusion*

As a conclusion, the *State-of-the-Art* models proposed in this research project proved to perform well on different social media contexts such as: YouTube comments, conversations or tweets. Also, it seems that it would be reliable to apply transfer learning on two different social media contexts, reducing the cost of development from using two different models for the same task to a single model. Additionally, all the code written for this project and how to run it can be found at the following *GitHub repository: https://github.com/AndreiDumitrescu99/SSL_Project*. The repository contains all the code developed for all the milestones, a REAME.md file which explains how to run the experiments and also all the documentation produced for this project.

7. *References*

- *Caselli, T., Basile, V., Mitrović, J., & Granitzer, M. (2020). Hatebert: Retraining bert for abusive language detection in english. arXiv preprint arXiv:2010.12472.*
- *PyTorch training tutorial: https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html*
- *HateBERT hugging face model: https://huggingface.co/GroNLP/hateBERT*