TüKaSt at SemEval-2019 Task 6: Something Old, Something Neu(ral): Traditional and Neural Approaches to Offensive Text Classification

Madeeswaran Kannan  Lukas Stein
Department of Linguistics
University of Tübingen, Germany
{mkannan,lstein}@sfs.uni-tuebingen.de

Abstract

We describe our system (TüKaSt) submitted for Task 6: Offensive Language Classification, at SemEval 2019. We developed multiple SVM classifier models that used sentence-level dense vector representations of tweets enriched with sentiment information and term weighting. Our best results achieved F1 scores of 0.734, 0.660 and 0.465 in the first, second and third sub-tasks respectively. We also describe a neural network model that was developed in parallel but not used during evaluation due to time constraints.

1 Introduction

We live in a day and age where social media pervades through every aspect of lives. Constant growth of social media platforms and rapid exposure to them are changing how human communication is perceived and working (Sticca and Perren, 2013). As much as social media and the general web help its users stay connected and informed, misuse of these channels grows with them. As long as anonymity continues to play a central role in online interactions, one must contend with individuals and entities who feel empowered to behave aggressively for that very reason (Burnap and Williams, 2015). Recent events have forced social media platforms to take a renewed interest in limiting the negative influence of such users for commercial and practical reasons. However, the sheer size of data on social media makes it impossible to manually observe, thereby calling the creation of systems that automatically detect potentially offensive content.

The SemEval 2019 - OffensEval Shared Task (Zampieri et al., 2019b) is aiming exactly at that need, calling for system able to detect offensive language in social media data. The task is split into three sub-tasks: Detecting offensive language, determining whether it is offensive directly towards a target, and target specification. This paper follows our approach of applying both traditional and more sophisticated neural models to the task of offensive text classification.

2 Preprocessing

All tweets were converted to lowercase and tokenised. Stop-words were removed unconditionally, but hashtags and user mentions were removed during the training of certain model variants.

3 Training Data

The SVM models were trained and evaluated exclusively on the training data provided by Zampieri et al. (2019a) with ten-fold cross-validation. For the RNN models, we also used the trail data that was provided before the start of the training phase. The samples were shuffled before each training run, and a 80-20 split was performed to generate training and validation sets.

300-dimensional, pre-trained FastText embeddings (Mikolov et al., 2018) were used to train the SVM models. The FastText embeddings were chosen due to their subword-information (Bojanowski et al., 2017), making it easier to deal with social media data, which is prone to contain spelling errors and abbreviations. 100-dimensional Glove vectors trained on the Twitter corpus (Pennington et al., 2014) were used to initialise the word embeddings in the RNN models.

For sentiment data, we used the Vader sentiment lexicon (Gilbert, 2014) to retrieve polarity scores for each word in the vocabulary. Scores
range from -4 (extremely negative) to +4 (extremely positive). A neutral score of zero was assigned to out-of-vocabulary words.

4 Models

4.1 Support Vector Machines Classifier
Support Vector Machines (SVMs) have been used successfully in many classification problems concerning natural language (Aggarwal and Zhai, 2012). In the proceedings of the First Workshop on Trolling, Aggression and Cyberbullying, Kumar et al. (2018) report on several teams using SVMs to identify aggressive texts. Moreover, SVMs with traditional features can still outperform neural networks in natural language tasks (Medvedeva et al., 2017; Çöltekin and Rama, 2017, 2018). However, traditional features like TF-IDF, character and word n-grams, bag of words/n-grams, and sentiment lookups dominate the majority of models used for identifying aggressive language.

Contrasting the use of traditional features used in natural language processing is the prominent research on neural networks using dense vector representations (word embeddings) as input. Since embeddings are able to capture syntactic and semantic features and represent them in a distributed manner, it makes them perfect for language modelling (Bojanowski et al., 2017). We chose to experiment with embeddings in SVMs for this classification task in order to keep pre-processing and feature-engineering to a minimum. Moreover, we wanted to see how the SVM could improve with the dense representations over sparse representations. However, we also experimented with combinations of continuous features and traditional features in some of our models.

The SVM models’ dense sentence representation was composed by summing the respective embeddings of every word in the tweet and normalising over the length of the tweet. We chose not to weight them by their TF-IDF scores to achieve an even combination of all constituents. Mikolov et al. (2013) report that adding vectors in a linear fashion yields meaningful phrase representations.

For the first of three SVM models (combined_fixed), we created vectors for the TF-IDF and the retrieved sentiment values in the following manner: The TF-IDF vectors were constructed with length $|V|$, where $V$ is the vocabulary constructed from the corpus and each dimension is corresponding to a specific word in the model’s vocabulary. The sentiment vector was constructed with length $|V'|$, where $V'$ corresponds to the Vader lexicon (Gilbert, 2014) and each dimension corresponds to a specific word in the lexicon. Both vectors were initialised with zeros and respective dimensions were replaced with the retrieved/computed values. Both vectors were then appended to the dense sentence representation.

The second model (combined_positioned) was trained with a word-order sensitive representation of the aforementioned features. Both the TF-IDF vector and the sentiment vectors were constructed with length $|\text{word\_count}(t)|$, where $t$ is the longest tweet in the training set and where each the dimension corresponds to the word encountered at the respective index in the tweet. The third and final SVM model was only trained on the composed sentence embeddings (emb_only).

4.2 Recurrent Neural Classifier
In addition to the SVM classifier, we parallelly trained a recurrent neural classifier using both Long Short-Term Memory (Hochreiter and Schmidhuber, 1997) and Gated Recurrent Unit (Cho et al., 2014) cells.

![RNN Classifier Architecture](image)

The architecture of our model is illustrated in figure 1. It accepts token and character n-gram
sequences as inputs. The embedding layer consists of individual embedding matrices for each input type. The word embedding matrix is initialised with pre-trained word embeddings while the character embedding matrix is randomly initialised, both of which are subsequently learned as parameters of the model during the training phase.

The embedding sequence is used as the input to a stacked bi-directional RNN layer. We used both LSTM (with peepholes) and GRU cells for the individual units in each RNN layer (trained as separate models). Dropout is additionally added to each layer during the training phase. The hidden states of the forward and backward RNNs of the top-most layer are concatenated to produce the sentential representation of the sequence. This is performed for both types of input sequences.

In addition to the output of the biRNN layers, we calculate TF-IDF (Term Frequency - Inverse Document Frequency) scores for all words in the corpus. This lets us construct a fixed-length dense vector of normalised TF-IDF values for each tweet. Similarly, we use a sentiment lexicon to lookup human-assigned polarity scores of individual words and construct a second fixed-length vector with sentiment information for each tweet.

The outputs of the above components are concatenated into a single, final representation. We expect the TF-IDF and sentiment vectors to encode relevant information about the importance of discriminating words in the input sequence. This final representation is used as the logits for a SoftMax layer that outputs the predicted label of the tweet.

5 Parameters & Training

We used sklear-n-kittex (Pedregosa et al., 2011) to build our SVM models. For both Sub-task A and B, a binary classifier was trained and for Sub-task C a one-versus-one, multi-class classifier was trained on the three different labels. All three models use sklear’s default RBF-kernel with C=2 and gamma=0. The final hyper-parameters used by the neural classifier are listed in table 7. Training was performed with early stopping.

During our initial tests, the neural network classifier particularly had trouble generalising to the training data. We found that the class imbalance in the data caused the classifier to be biased towards the majority class and to over-fit the training data. To mitigate this, we weighted the output of the loss function with the ratios of the different classes in each dataset and performed L2 regularisation on the losses of each mini-batch.

6 Results

The results that follow are those only those of the SVM models evaluated during the evaluation phase of the OffenEval task (Zampieri et al., 2019b). Since we were unable to train the RNN classifier in time for the official evaluation phase, we will instead be reporting the scores of the classifier on the validation set (mean and best macro-F1 scores and their corresponding accuracies) in section A.

The model using the sentence embedding on its own is denoted by emb_only, the combination of sentence embedding and multiple-hot TF-IDF and sentiment vectors by combined_fixed and the position sensitive combination of the vectors by combined_positioned.

For Sub-task B and C, all three models were submitted and for sub-task A, only the emb_only and the combined_fixed were submitted.

| System            | F1 (macro) | Accuracy |
|-------------------|------------|----------|
| All NOT baseline  | 0.4189     | 0.7209   |
| All OFF baseline  | 0.2182     | 0.2790   |
| combined_fixed    | 0.7340     | 0.7942   |
| emb_only          | 0.7127     | 0.7849   |

Table 1: Results for Sub-task A.

The results for Sub-task A (Table 1) reveal a slight improvement for adding traditional features to the dense sentence representation. This improvement is likely to be due to the sentiment vector, whose values should be very discriminative for tweets containing offensive language. Nevertheless, this information should be encoded in the embeddings as well.

For Sub-task B (Table 2) and Sub-task C, the emb_only model actually outperforms the combined vectors. This might be due to the fact that
tasks are more fine-grained than Task A and these
nuances are captured by the word embeddings,
while the sentiment and TF-IDF vectors end up
contributing more to the noise than information
in the signal. Nevertheless, the SVM seems to
be able to handle the data from the composed
sentence embeddings quite well.

| System                      | F1 (macro) | Accuracy |
|-----------------------------|------------|----------|
| All TIN baseline            | 0.4702     | 0.8875   |
| All UNT baseline            | 0.1011     | 0.1125   |
| combined_positioned         | 0.6470     | 0.8      |
| combined_fixed              | 0.4702     | 0.8875   |
| emb_only                    | 0.6602     | 0.8208   |

Table 2: Results for Sub-task B.

| System                      | F1 (macro) | Accuracy |
|-----------------------------|------------|----------|
| All GRP baseline            | 0.1787     | 0.3662   |
| All IND baseline            | 0.2130     | 0.4695   |
| All OTH baseline            | 0.0941     | 0.1643   |
| combined_positioned         | 0.4421     | 0.5305   |
| combined_fixed              | 0.4213     | 0.4695   |
| emb_only                    | 0.4652     | 0.5164   |

Table 3: Results for Sub-task C.

The results obtained for Sub-task B reveal that
the model trained on combined_fixed performs
remarkably badly when evaluated on the F1
score. Moreover, the accordance on Accuracy
with the All-TIN-Baseline indicates that our
model over-fits the training data and classifies
all samples as being targeted, which is actually
the case. This problem and general performance
could probably be improved by spending more
time on hyper-parameter training.

The per-class scores of the best performing
model in each sub-task are listed in tables 4-6.

| System      | Precision | Recall | F1-score | Samples |
|-------------|-----------|--------|----------|---------|
| TIN         | 0.9427    | 0.8498 | 0.8938   | 213     |
| UNT         | 0.3333    | 0.5926 | 0.4267   | 27      |

Table 5: Per-class performance in Sub-task B, SVM model emb_only

| System      | Precision | Recall | F1-score | Samples |
|-------------|-----------|--------|----------|---------|
| GRP         | 0.5306    | 0.6667 | 0.5909   | 78      |
| IND         | 0.7424    | 0.4900 | 0.5904   | 100     |
| OTH         | 0.1837    | 0.2571 | 0.2143   | 35      |

Table 6: Per-class performance in Sub-task C, SVM model emb_only

7 Conclusion

In this paper, we demonstrate the use of both Support
Vector Machine models and Recurrent Neural
models to classify offensive tweets. We show
how sentential representations derived from word
and character n-gram embeddings can be enriched
by including term salience and sentiment informa-
tion for certain tasks. For more fine-grained
tasks, dense vector representations of sentences
work well with SVM classifiers. While our results
show that traditional methods still outperform neu-
ral network models, there is much room for im-
provement. Future work could investigate the use
of more sophisticated mechanisms like attention to
potentially increase performance even further.

References

Charu C Aggarwal and ChengXiang Zhai. 2012. A sur-
vey of text classification algorithms. In Mining text
data, pages 163–222. Springer.
Piotr Bojanowski, Edouard Grave, Armand Joulin, and
Tomas Mikolov. 2017. Enriching word vectors with
subword information. Transactions of the Association
for Computational Linguistics, 5:135–146.
Pete Burnap and Matthew L Williams. 2015. Cyber
hate speech on twitter: An application of machine
classification and statistical modeling for policy and
decision making. Policy & Internet, 7(2):223–242.
Kyunghyun Cho, Bart van Merrienboer, Çağlar
Gülçehre, Fethi Bougares, Holger Schwenk, and
Yoshua Bengio. 2014. Learning phrase representa-
tions using RNN encoder-decoder for statistical ma-
chine translation. CoRR, abs/1406.1078.
Çağrı Çöltekin and Taraka Rama. 2017. Tübingen
system in vardial 2017 shared task: Experiments
with language identification and cross-lingual parsing. In Proceedings of the Fourth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial), pages 146–155.

Çağrı Çöltekin and Taraka Rama. 2018. Tübingen-oslo at semeval-2018 task 2: Svms perform better than rnns in emoji prediction. In Proceedings of The 12th International Workshop on Semantic Evaluation, pages 34–38.

CJ Hutto Eric Gilbert. 2014. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In Eighth International Conference on Weblogs and Social Media (ICWSM-14). Available at (20/04/16) http://comp. social. gatech. edu/papers/icwsm14. vader. hutto. pdf.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. Neural computation, 9(8):1735–1780.

Ritesh Kumar, Atul Kr. Ojha, Shervin Malmasi, and Marcos Zampieri. 2018. Benchmarking Aggression Identification in Social Media. In Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC), Santa Fe, USA.

Maria Medvedeva, Martin Kroon, and Barbara Plank. 2017. When sparse traditional models outperform dense neural networks: the curious case of discriminating between similar languages. In Proceedings of the Fourth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial), pages 156–163.

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2018. Advances in pre-training distributed word representations. In Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018).

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In Advances in neural information processing systems, pages 3111–3119.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Pas- sos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12:2825–2830.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), pages 1532–1543.

Fabio Sticca and Sonja Perren. 2013. Is Cyberbullying Worse than Traditional Bullying? Examining the Differential Roles of Medium, Publicity, and Anonymity for the Perceived Severity of Bullying. Journal of Youth and Adolescence, (4):739–750.
A Appendix

|                |       |
|----------------|-------|
| Epochs         | 50    |
| Batch size     | 512   |
| L2 Beta        | 0.001 |
| RNN per-layer dropout | 0.15 |
| Character n-gram size | 3     |
| RNN output dimension | 100   |
| Auxiliary vector dimension | 30    |

Table 7: RNN Model Hyper-parameters

A.1 Recurrent Neural Classifier Validation

Results

The -hm suffix denotes the models that were trained on tweets from which hash tags and user mentions were removed. -aux denotes the models that did not use the fixed-length TF-IDF and sentiment vectors in the final representation. In all other models, both of the aforementioned features were preserved. We also report the validation results of the SVM models for comparison.

In Sub-task A (Table 8), the full-LSTM model with TF-IDF and sentiment vectors scored best, followed very closely by the model without hashtags or user mentions. Removing the auxiliary vectors resulted in a net decrease in performance, which shows that sentiment and [term] salience-related information do indeed help the model learn and generalise better. Interestingly, removing hash tags/mentions along with the auxiliary vectors increased performance relative to the previous case. The GRU scored the lowest; this could potentially be attributed to its weakness with long-distance dependencies.

The full-LSTM model continued to outperform the other models in Sub-task B (Table 9) as well. Removing hash tags and user mentions caused a more substantial drop in performance compared to the first sub-task. This follows logically as the given task is to identify targeted tweets; removing information that is intrinsically indicative of the same results in a worse-performing model. Removing the auxiliary vectors causes performance to drop further, further corroborating their informativity. And as before, the GRU finished in the last place.

Sub-Task C (Table 10) saw a reversal of roles between the GRU and full-LSTM models. Between the different variants of the LSTM-based models, the general trend seen in Sub-task A re-emerged. It must, however, be noted that all models performed poorly at this sub-task. We conjecture that this is partly due to the limited amount of training data available for this sub-task. It is also likely that the chosen architecture of the classifier is not sophisticated enough to model the non-linearities in the training data.
### Table 8: Validation results for Sub-task A.

| System                  | Mean F1 (macro) | Mean Accuracy | Best F1 (macro) | Best Accuracy |
|-------------------------|-----------------|---------------|-----------------|---------------|
| SVM combined_positioned | 0.7215          | 0.7540        | 0.7441          | 0.7751        |
| SVM combined_fixed      | 0.6171          | 0.7045        | 0.6321          | 0.7177        |
| SVM emb_only            | 0.7160          | 0.7483        | 0.7373          | 0.7691        |
| GRU                     | 0.6240          | 0.6584        | 0.6442          | 0.6941        |
| LSTM                    | 0.6958          | 0.7332        | 0.7121          | 0.7629        |
| LSTM -hm                | 0.6941          | 0.7360        | 0.7126          | 0.7488        |
| LSTM -aux               | 0.6830          | 0.7361        | 0.6873          | 0.7559        |
| LSTM -hm -aux           | 0.6852          | 0.7166        | 0.6997          | 0.7309        |

### Table 9: Validation results for Sub-task B.

| System                  | Mean F1 (macro) | Mean Accuracy | Best F1 (macro) | Best Accuracy |
|-------------------------|-----------------|---------------|-----------------|---------------|
| SVM combined_positioned | 0.5877          | 0.8117        | 0.6341          | 0.8616        |
| SVM combined_fixed      | 0.4677          | **0.8791**    | 0.4737          | **0.9002**    |
| SVM emb_only            | 0.6128          | 0.8129        | **0.6676**      | 0.8594        |
| GRU                     | 0.5949          | 0.6350        | 0.6129          | 0.6578        |
| LSTM                    | **0.6527**      | 0.7065        | 0.6642          | 0.7039        |
| LSTM -hm                | 0.6322          | 0.6880        | 0.6372          | 0.6859        |
| LSTM -aux               | 0.6267          | 0.6696        | 0.6527          | 0.6809        |
| LSTM -hm -aux           | 0.6258          | 0.6528        | 0.6655          | 0.6973        |

### Table 10: Validation results for Sub-task C.

| System                  | Mean F1 (macro) | Mean Accuracy | Best F1 (macro) | Best Accuracy |
|-------------------------|-----------------|---------------|-----------------|---------------|
| SVM combined_positioned | 0.4003          | 0.5788        | 0.4934          | 0.6340        |
| SVM combined_fixed      | 0.4919          | 0.6129        | 0.5163          | 0.6417        |
| SVM emb_only            | **0.5086**      | 0.6273        | **0.5672**      | 0.6494        |
| GRU                     | 0.2953          | 0.5748        | 0.3109          | 0.5746        |
| LSTM                    | 0.2878          | 0.6045        | 0.2961          | 0.6164        |
| LSTM -hm                | 0.2868          | 0.6365        | 0.3390          | 0.6152        |
| LSTM -aux               | 0.2586          | **0.6432**    | 0.2637          | **0.6762**    |
| LSTM -hm -aux           | 0.2617          | 0.5370        | 0.2896          | 0.4605        |