UNBNLP at SemEval-2019 Task 5 and 6: Using Language Models to Detect Hate Speech and Offensive Language

Ali Hakimi Parizi, Milton King and **Paul Cook**
Faculty of Computer Science, University of New Brunswick
Fredericton, NB E3B 5A3, Canada
ahakimi@unb.ca, milton.king@unb.ca, paul.cook@unb.ca

Abstract

In this paper we apply a range of approaches to language modeling — including word-level n-gram and neural language models, and character-level neural language models — to the problem of detecting hate speech and offensive language. Our findings indicate that language models are able to capture knowledge of whether text is hateful or offensive. However, our findings also indicate that more-conventional approaches to text classification often perform similarly or better.

1 Introduction

SemEval 2019 Task 5 focuses on detecting hate speech in social media text, while Task 6 considers identifying offensive language. Despite the differences between hate speech and offensive language, each of these tasks can be viewed as a binary classification problem. In each case, gold-standard training data is provided on which supervised approaches can be trained.

In this paper we consider whether approaches to classification based on language models — including word- and character-level neural language models, as well as more-conventional (word-level) n-gram language models — are able to distinguish between hateful and not hateful, and offensive and not offensive, language. We find that these approaches outperform a most-frequent class baseline, indicating that language models can capture some knowledge of whether text is hateful or offensive. However, for task 5 — for which the testing data was made available for follow up experiments — we also find that more-conventional approaches to supervised classification, such as naive Bayes and fastText (Joulin et al., 2017), often give similar or better results.

2 Related Work

Social media such as Twitter and Facebook are widely used, and hate speech has become prevalent in these platforms. In response to this, a substantial amount of research has recently focused on detecting such disturbing comments.

Prior work on hate speech and offensive language detection has mostly focused on supervised machine learning techniques (Mathur et al., 2018; Davidson et al., 2017). A recent shared task on identifying aggression in social media (Kumar et al., 2018) observed there is no significant difference between the performance of neural networks and linear classifiers. Furthermore, this shared task received just one lexicon-based approach, and its performance was not promising. Moreover, all of these approaches required expensive feature engineering and pre-processing.

Instead of engineering specific features to use in supervised classifiers, we can instead employ language models to model the type of text we want to detect. Language models have previously been applied for the purpose of text classification (e.g., Bai et al., 2004; Howard and Ruder, 2018). Among different types of language models, recurrent neural network (RNN) language models with LSTM and GRU units have shown promising results for sequence modeling (Mikolov et al., 2012). To the best of our knowledge, RNN language models have not been widely used for detecting hate speech or offensive language. The most closely related work is that of Mehdad and Tetreault (2016), which used both word-level and character-level RNNs to detect abusive language. In their experiments, Mehdad and Tetreault found that character-level language models outperformed word-level language models. A further advantage of character-level language models is that they are able to model out-of-vocabulary

Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019), pages 514–518
Minneapolis, Minnesota, USA, June 6–7, 2019.

words (Mikolov et al., 2012).

3 Task 5: HatEval

SemEval 2019 Task 5 includes two sub-tasks: (A) detecting hate speech in English and Spanish tweets, and (B) classifying hate speech tweets as aggressive or not, and as targeting an individual or group. We only consider sub-task A. In Section 3.1, we describe the approaches we considered for this task, and in Section 3.2 we present our results.

3.1 Approaches

3.1.1 Word-level LMs: n-gram and LSTM

For each language, we grouped the training instances based on their gold standard labels — giving us two corpora per language — with one consisting entirely of hateful tweets, and the other consisting of not hateful tweets. For each language, we then trained two language models (LMs), one on the hateful instances, and the other on the non-hateful instances. Given a test instance, we calculate the probability of the tweet under each LM. If the LM that was trained on the hateful text gave a higher probability then we labeled the instance as hateful, otherwise we labeled it as non-hateful. The two word-level LMs that we considered are an n-gram LM and a long short-term memory (LSTM) LM. The n-gram model was a 3-gram model with Kneser-Ney smoothing trained using Kenlm (Heafield et al., 2013) with its default settings. We tuned the parameters for the LSTM model on the English development dataset using grid search. Specifically we considered the following settings for the embedding size (256, 512, 1024), number of hidden units (128, 256, 512), and number of epochs (1,2,3,4,5). The final parameters for the LSTMs used on the test datasets were 1 hidden layer, an embedding size of 1024, 128 hidden units, and they were trained using a batch size of 2 and 1 epoch.

3.1.2 Character-level LM

This approach is the same as the previous word-level LM approach, except that character-level, as opposed to word-level, LMs are trained. We use a publicly available TensorFlow implementation of a character-level RNN language model.[1] The following parameters are used: a two-layer GRU with one-hot character embeddings and a hidden layer size of 64 dimensions. The batch size, learning rate, and dropout are set to 20, 0.002, and 0, respectively. The hidden layer size and unit were tuned on the development data. Specifically we considered hidden layer sizes of 64, 128, and 256, and an LSTM and GRU for the unit. The other parameters are their default settings.

3.1.3 Neural LMs with Class Token

In the previous approaches, two separate LMs are trained — one on the hateful tweets, the other on the non-hateful ones. In contrast, in this approach a single LM is trained.

We append a special token to the end of each tweet in the training data representing its gold standard class. We randomly shuffle the order of the tweets in the training data, and then train a LM model on them. At test time, we feed a tweet (which has not been augmented with a special token indicating its class) to the LM. We then query the LM for the probability of the special tokens representing the hateful and non-hateful classes, and classify the tweet as the class corresponding to the special token with higher probability.

We consider both word-level and character-level neural LMs for this approach. For the word-level LM we again use an LSTM. We performed a grid search to tune its parameters using the English development dataset. Specifically we considered the following settings for the number of layers (1,2), embedding size (128, 256, 512), number of hidden units (128, 256, 512), and number of training epochs (1,2,3). The final model consisted of two hidden layers, an embedding size of 512, 128 hidden units, and was trained using a batch size of two for three epochs. For the character-level LM we use the same model as in Section 3.1.2, with the same parameter settings.

3.1.4 Baselines

In addition to the most-frequent class and SVC baselines provided by the shared task (Basile et al., 2019), we also compare our approaches against multinomial naive Bayes[2] and fastText (Joulin et al., 2017). We use the default settings for fastText, and do not attempt to tune it to this task.

[1] https://github.com/crazydonkey200/tensorflow-char-rnn

[2] Note that the likelihood term in multinomial naive Bayes corresponds to a unigram LM for each class. As such it is similar to the LM-based approaches we consider, but also incorporates a class prior.

	English								Spanish							
	Dev				Test				Dev				Test			
Method	F	P	R	A	F	P	R	A	F	P	R	A	F	P	R	A
n-gram	**0.70**	0.71	0.71	**0.70**	0.45	**0.60**	**0.55**	0.49	0.71	0.72	0.72	0.71	0.66	0.67	0.68	0.66
LSTM	0.60	0.61	0.61	0.60	0.48	0.53	0.52	0.49	0.68	0.68	0.68	0.69	0.64	0.64	0.64	0.65
Char	**0.70**	0.71	0.71	**0.70**	0.45	0.57	0.54	0.49	0.67	0.68	0.67	0.68	0.66	0.66	0.66	0.67
LSTM+CT	0.50	0.54	0.54	0.51	**0.49**	0.53	0.53	0.50	0.55	0.56	0.56	0.56	0.52	0.57	0.56	0.53
Char+CT	0.52	0.53	0.53	0.53	0.47	0.47	0.47	0.49	0.53	0.54	0.53	0.55	0.48	0.48	0.48	0.51
NB	**0.70**	**0.73**	**0.72**	**0.70**	0.41	**0.60**	0.54	0.47	**0.74**	**0.77**	**0.73**	**0.75**	0.69	**0.70**	0.69	**0.71**
fastText	0.64	0.66	0.64	0.66	0.47	0.59	**0.55**	0.50	0.70	0.72	0.70	0.71	**0.70**	0.69	0.70	0.70
SVC	-	-	-	-	0.45	**0.60**	**0.55**	0.49	-	-	-	-	**0.70**	**0.70**	**0.71**	**0.71**
MFC	0.36	0.29	0.50	0.57	0.37	0.29	0.50	**0.58**	0.36	0.28	0.50	0.56	0.37	0.29	0.50	0.59

Table 1: Macro average F1-score (F), macro average precision (P), macro average recall (R), and accuracy (A) on Task 5 subtask A using word-level n-gram and LSTM LMs, a character-level LM (Char), a word-level LSTM and character-level LM augmented with a special class token (LSTM+CT and Char+CT), multinomial naive Bayes (NB), and fastText on the development and test sets. The SVC and most-frequent class (MFC) baselines provided by the shared task are also shown. The best result for each language, dataset, and evaluation measure is shown in boldface.

3.2 Results

The English and Spanish training sets contain 9,000 and 4,500 tweets, respectively, labelled as being hateful or non-hateful. The development and test sets contain 1,000 and 2,971 tweets, respectively, for English, and 500 and 1,600 tweets, respectively, for Spanish. Further details of the datasets are provided in Basile et al. (2019).

Results are shown in Table 1.[3] The character-level LM, which performed best on the development data, corresponds to our official submission for the shared task. In terms of F-score, for both languages, all LM-based approaches outperform the most-frequent class baseline. This indicates that LMs are able to capture information about whether a tweet is hate speech or not. However, more-conventional approaches to classification perform similarly to, or better than, the LM-based approaches. Focusing on the test data, for English, although the LSTM with class token approach achieves the best F-score of 0.49, fastText achieves only a slightly lower F-score of 0.47. For Spanish, fastText and SVC achieve the best F-score of 0.70, while the best LM-based approaches, the n-gram and character LMs, obtain an F-score of 0.66.

4 Task 6: OffensEval

SemEval 2019 Task 6 includes 3 sub-tasks. In contrast to Task 5, we participated in all subtasks of Task 6. Sub-task A is a binary classification task to determine if a tweet is offensive or non-offensive. Sub-task B is to classify tweets that are offensive into two groups of targeted — a post containing an insult or threat to an individual, a group, or others — or untargeted — a post containing non-targeted profanity and swearing. Finally, sub-task C is a three-way classification task in which targeted tweets (from sub-task B) are classified as targeting an individual, group of people, or other.

Because of the similarities between hate speech and offensive language, we apply approaches that we used for Task 5 to Task 6. We used the development data for Task 5 for model tuning and selection, and only considered the three best models for Task 6: the word-level n-gram and LSTM language models and the character-level language model. We describe these approaches in Section 4.1 and report results over the test data in Section 4.2.[4]

4.1 Approaches

4.1.1 Word-level LMs: n-gram and LSTM

Similar to Task 5, for each sub-task of Task 6, we group the training instances based on their gold-standard classes. We then train one LM on the documents from each class. I.e., in the case of sub-task A we train one LM on tweets labeled offensive, and another LM on tweets labelled non-offensive. At test time we measure the probability

[3] The SVC baseline was only provided for the test data.

[4] We do not report results for the development data because we used Task 5 data for model tuning and selection.

	Sub-task A		Sub-task B		Sub-task C	
Method	F	A	F	A	F	A
n-gram	**0.62**	0.66	0.45	0.50	**0.43**	0.44
LSTM	0.55	0.59	0.54	0.73	0.40	**0.48**
Char	0.59	0.63	**0.61**	0.88	-	-
MFC	0.42	**0.72**	0.47	**0.89**	0.21	0.47

Table 2: Macro-average F1-score (F) and accuracy (A) for each sub-task of Task 6 using word-level n-gram and LSTM LMs, a character-level LM (Char), and a most-frequent class baseline (MFC). The best result for each sub-task and evaluation measure is shown in boldface.

of a test tweet under each LM, and then classify it as the class corresponding to the LM giving the highest probability. Note that sub-task C is a three-way, as opposed to binary, classification task, and so we therefore train three LMs for this sub-task.

We consider the same word-level LMs as for Task 5 — an n-gram LM and an LSTM LM. We tuned the parameters for the LSTM on the English development dataset of Task 5, sub-task A, using grid search as described in Section 3.1.1. We did not further tune this model to Task 6. For the n-gram model we again use a 3-gram model as described in Section 3.1.1.

4.1.2 Character-level LM

We apply character-level LMs to each sub-task of Task 6 in the same manner as we use the word-level LMs described above. We use the same character-level LM as for Task 5, described in Section 3.1.2, with the same parameter settings. We do not attempt to further tune the parameters to Task 6.

4.2 Results

Details of the training and test data can be found in Zampieri et al. (2019). Results for the LM-based approaches described above, as well as a most-frequent class baseline, are presented in Table 2. In terms of F1-score, for each sub-task, each LM-based approach outperforms the most-frequent class baseline, with the exception of the n-gram LM on sub-task B.[5] These results, and in particular those on sub-task A, demonstrate that LMs can capture knowledge about whether text is offensive.

On sub-task A the n-gram LM achieved the best F1-score, while on sub-task B the character-level LM did. One reason for his could be differences in the size of the training data. For sub-task A, we use all of the training data (13,240 instances) to train our models. However, for sub-task B, we are limited to just those tweets that were labeled as offensive. There are only 4,400 such tweets. Although this reduction in the amount of training data caused the F1-score of the word-level LM-based approaches to decrease, the amount of training data seems to still be sufficient to train a character-level LM. The F1-scores on sub-task C are lower than for the other sub-tasks. One possible explanation for this relatively poor performance is that the training data for sub-task C is smaller than that for the other sub-tasks (3,876 instances), because the sub-task C training data is a subset of that for sub-task B.

5 Conclusions

In this paper we employed language models to the problems of detecting hate speech and offensive language in social media text. We considered a range of approaches to language modeling including word-level n-gram and neural language models, and a character-level neural language model. Our results indicated that language model-based approaches are able to capture knowledge of whether text is hateful or offensive. However, further experiments on identifying hate speech indicated that more-conventional approaches to text classification often perform comparably or better.

In this paper we only considered language models trained on the training data provided for the shared tasks. In future work, we intend to consider pre-training language models on other corpora (e.g., Twitter corpora) in an effort to improve the performance of language model-based approaches to detecting hate speech and offensive language.

[5] Due to an error in our submission for the character-level model for sub-task C, we did not receive results for this approach on this sub-task.

References

Jing Bai, Jian-Yun Nie, and François Paradis. 2004. Using language models for text classification. In *Proceedings of the Asia Information Retrieval Symposium (AIRS)*.

Valerio Basile, Cristina Bosco, Elisabetta Fersini, Debora Nozza, Viviana Patti, Francisco Rangel, Paolo Rosso, and Manuela Sanguinetti. 2019. Semeval-2019 task 5: Multilingual detection of hate speech against immigrants and women in twitter. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019)*. Association for Computational Linguistics.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. In *InProceedings of ICWSM*, pages 512–515.

Kenneth Heafield, Ivan Pouzyrevsky, Jonathan H. Clark, and Philipp Koehn. 2013. Scalable modified Kneser-Ney language model estimation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 690–696, Sofia, Bulgaria.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339. Association for Computational Linguistics.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431, Valencia, Spain. Association for Computational Linguistics.

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-2018)*, pages 1–11.

Puneet Mathur, Rajiv Shah, Ramit Sawhney, and Debanjan Mahata. 2018. Detecting offensive tweets in hindi-english code-switched language. In *Proceedings of the Sixth International Workshop on Natural Language Processing for Social Media*, pages 18–26.

Yashar Mehdad and Joel Tetreault. 2016. Do characters abuse more than words? In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 299–303.

Tomáš Mikolov, Ilya Sutskever, Anoop Deoras, Hai-Son Le, Stefan Kombrink, and Jan Cernocky. 2012. Subword language modeling with neural networks. *preprint (http://www. fit. vutbr. cz/imikolov/rnnlm/char. pdf)*, 8.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019. A Hierarchical Annotation of Offensive Posts in Social Media: The Offensive Language Identification Dataset. In *arxiv preprint*.