Sentiment Analysis and Transfer Learning for Hindi and Bengali Languages

Abhinav Bhatt
7010946
Universität des Saarlandes, Germany
abbh00001@stud.uni-saarland.de

Deepa Rani Mahato
7012336
Universität des Saarlandes, Germany
dema00001@stud.uni-saarland.de

Abstract

Sentiment analysis has innumerable applications nowadays. We perform analysis of some deep neural network architectures on the sentiment analysis task for Hindi and the Bengali language hate speech identification datasets. We also try to see if we can transfer the knowledge learned by the model from the Hindi dataset to the Bengali dataset. We train skip-gram implementation of word2vec model to get distributed representations for the words in our datasets. We observe that Bi-LSTM models with attention perform better than CNN based model which are initialized with pre-trained embeddings. We also observe that transfer learning for our models does not help much in the knowledge transfer of Hindi to Bengali language. We believe this is due to the small size of the datasets which we have and the relatively small size of the models(compared to today's standards) we use.

1 Introduction

Sentiment analysis is a research area that uses natural language processing, text analysis, or computational linguistic techniques to determine people's opinions or sentiments towards different things. Sentiment analysis has a wide range of applications [10]. One of the applications of sentiment analysis is Hate speech identification. Hate speech is one of the most important problems in social media nowadays. Hate speech and Offensive content severely limit the effectiveness of social media platforms. Harmful effects of hate speech have been widely studied [5, 15], and a lot of research is being done in this direction. Sentiment analysis especially on social media poses some problems as the sentences written sometimes are not grammatically correct and the vocabulary used by all the people is also not consistent. There are things such as hashtags, emojis, etc. which make the detection of sentiment more difficult. Since the meaning of most words differs based on the context in which they are used, it becomes more tedious to identify the sentiment [8].

Transfer learning is a technique where the knowledge learned from one kind of problem is applied to a related but different problem. Transfer Learning became very popular for image based models since these [9] results on ImageNet dataset [2], but transfer learning for NLP started showing promising results since a few years back [3, 6].

In this report, we utilize neural network based approaches for the classification of hate speech and offensive content on the HASOC Hindi [12] dataset and also on a similar Bengali dataset. We implement and train word2vec [13, 14] skip-gram implementation for getting the distributed representations of words in our datasets. We also try transfer learning to see whether we can transfer knowledge learned during the training on the Hindi dataset and use it for the Bengali dataset. We observe that Bi-LSTM with attention performs the best for sentiment analysis on both datasets. We also observe that transfer of knowledge from the Hindi classifier is not very helpful for the sentiment analysis on the Bengali dataset.

Table 1: Some summary statistics about the datasets after data cleaning

Dataset	Dataset size (sentences)	Vocabulary size (words)	Average sentence length (words)	Maximum sentence length (words)
Hindi dataset	4378	15672	16.21	69
Bengali dataset	4331	14710	14.22	47

Table 2: Distribution of target classes in the datasets

Dataset	HOF (Hate and Offensive)	NOT (Non Hate-Offensive)
Hindi dataset	2392	1986
Bengali dataset	2376	1955

2 Methodology

For all of our experiments, we use the HASOC Hindi [12] dataset and a similar Bengali dataset.

2.1 Data cleaning and pre-processing

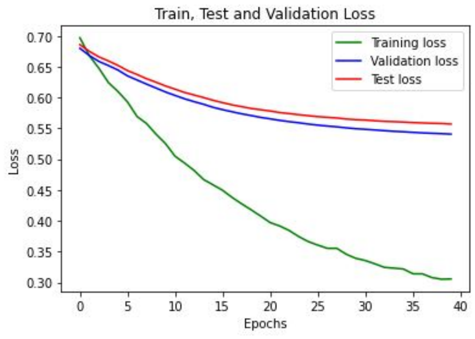
Both the Hindi and Bengali datasets contain tweets. Since social media data is particularly noisy, we apply some data cleaning strategies. Particularly, we remove the usernames (these start with @), we remove the URLs, we remove the punctuation symbols and we also remove the digits and English characters since we feel that all these things do not contribute much information. We also remove the Hindi stop-words from the dataset. Stop words are words that do not add much meaning to a sentence. Also, in datasets containing tweets we have a lot of hashtags and emojis. We remove the hashtags from our dataset since, in our datasets, most of the hashtags contain English words, which are not adding much value from hate speech identification standpoint in Hindi and Bengali language. We also remove the emojis from our dataset. We lowercase all of the words. Additionally, we also remove the sentences with only one word in them since they would have no context word. We also make the distribution of the Bengali dataset and labels similar to that of the Hindi dataset. The summary statistics for the cleaned datasets can be viewed in Table 1.

2.2 Training the word2vec model on the Hindi and Bengali datasets (Task 1)

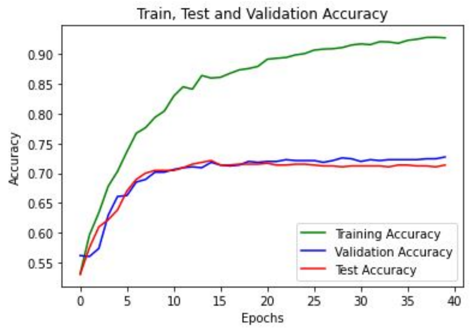
Word embeddings represent a word in the vector space such that the words that are similar to each other lie close in the vector space. Word2vec papers [13, 14] gave us efficient ways (architecture and training strategies) to train the word embeddings. After performing the data cleaning and pre-processing, we create word embeddings for the Hindi and the Bengali dataset using word2vec in our experiments. We do the skip-gram implementation of word2vec. In the skip-gram implementation, we have to predict the context words based on the center word. We employ a subsampling strategy which helps in reducing the size of (word, context) pairs and also reducing the percentage of words that appear a lot of times in the dataset. We find the probability of keeping the word in the context by the formula:

$$P_{keep}(w_i) = \left(\sqrt{\frac{z(w_i)}{0.000001}} + 1\right) \cdot \frac{0.000001}{z(w_i)}$$

Here $z(w_i)$ is the relative frequency of the word in the corpus. We do not remove the words from the context which appear a single time in our dataset because the size of the dataset is small and there are many words that occur only a single time. We use the embedding dimension as 300 which is the same as recommended by the word2vec paper. Word2vec paper recommends a window size of 10, but we use a window size of 5 in our experiments. There are two reasons for this. One reason for this is to reduce the size of the (word, context) pairs as we are not using negative sampling. The other reason is that since tweets are limited in length, we believe a window size of 5 would be able to capture most of the context for each word and reduce the noise. We experiment with learning rates and find

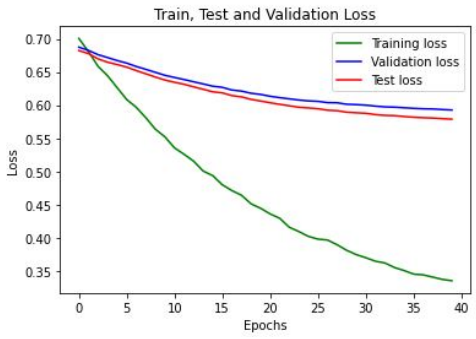


(a) Loss

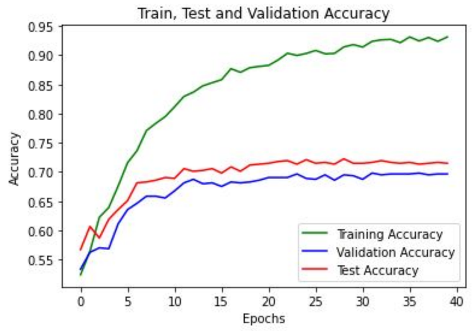


(b) Accuracy

Figure 1: Plots for loss and accuracy on the Hindi dataset using CNN based model.

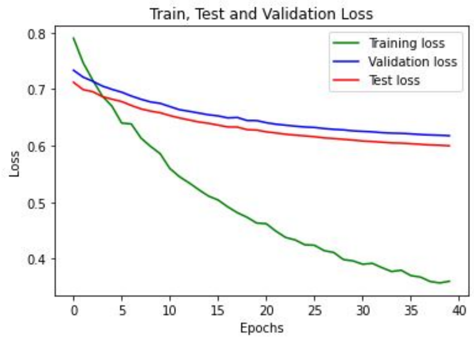


(a) Loss

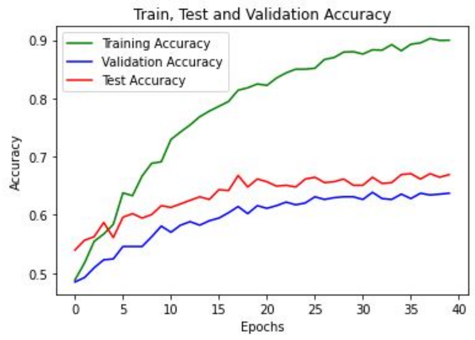


(b) Accuracy

Figure 2: Plots for loss and accuracy on the Bengali dataset using CNN based model.



(a) Loss



(b) Accuracy

Figure 3: Plots for loss and accuracy on the Bengali dataset using transfer learning with CNN.

that learning rate of 0.03 suits us the best. We train our model for 300 epochs. We use stochastic gradient descent as the optimizer and use negative log likelihood loss which is the same as used by the word2vec paper. To improve the training speed, we do a batched implementation and use batch size of 16. We then save our trained embeddings with the corresponding vocabulary for both Hindi and Bengali datasets.

2.3 Hindi and Bengali Classifier Training and Transfer Learning (Task 2)

With our trained embeddings on the Hindi and Bengali dataset from Task 1, we train our Hindi and Bengali hate speech classifiers and also try various transfer learning approaches. We are given tweets that we have to classify into either HOF(Hate and Offensive) or NOT(Non-Hate Offensive) labels.

We follow the same data pre-processing pipeline as we used in Task 1. We split our data into training, validation, and test sets. We keep 70% of the data in the training set, 15% in the validation set, and 15% in the test set. We do a stratified split of the data so that the distribution of labels follows the same ratio in all three sets. The distribution of the labels for the Hindi and the Bengali dataset can be viewed in Table 2.

The end-to-end neural architecture that we create for training the hate speech classifier is a convolution neural networks based architecture [7]. The architecture of our model is mainly inspired from [17]. We select CNNs because we think that they would be good for sentiment analysis due to their ability to look at multiple words at a time. CNNs for text classification are also fast to train and offer good performance. We use the word embeddings trained in Task 1 and train our model on top of those embeddings (Hindi embeddings for the Hindi dataset and Bengali embeddings for the Bengali dataset). We keep the embedding layer frozen in the experiments in this subsection, which means that we do not train the Task-1 trained word embeddings along with our classifier. We pad our input sentences. For the 1D CNNs, we use 4 kernel sizes which are 2, 3, 4, 5, and 18 filters for each of the sizes. We also use a dropout probability of 0.3 in the output layer. We take the outputs of each of these filters after applying them to input, max-pool them and then concatenate the outputs. Then we apply a linear layer to get the output values. We use Adam optimizer with a learning rate of 4e-5. We also exponentially decay our learning rate every epoch with a decay rate of 0.96. We use binary cross-entropy as our loss function. We train our model for 40 epochs. We train this classifier on the Hindi dataset. We save the weights of this classifier for further use while transfer learning.

For the initial exploration of transfer learning, we take the Hindi trained CNN classifier, give input to it the Bengali embeddings from Task 1 and test it on the Bengali dataset. We do this to see if there is any implicit transfer learning between Hindi and Bengali languages. We expect that this approach would not give good results since the model is trained on Hindi and we are testing Bengali data on it which it has never seen.

For the next transfer learning approach, we add one linear layer between the concatenated outputs of the CNN layers and the output layer. This approach is widely used nowadays in large pre-trained models such as Bert [3], Roberta [11], etc. In these models, a large pre-trained model gives the contextual representation of the sentence/words which is then used for downstream tasks. We add a linear layer containing 256 neurons and then train only this layer and the output layer while keeping all other parameters fixed. The rest of the procedure remains the same, we use Bengali embeddings and the weights of the CNNs are initialized to that of the Hindi classifier. Our aim is to see that whether Hindi classifier weights provide good feature vectors for the Bengali dataset.

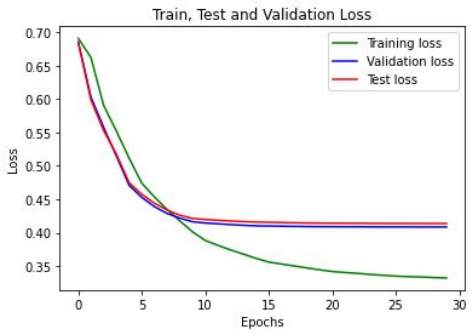
The next transfer learning approach we try is that we initialize the weights of the model to Hindi classifier weights and we train the whole model (except the Bengali embeddings). This is the approach in which large language models are pre-trained on large datasets and their weights are then utilized for the initialization of the network for another related task [6, 4]. Although most of these models keep different learning rates for different layers of the neural network, we do not do so. We aim to see whether the weights provide good initialization for the full model training on the Bengali dataset.

Then lastly, we retrain a new model with the Bengali dataset following the same procedure we used for the Hindi classifier using our CNN based classifier.

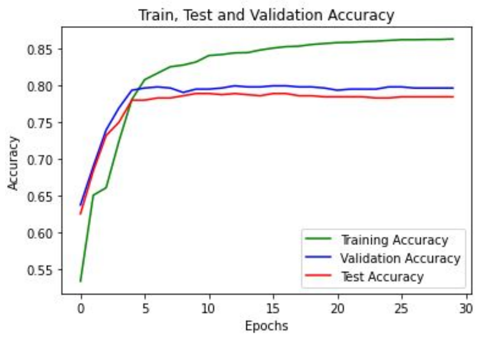
2.4 Improved Hindi and Bengali Classifier (Task 3)

To improve the scores on our previous CNN based architecture, we create a Bi-LSTM (Bidirectional Long Short Term Memory Networks) based architecture with attention. Bidirectional LSTMs capture both forward and backward contexts and attention pays importance to the certain important words in the sentence, both of these things we think will be helpful for sentiment analysis in our case. We evaluate the Bi-LSTM model both, with attention and without attention. Attention was introduced for sequence to sequence models initially in [1]. In our model, we use the attention mechanism similar to that described in [16]. In [16], the authors use hierarchical attention at the word level and at the sentence level, we remove the sentence part and only use the attention over words. The data preprocessing is the same as that in Task 1 and Task 2 for both of the datasets.

First, we train the model on the Hindi dataset. We do not use any pre-trained embeddings. The embeddings are randomly initialized and are trained along with the classifier. We keep the embedding dimension as 300. We use a single layer bidirectional network because of the dataset size being small,

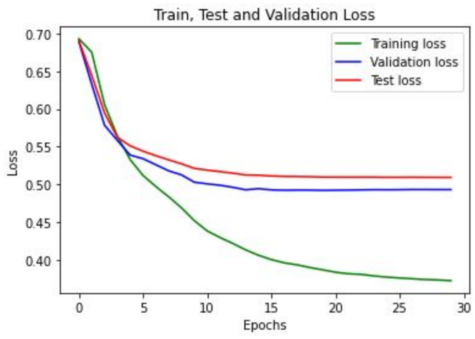


(a) Loss

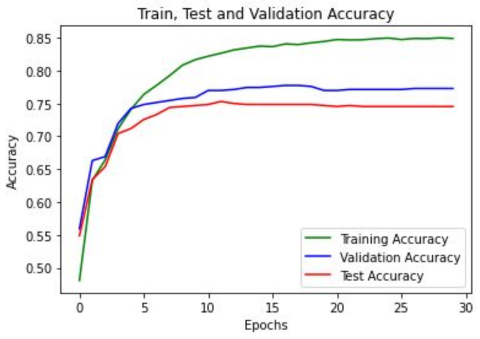


(b) Accuracy

Figure 4: Plots for loss and accuracy on the Hindi dataset using Bi-LSTM with attention based model.

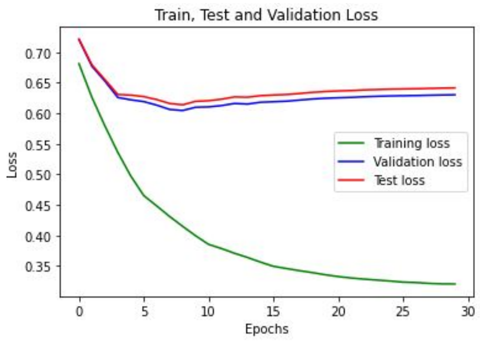


(a) Loss

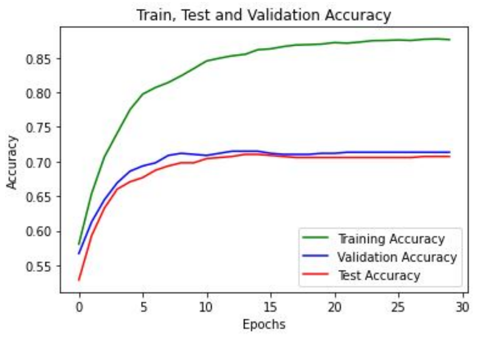


(b) Accuracy

Figure 5: Plots for loss and accuracy on the Bengali dataset using Bi-LSTM with attention based model.



(a) Loss



(b) Accuracy

Figure 6: Plots for loss and accuracy on the Bengali dataset using transfer learning with Bi-LSTM with attention.

we believe that one layer would be enough to capture the information. We keep the size of the hidden layer of the LSTM as 256. We use Adam optimizer with a learning rate of 2e-5. We use a step based learning rate scheduler which decays the learning rate by 0.5 after every 5 epochs. We use a dropout probability of 0.5 after concatenating the last hidden states of the forward and backward LSTMs for the model without attention. For the model with attention, we get the attention-weighted hidden state and pass them through the output layer. We train the models for 30 epochs. We follow the same process for the Bengali dataset and train Bi-LSTM based classifier with and without attention on that dataset with the same configuration. We will use the Bengali embeddings from this model and the Hindi classifier weights for transfer learning next.

Table 3: Accuracies for the CNN based model.

Experiment Type	Hindi dataset	Bengali Dataset
CNN based architecture	71.39%	71.49%
Hindi weights + no training	-	45.58%
Hindi weights + last 2 layers training	-	55.18%
Hindi weights + full training	-	66.92%

Table 4: Accuracies for the Bi-LSTM and Bi-LSTM with attention models.

Experiment Type	Hindi Dataset	Bengali Dataset
Bi-LSTM	75.75%	72.71%
Bi-LSTM hindi weights + full training	-	71.19%
Bi-LSTM with attention	**78.46%**	**74.54%**
Bi-LSTM with attention hindi weights + full training	-	70.73 %

For transfer learning, we follow one approach which performed best in the previous Task 2 experiments. We initialize the weights of the model to that of the Hindi Bi-LSTM based classifier and give input to it the embeddings trained while training the Bengali Bi-LSTM based classifier on the Bengali dataset. We then train this model and our aim is to see whether Hindi classifier weights provide a good initialization for the Bengali classifier. We do this for both the models-with attention and without attention.

3 Results and Analysis

We present the results on the test sets for all the tasks that we performed in Table 3 and Table 4. We also show the curves for the losses and accuracy for some of these results.

3.1 CNN based model

The results for our CNN based model for the Hindi and Bengali datasets are present in Table 3. The first row shows the result for training the model on the Hindi and Bengali dataset from scratch and the other three rows show results of transfer learning from Hindi to Bengali dataset. For the transfer learning schemes, we initialize the Hindi trained model weights with Bengali embeddings. 'Hindi weights + no training' corresponds to just testing the Hindi classifier on the Bengali dataset. 'Hindi weights + last 2 layers training' refers to just training the last 2 linear layers in the model. 'Hindi weights + full training' refers to training the model on Bengali with starting weights of Hindi classifier. The loss and accuracy curves of the CNN based models for the Hindi training, Bengali training and 'Hindi weights + full training' can be viewed in Figures 1, 2, 3 respectively.

We observe that the performance on Hindi language is better than on the Bengali language for the model. In the case of transfer learning, as we expected the Hindi model does not do well when it is directly tested on the Bengali dataset. We think this is because it has never seen Bengali and implicit similarity between these languages is very less, or even if the implicit similarity is there, the data and training steps are not enough to reflect that similarity. Also, adding an extra linear layer to the CNN and training the last layers does not work well. We believe that this is due to the representations given by Hindi trained CNN are not well representative of the Bengali data and do not provide good representations to the linear layers. We believe this is due to the small size of the Hindi dataset on which we have pre-trained the model. We also think that increasing the size of the model might also help in getting good representations. For transfer learning, we observe that one of our approaches works better on the Bengali dataset. The approach where we initialize the model to the Hindi weights and then train the full model(except the embedding layer) on the Bengali dataset works little bit comparable to training from scratch. We believe that even though the results are not bad, the initial Hindi weights are not providing a lot of information for the Bengali task as the accuracy is less than training from scratch. Although Figure 3 shows a steep decrease in loss in the beginning which might indicate that Hindi weights are providing a better than random initialization.

3.2 Bi-LSTM with/without attention

The results for our Bi-LSTM model with and without attention for the Hindi and Bengali datasets are present in Table 4. The first and third rows show the results for training the Bi-LSTM model and Bi-LSTM model with attention respectively on the Hindi and Bengali dataset. The 'Bi-LSTM hindi weights + full training' corresponds to initializing Bi-LSTM with Hindi weights and then training the full model on Bengali dataset. the 'Bi-LSTM with attention hindi weights + full training' corresponds to initializing Bi-LSTM with attention with Hindi weights and then training on Bengali dataset. The loss and accuracy curves of the Bi-LSTM with attention based models for the Hindi training, Bengali training and 'Bi-LSTM with attention hindi weights + full training' can be viewed in Figures 4, 5, 6 respectively.

We observe that the Bi-LSTM(with and without attention) based models perform well than CNN based models for both the languages with a large amount of accuracy gain observed for the Hindi language. For transfer learning, here also similar to the CNN case, training from scratch on Bengali is more helpful than using Hindi trained weights. We believe this is due to the same reason as for CNNs - the initial weights are not providing much information for further training on the Bengali dataset. It means that the pre-training on Hindi is not very helpful for the Bengali dataset. We also observe that adding attention provides significant gains, especially for Hindi. We also observe that the transfer learning on the Bengali dataset for Bi-LSTM with attention does not work very well. We believe this is due to the attention scores which have to be significantly changed from Hindi to Bengali. Whereas, adding attention improves the score on training the Bengali model from scratch.

4 Conclusion

We try out various models and transfer learning approaches on the Hindi and Bengali datasets for sentiment analysis. Bi-LSTM with attention models outperform the CNN based models initialized with word2vec trained embeddings from Task 1. While transfer learning does not show much gains for the models which we have tried, the success of transfer learning when pre-training is done on large models and with large datasets is widespread. The performance from our results can further be improved with more data in the test(target) language or with pre-training on a larger amount of data on larger models in the training(source) language.

References

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2016.

[2] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[4] Julian Martin Eisenschlos, Sebastian Ruder, Piotr Czapla, Marcin Kardas, Sylvain Gugger, and Jeremy Howard. Multifit: Efficient multi-lingual language model fine-tuning. *arXiv preprint arXiv:1909.04761*, 2019.

[5] Katharine Gelber and Luke McNamara. Evidencing the harms of hate speech. *Social Identities*, 22(3):324–341, 2016.

[6] Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*, 2018.

[7] Yoon Kim. Convolutional neural networks for sentence classification, 2014.

[8] György Kovács, Pedro Alonso, and Rajkumar Saini. Challenges of hate speech detection in social media. *SN Computer Science*, 2, 04 2021.

[9] Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. Imagenet classification with deep convolutional neural networks. *Neural Information Processing Systems*, 25, 01 2012.

[10] Pingping Lin and Xudong Luo. A survey of the applications of sentiment analysis. *International Journal of Computer and Information Engineering*, 14(10):334 – 346, 2020.

[11] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[12] Thomas Mandl, Sandip Modha, Prasenjit Majumder, Daksh Patel, Mohana Dave, Chintak Mandlia, and Aditya Patel. Overview of the hasoc track at fire 2019: Hate speech and offensive content identification in indo-european languages. In *Proceedings of the 11th Forum for Information Retrieval Evaluation*, FIRE '19, page 14–17, New York, NY, USA, 2019. Association for Computing Machinery.

[13] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[14] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.

[15] Karsten Müller and Carlo Schwarz. Fanning the flames of hate: Social media and hate crime. *SSRN Electronic Journal*, 01 2017.

[16] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489, San Diego, California, June 2016. Association for Computational Linguistics.

[17] Ye Zhang and Byron Wallace. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *arXiv preprint arXiv:1510.03820*, 2015.