IR3218-UI at SemEval-2020 Task 12: Emoji Effects on Offensive Language Identification

Sandy Kurniawan[1], Indra Budi[2], Muhammad Okky Ibrohim[3]
Faculty of Computer Science
Universitas Indonesia
Kampus UI, Depok, 16424, Indonesia
[1]sandy.kurniawan@ui.ac.id, {[2]indra, [3]okkyibrohim}@cs.ui.ac.id

Abstract

In this paper, we present our approach and the results of our participation in OffensEval 2020. There are three sub-tasks in OffensEval 2020, namely offensive language identification (sub-task A), automatic categorization of offense types (sub-task B), and offense target identification (sub-task C). We participated in sub-task A of English OffensEval 2020. Our approach emphasizes on how the emoji affects offensive language identification. Our model used LSTM combined with GloVe pre-trained word vectors to identify offensive language on social media. The best model obtained macro F1-score of 0.88428.

1 Introduction

Offensive language has become severe problems with the rapid growth of people using social media platforms such as Twitter. On social media, users can easily express their opinion about any topic which leads to some users posting offensive content while engaging in social media. The offensive contents sometimes are used to target another user, whether individual or particular groups based on the user's views. An automatic method of identifying offensive language is needed to prevent the spread of offensive content on social media.

Zampieri et al. (2020) organized the OffensEval 2020: Multilingual Offensive Language Identification in Social Media as Task 12 on SemEval-2020. OffensEval 2020 covers multilingual language consists of Arabic, Danish, English, Greek, and Turkish. The main focus of OffensEval 2020 is to solve problems on how to identify the offensive language (sub-task A), offense types categorization (sub-task B), and offense target identification (sub-task C). Sub-task A ran for all languages while sub-task B and sub-task C only ran for English.

Our approach used a neural network model based on Long Short-Term Memory (LSTM) in order to identify offensive language on English tweets. This paper mainly describes our submission for OffensEval-2020 Sub-task A, but our model can be further developed to learn other sub-tasks.

This paper is organized as follows. Related work has been discussed in section 2, dataset and methodology have been described in section 3. The experiment results and discussions described in section 4. Section 5 concludes the research and the future works of the research.

2 Related Work

Many research on offensive language identification has been done. Wiedemann et al. (2018) studied the potential of transfer learning in automatic offensive language detection. They employ many transfer learning scenarios to train their neural network. The results proved that transfer learning improves offensive language detection performance. Ramakrishnan et al. (2019) used an ensemble model based on logistic regression and tree-based model to identify offensive language on SemEval-2019 Task 6. Char n-grams, word n-grams, part of speech and GloVe embedding were used as features. By combining 5 different models with the varying feature set as input, an ensemble model was built. Using the ensemble model, the offensive language identification obtained an accuracy of 0.80 and macro F1 score of 0.74.

Proceedings of the 14th International Workshop on Semantic Evaluation, pages 1998–2005
Barcelona, Spain (Online), December 12, 2020.

Recently, deep learning approach was used in many text classification problems. Xiao et al. (2018) compared several classification models such as K-Nearest Neighbors (KNN), convolutional neural network (CNN) and LSTM for patent text classification. Their results showed that LSTM outperformed other models with 93.48% accuracy. LSTM was also used by Makarenkov et al. (2019) for political perspective identification in online news articles. They experimented using various hyper-parameter setting for the LSTM network such as word embedding source, memory size, word cutoff and batch size. Their best result obtained Area under the ROC Curve (AUC) score of 0.966.

Ayvaz and Shiha (2017) conducted a study about emoji usage in social media and its impact on sentiment analysis. Their study reveals that emoji have a significant effect on improving the sentiment scores both on the positive and negative opinions. Another emoji-related study was done by Singh et al. (2019). This study compared two emoji-handling strategies, the first is to use separate embedding for emojis, and the other is to use textual replacement for emojis combine with word embedding. Emoji2Vec (Eisner et al., 2016) is used as the emoji embedding, and the word embedding used is provided by Baziotis et al. (2018) using Word2Vec trained on 550 million tweets. The results showed that using textual replacement for emoji produced better performance than using separate embedding for emojis.

Based on the previous description, in this study, we want to identify offensive language on Twitter using LSTM as the model classification. LSTM was used as the model classification since it showed good performance in text classification task based on the study done by Xiao et al. (2018) and Makarenkov et al. (2019). Based on the study of Singh et al. (2019), we want to implement the same strategy in order to study the effect of emojis on offensive language identification. The difference is that the word embedding used for our study is provided by GloVe (Pennington et al., 2014), which provide pre-trained Twitter corpus more than the one provided by Baziotis et al. (2018).

3 Dataset and Methodology

3.1 Dataset

The dataset used in this task is Semi-Supervised Offensive Language Identification Dataset (SOLID), a semi-supervised annotated dataset based on OLID (Zampieri et al., 2019) as the seed dataset, provided by Rosenthal et al. (2020). The dataset consists of 9089140 tweets, and the testing data consists of 3887 tweets. The dataset example is shown in Figure 1. The dataset provided information about the tweet id, the tweet text, average agreement score, and standard deviation agreement score. The average agreement score is the average of confidences obtained from several supervised models for the tweet belong to the offensive class (OFF). The standard deviation agreement score is the confidence standard deviation of average agreement score for each tweet.

	id	text	average	std
0	1159533701283352576	First time I heard his name in camp, he seems ...	0.195773	0.187379
1	1159533703522992128	When I go to drink with Tsubaki he would alway...	0.262401	0.145998
2	1159533703758061570	@USER His ass need to stay up 😂😂	0.833391	0.140628

Figure 1: Dataset Example

Since the dataset did not provide the final label, we have to decide the final label using our method. In addition, the dataset also consists of tweets that use languages other than English. Based on these, we did not use all the provided dataset as training data in our research.

The first process in selecting training data is to detect the language of the tweet. We decided to use language detection because when we explored the dataset, we found that some of the tweet languages were not in English. Thus, we used language detection and only kept English tweet in our dataset. For detecting the tweet's language, we used whatlangid[1] python library. The tweets in English are selected for

[1]https://pypi.org/project/whatlangid/

the next process. The following process is to label the tweet into offensive class (OFF) or not-offensive class (NOT). The tweet is labelled into offensive class if its average agreement score is more than 0.5. As for the not-offensive class, we select the tweet with the lowest average agreement score after sorting the tweet's ranking based on the average agreement scores. Because the amount of offensive tweet is less than the not-offensive tweet, we only keep the amount of not-offensive as many as the offensive tweets to balance the number of instances. The selected data then used as training data.

	OFF	NOT	Total
Training	1372913	1372913	2745826
Testing	1080	2807	3887

(a) Dataset Details

	OFF	NOT	Total
Training	265787	297509	563296
Testing	168	634	802

(b) Dataset with Emoji Details

Table 1: Dataset with Emoji Details

Table 1a shows the detail amount of data used in this research. From 9089140 tweets provided in the dataset, we selected 2745826 tweets as the training data. The training data consists of 1372913 for both offensive and not-offensive class tweets. For classification model development, 20% of the training data was used as the validation set. Thus, 549165 tweets were used as validation data. As for testing data, a total of 3887 tweets consists of 1080 offensive class tweets and 2807 not-offensive tweets.

As we were trying to understand the effect of emoji, Table 1b shows our result in detecting the emoji contained in the data. From the training data, 563296 tweets contained emoji, consist of 265787 offensive class tweets and 297509 not-offensive class tweets. There are 802 tweets with emoji in the testing data, 168 tweets from offensive class and 634 tweets from not-offensive class.

3.2 Methodology

In this subsection, we described our approach for OffensEval 2020. The research flowchart of our approach is shown in Figure 2. The research flowchart consists of some processes, including data preprocessing, model building, and testing.

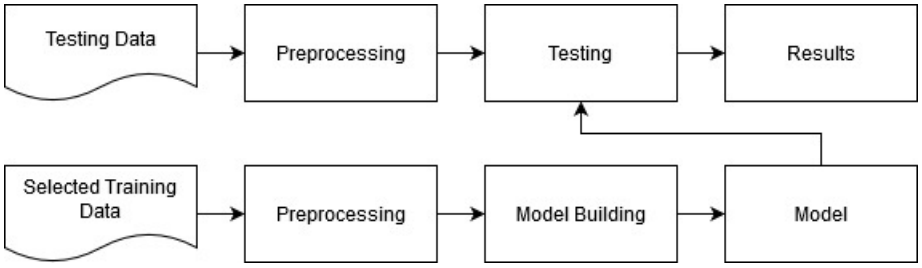

Figure 2: Research Flowchart

3.2.1 Data Preprocessing

Data preprocessing was meant to prepare the data so that it can be used in the next process. The output of preprocessing made it available for machine learning to understand the data. For our experiment, we used several preprocessing methods, including emoji handling, punctuation removal, case folding, stopwords removal, and word stemming.

The emoji can be handled in two different ways. The first is to remove the emoji completely from the data, and the later is to replace the emoji into a phrase that represents the emoji. In order to replace the emoji with phrases that represent the emoji, we used the emoji[2] python library provided by Taehoon Kim and Kevin Wurster. The emoji replacement phrase example is shown in Table 2. Then, we cleaned the data by doing punctuation removal, case-folding the data into lower case, and removing stopwords in the

[2]https://pypi.org/project/emoji/

data. Following this process, a stemming process is applied to the tweets using the snowball stemmer algorithm (Porter, 2001). For stopwords removal and stemming process, we used NLTK[3] python library provided by (Bird et al., 2009).

Emoji	Replacement Phrase
😀	:grinning_face:
😂	:face_with_tears_of_joy:
😤	:face_with_steam_from_nose:

Table 2: Example of Phrase Replacement for Emojis

3.2.2 Model Building

After the training data was preprocessed, the training data was used to train our neural network model. Several layers are used in our neural network model such as embedding layer, LSTM layer, pooling layer, dropout layer and fully connected layer. The structure of the classification model used in this research is shown in Figure 3.

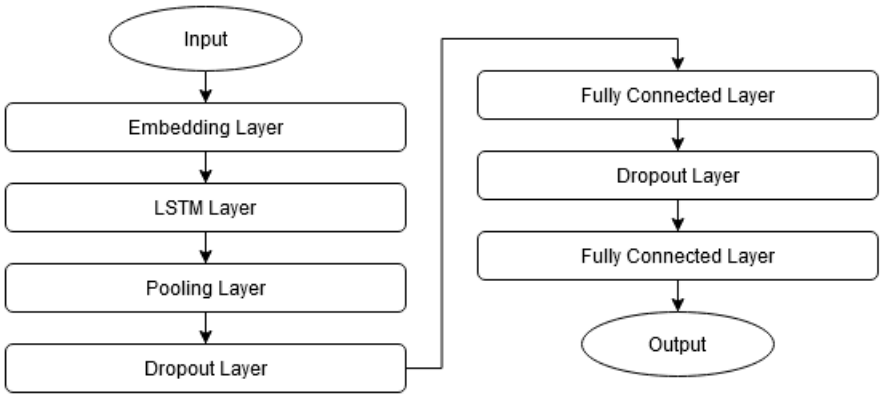

Figure 3: Model Structure

The first layer is the embedding layer. In this layer, the data representation was changed into a dense vector representation. The vector can be learned from scratch using the training data or build from pre-trained word vectors from another data source. We used pre-trained word vectors provided by Pennington et al. (2014) called GloVe. GloVe has several pre-trained word vectors that can be used which are created based on Wikipedia data, Twitter data and Common Crawl data. Since the training data was tweets, we used the Twitter pre-trained GloVe word vectors. As the emoji handling is done first before other preprocessing step, the replacement phrase also gets preprocessed like any other text in the data. This way, the replacement phrase won't become out-of-vocabulary features from the GloVe Twitter pre-trained training corpus. The replacement phrase will become features that represent the emoji in the data.

Then the vectors were used as input of the LSTM layer. Introduced in 1991, LSTM (Hochreiter and Schmidhuber, 1997) has been used in many tasks such as sentiment classification (Rao et al., 2018), image captioning (Xu et al., 2017), and language translation (Guo et al., 2018). We used LSTM in our architecture because LSTM is believed able to overcome the vanishing gradient problem. LSTM solves the problem by utilizing its component such as input, output, and forget gates, to manage which information needed to keep or forget.

The next layer is the pooling layer. Pooling layer was used to reshape the 3D tensor into a 2D one. We used a global average pooling to reduce complexity. The next four-layer consists of two pair of dropout layer and fully connected layer. We used dropout (Srivastava et al., 2014) in order to avoid overfitting. Dropout is done by randomly dropping neurons on the network with their connection in each iteration on

[3]http://nltk.org/

the training phase, so it is equally the same as training different neural networks. In our model, we applied a dropout rate of 0.1 in the dropout layer. Fully connected layer are layers where the inputs are connected to every activation unit of the next layer. This layer can be used as feature extraction or classification. We used two fully connected layers, the first layer was used as feature extraction with ReLU as the activation function, and the later was used to classify the dataset with Sigmoid activation function.

The neural network model training process was running for ten epochs. We tried to use 15 as the epochs training, but the model validation results using validation data showed signs of overfitting, so we decided to use ten epochs as our training epochs. Binary cross entropy was used as loss function, and Adam (Kingma and Ba, 2014) was used as the optimizer with learning rate of 0.001. The training process was set to save the model with the best validation results within ten epochs. The model will be used as the classification model in the testing process.

3.2.3 Testing

The testing process started after the classification model has been determined. The testing data was preprocessed first using the same method as the preprocessing process done for training data. After it was preprocessed, the classification model predicted the class for each testing data. Then the result was evaluated using precision, recall, and macro F1-score.

4 Experiments Results and Discussion

This research was done to explore the effect of emoji on offensive language identification. To do that, we proposed two preprocessing scenarios for offensive language identification. In general, each scenario follows the following processes: emoji handling, punctuation removal, case folding, stopwords removal, and word stemming. The difference is that in the first scenario, the emoji will be handled by removing them from the data while in the second scenario, the emoji will be handled by replacing the with phrases that represent the emoji. We used Macro F1-score to evaluate the performance of our results as it was the standard evaluation score requested by OffensEval 2020. We also evaluate the results in precision and recall to understand the results better. The evaluation metrics were computed using Scikit-Learn[4] python library (Pedregosa et al., 2011).

		PREDICTED		
		NOT	OFF	ALL
ACTUAL	NOT	2251	556	2807
	OFF	2	1078	1080
	ALL	2253	1634	3887

(a) Scenario 1

		PREDICTED		
		NOT	OFF	ALL
ACTUAL	NOT	2411	396	2807
	OFF	1	1079	1080
	ALL	2412	1475	3887

(b) Scenario 2

Table 3: Result Confusion Matrix

OffensEval 2020 only announce the F1-score and ranking of the participants' last submission for the competition. However, the gold labels for the testing data have been provided after the competition ends. We could analyze our approach based on this provide gold labels. We used the confusion matrix to explore the distribution of the classification for each class. The confusion matrix is shown in Table 3. From the confusion matrix, we calculate precision, recall and F1-score for each class and the macro average precision, recall and F1-score for each scenario. The evaluation metrics for each scenario is shown in Table 4.

Based on the results, our approach shows good performance by successfully classifying almost all offensive tweets correctly, with only two misclassifications in scenario 1 and one misclassification in scenario 2. However, our method is too sensitive towards offensive language. This is indicated by the many misclassifications of not-offensive tweets which are considered as offensive tweets.

[4]https://scikit-learn.org/

	Prec	Rec	F1
NOT	0.99911	0.80192	0.88972
OFF	0.65973	0.99814	0.79439
Macro	0.82942	0.90003	0.84206

(a) Scenario 1

	Pre	Rec	F1
NOT	0.99958	0.85892	0.92393
OFF	0.73152	0.85892	0.84461
Macro	0.86555	0.92899	0.88428

(b) Scenario 2

Table 4: Sub-task A Results.

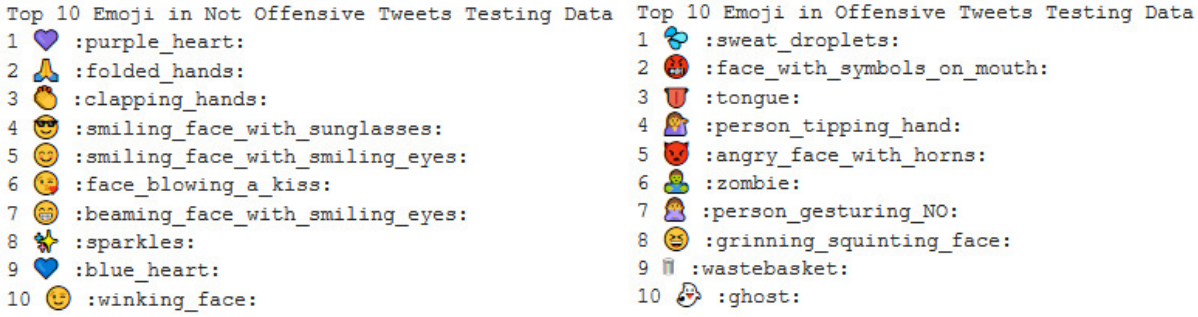

Figure 4: Top-10 Emoji for Each Class

In order to analyze the effect of emoji on offensive language identification, we count the emoji occurrences in the dataset. The emojis were ranked for its appearance in each class. The top-10 ranked emoji in the testing dataset is shown in Figure 4. The top-10 emoji in not-offensive class consists of positive emoji such as heart-shaped emoji, hand-gestures emoji, and smiling-face emoji while for the offensive class top-10 emoji consists of negative emoji such as angry-face emoji, and emoji with negative interpretation such as sweat droplets, tongue, and zombie. The various emojis used in each class can help the classifier distinguish between the two classes.

5 Conclusion

Offensive language identification is crucial in nowadays era. Almost in every social media platform, offensive language exists and disturbing its users. OffensEval 2020 is organized to explore and improve on offensive language identification. This paper presents our system description on participating at OffensEval 2020/ SemEval-2020 Task 12 sub-task A. We studied the effect of emoji in offensive language identification using LSTM and GloVe embedding.

We can conclude that replacing emoji into phrase improve the model performance. The improvement is indicated by an increase in the F1-score between scenario 1 (emoji removal) and scenario 2 (emoji replacement). Scenario 1 obtained the F1-score value of 0.84206 while scenario 2 obtained a better F1-score, with a value of 0.88428. This improvement occurs because the dataset has more features to analyze from the emoji replacement phrase where each emoji has a tendency towards certain classes. However, our models were too sensitive with offensive language. This is proven by the number of misclassification obtained for both scenarios. Out of 2807 not-offensive tweets, 556 tweets in scenario 1 and 396 tweets in scenario 2 were misclassified as offensive tweets by our models.

For future work, we want to explore the dataset selection method to improve our results. The dataset used in this research showed that the model was too sensitive in detecting offensive language. In addition, as OffensEval 2020 provides multilingual offensive language dataset, we plan to study how to identify offensive language on a multilingual dataset.

Acknowledgements

The authors gratefully thank Universitas Indonesia for the International Publication (PUTI Prosiding) Grants No. NKB-877/UN2.RST/HKP.05.00/2020 Year of 2020.

References

Serkan Ayvaz and Mohammed Shiha. 2017. The effects of emoji in sentiment analysis. *International Journal of Computer and Electrical Engineering*, 9:360–369, 01.

Christos Baziotis, Nikos Athanasiou, Pinelopi Papalampidi, Athanasia Kolovou, Georgios Paraskevopoulos, Nikolaos Ellinas, and Alexandros Potamianos. 2018. Ntua-slp at semeval-2018 task 3: Tracking ironic tweets using ensembles of word and character level attentive rnns. In *Proceedings of The 12th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT*, pages 613–621.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*. O'Reilly Media, Inc., 1st edition.

Ben Eisner, Tim Rocktäschel, Isabelle Augenstein, Matko Bošnjak, and Sebastian Riedel. 2016. emoji2vec: Learning emoji representations from their description. In *Proceedings of The Fourth International Workshop on Natural Language Processing for Social Media, SocialNLP@EMNLP 2016*, pages 48–54.

Dan Guo, Wengang Zhou, Houqiang Li, and Meng Wang. 2018. Hierarchical lstm for sign language translation. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

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

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Victor Makarenkov, Ido Guy, Niva Hazon, Tamar Meisels, Bracha Shapira, and Lior Rokach. 2019. Implicit dimension identification in user-generated text with lstm networks. *Information Processing & Management*, 56(5):1880–1893.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, 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 D 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.

M. F. Porter. 2001. Snowball: A language for stemming algorithms, October.

Murugesan Ramakrishnan, Wlodek Zadrozny, and Narges Tabari. 2019. Uva wahoos at semeval-2019 task 6: Hate speech identification using ensemble machine learning. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 806–811.

Guozheng Rao, Weihang Huang, Zhiyong Feng, and Qiong Cong. 2018. Lstm with sentence representations for document-level sentiment classification. *Neurocomputing*, 308:49–57.

Sara Rosenthal, Pepa Atanasova, Georgi Karadzhov, Marcos Zampieri, and Preslav Nakov. 2020. A large-scale semi-supervised dataset for offensive language identification.

Abhishek Singh, Eduardo Blanco, and Wei Jin. 2019. Incorporating emoji descriptions improves tweet classification. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2096–2101.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

Gregor Wiedemann, Eugen Ruppert, Raghav Jindal, and Chris Biemann. 2018. Transfer learning from lda to bilstm-cnn for offensive language detection in twitter. *arXiv preprint arXiv:1811.02906*.

Lizhong Xiao, Guangzhong Wang, and Yang Zuo. 2018. Research on patent text classification based on word2vec and lstm. In *Proceedings - 2018 11th International Symposium on Computational Intelligence and Design, ISCID 2018*, volume 1, pages 71–74.

Kaisheng Xu, Hanli Wang, and Pengjie Tang. 2017. Image captioning with deep lstm based on sequential residual. In *2017 IEEE International Conference on Multimedia and Expo (ICME)*, pages 361–366. IEEE.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019. Predicting the Type and Target of Offensive Posts in Social Media. In *Proceedings of NAACL*.

Marcos Zampieri, Preslav Nakov, Sara Rosenthal, Pepa Atanasova, Georgi Karadzhov, Hamdy Mubarak, Leon Derczynski, Zeses Pitenis, and Çağrı Çöltekin. 2020. SemEval-2020 Task 12: Multilingual Offensive Language Identification in Social Media (OffensEval 2020). In *Proceedings of SemEval*.