# TECHSSN at SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Tweets using Deep Neural Networks

**Logesh Balasubramanian, Harshini Sathish Kumar,**
**Geetika Bandlamudi, Dyaneswaran Sivasankaran,**
**Rajalakshmi Sivanaiah, Angel Deborah Suseelan,**
**Sakaya Milton Rajendram, Mirnalinee Thanka Nadar Thanagathai**
Department of Computer Science and Engineering
SSN College of Engineering
Chennai 603 110, Tamil Nadu, India
{logesh16054, harshini16040}@cse.ssn.edu.in,
{geetika16032, dyaneswaran16028}@cse.ssn.edu.in,
{rajalakshmis, angeldeborahs}@ssn.edu.in
{miltonrs, mirnalineett}@ssn.edu.in

## Abstract

Task 6 of SemEval 2019 involves identifying and categorizing offensive language in social media. The systems developed by TECHSSN team uses multi-level classification techniques. We have developed two systems. In the first system, the first level of classification is done by a multi-branch 2D CNN classifier with Google's pre-trained Word2Vec embedding and the second level of classification by string matching technique supported by offensive and bad words dictionary. The second system uses a multi-branch 1D CNN classifier with Glove pre-trained embedding layer for the first level of classification and string matching for the second level of classification. Input data with a probability of less than 0.70 in the first level are passed on to the second level. The misclassified examples are classified correctly in the second level.

## 1 Introduction

The growth of social media networks in recent days has been phenomenal and Twitter is no exception. However, this rapid growth of social media also poses a serious challenge of maintaining ethics in social media because of the degradation of moral values in society. Offensive micro tweets are generated on a daily basis targeting a particular person, organization, race, caste, community, religion, gender and so forth. For this reason, our task for the SemEval2019 (Zampieri et al., 2019b) mainly focuses on the detection of offensive language in tweets and classify them.

In Subtask-A, we tried to classify the tweets into two classes, namely offensive and non-offensive. The offensive tweets in the Subtask-A are then categorized in Subtask-B into targeted and untargeted tweets, where targeted tweets are aimed at a specific person, organization, religion or political parties. Further, Subtask-C deals with the fine-grained classification of offensive tweets into three classes viz person, organization and others.

The training dataset provided by the organizers contains 13240 tweets. The given dataset is used as the preliminary dataset to train our model. In addition, Impermium dataset from Kaggle and TRAC dataset are added to improve the accuracy of the model. We have also used a dictionary of offensive words in the second level of classification. Manually classifying the tweets is ambiguous and highly subjective, and is one of the biggest challenges. The mix of colloquial slang in tweets, veiled references, missing data, usage of symbols and emojis are further hurdles that lowered the prediction accuracy (Founta et al., 2018).

## 2 Related Work

Many researchers in the field of Artificial Intelligence and Natural Language Processing have been working to detect offensive speech in tweets using sentiment analysis. Pang et al. (2002) used a three level classification system with Naive Bayes classifier in the first level, Multinomial Updatable Naive Bayes in the second level and a rule based classifier named DTNB in the third level. The second level of classification increased the accuracy by 7% while the third level results showed an improvement in performance by a 6% rise in accuracy. The results were boosted by the usage of an insulting and abusive language dictionary.

Stammbach et al. (2018) developed an ensemble model of Convolutional Neural Network (CNN), Recurrent Neural Network (RNN) and CNN+RNN that gave a macro averaged F1-score of 77.7%, 78.6% and 77.6% respectively on 10-



*Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019)*, pages 753–758
Minneapolis, Minnesota, USA, June 6–7, 2019. 

fold cross validation. It is stated that the correct words generated by the spell-checker did not occur in the embeddings and this might be one of the reasons for the low performance of the model. Che et al. (2017) presented a review of the recent deep neural networks used for text classification and a comparison of word embeddings to Support Vector Machine (SVM) classifiers. It points out that a critical issue with CNN is the restriction of a fixed input size and hence the inability to handle sentences of variable length as input, and therefore, focuses on RNN. It is also mentioned that n-grams with syntactic and semantic information achieve significantly better results than the standard n-grams.

(Le and Mikolov, 2014) uses paragraph vectors instead of Bag-of-Words (BOW) feature representation and reduces the classification error by approximately 39%. Dinakar et al. (2011) and ElSherief et al. (2018) discuss the problem of overlapping classes in target identification. Malmasi and Zampieri (2018) discuss the various challenges in discriminating hate, offensive and non-offensive text using ensemble techniques and stacked generalization meta learning methods.

Razavi et al. (2010) discuss about the multilevel classification for flame detection using complement naive Bayes on first level to select discriminative features and multinomial updatable naive Bayes classifier on second level to enhance the model with new features for adaptive learning. They have used rule-based classifier named DTNB (Decision Table/Naive Bayes hybrid classifier) as the last level to classify the text into Flame/Not.

## 3 Methodology and Data

The task of classifying offensive tweets is difficult as it needs to discover the intention of the user. Moreover, people who follow the chatting convention use offensive words to express their feelings. The architecture diagram for the offensive text classification is shown in Figure 1.

### 3.1 Acquiring Datasets

The classifier can make a well-informed decision if we procure and supply more data to it. For good performance, deep learning requires sufficiently large amount of data. Therefore, in addition to the dataset given by Zampieri et al. (2019a), we also compiled a variety of datasets for our tweet classification. We have added the TRAC training

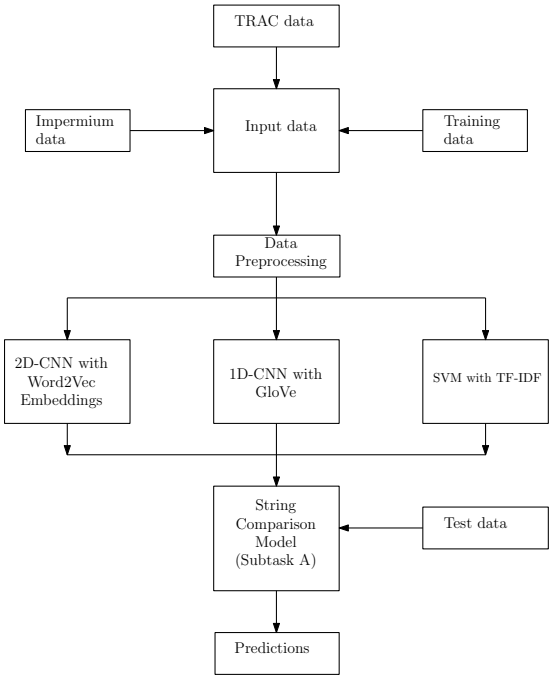


Figure 1: Architecture of Proposed System

dataset (Kumar et al., 2018) consisting of online posts from Facebook, the Impermium Dataset of Kaggle (Impermium, 2013), and used a comprehensive list of known offensive words banned by Google in all their different forms. TRAC dataset is based on multiclass classification (OAG, CAG, NAG). We have considered OAG and CAG class labels as OFF label and NAG as NOT for the subtask A.

### 3.2 Data Preprocessing

Data preprocessing critical for the success of any machine learning solution. The given dataset shows many signs of irregularities which is a classic signature of any collection of tweets. Normalizing the data involves flattening the dimensions of data into textual form. The dataset is cleaned and processed using functions from NLTK and spacy toolkit.

During preprocessing, we

(a) remove URLs,
(b) annotate emojis, emoticons,
(c) convert uppercase to lowercase,
(d) expand contractions,
(e) remove stopwords,
(f) remove special characters,
(g) remove accented characters,
(h) reduce lengthened words,
(i) lemmatize text, and

(j) remove extra whitespace

Among the steps listed above, step (e) is omitted for subtask-B and subtask-C, since stopwords are significant for target identification.

### 3.3 Model Description

We classified the text using these three models and compare the results:

1. 2D-CNN with Word2Vec Learned Embeddings
2. 1D-CNN with GloVe
3. SVM with BOW

#### 3.3.1 2D-CNN with Word2Vec Learned Embeddings

Even though it is unconventional to use a two dimensional convolutional network to work on text sequences, it has proved its usefulness quite satisfactorily (Prusa and Khoshgoftaar, 2017). The model is built by taking the pre-trained Google's Word2Vec weights and learn more with the following additional layers.

1. Input layer
2. Embedding layer
3. Convolutional layer with kernel size 2
4. Convolutional layer with kernel size 3
5. Convolutional layer with kernel size 4
6. Respective pooling layers for CNN layers
7. Fully connected dense layer
8. Output layer

We have used an embedding layer which learns the weights of the embedding matrix during training. The bigrams, trigrams and fourgrams of the words are obtained by applying filters and kernels of the right size on the embedding layer output. After applying the filters, a max pooling operation is performed on each CNN layer to scale down the output vectors into dense feature vectors, each of size 100. The three dense vectors are concatenated and flattened into a single dense vector of size 300 in the fully connected layer. The final output is obtained through a dense layer with 2 output units.

The main ideabehind the concatenation of the word grams is to compute n-grams in parallel, add them together and extract as much information as possible from the vectors. This enables the classifier to learn and understand the relationships between the underlying words. The parameters for the model are as set as follows: sequence length of the model is 43, learning rate is set as 0.001 and dropout is set as 0.5, Softmax activation function for output layer and Relu activation function in other layers.

#### 3.3.2 1D-CNN with GloVe

GloVe embeddings with 1 million word vectors of 200 dimension from twitter is used as an alternative method to create the embedding matrix of the embedding layer in the network. We used a conventional convolutional neural network in single dimension and extracted the skip-grams in parallel by using filters of size 2, 3 and 4, each vector of size 100, in three different branches and concatenated them to form one flattened dense vector of size 300. We have used 100 filters and dropout value as 0.2. Softmax activation function is used in output layer and Relu function is used in other layers. To increase the representational power of the neural network, a couple of dense layers are added before the final output layer. This model has fewer trainable parameters, takes less time to train, yet performs on par with 2D-CNN.

#### 3.3.3 SVM with Bag-Of-Words

We implemented a Support Vector Machine using Term Frequency-Inverse Document Frequency (TF-IDF) model and found it to be as good as neural networks. The input document is converted into binary vectors using TF-IDF method. These feature vectors are fed as input to the SVM which uses a linear function as the kernel. The confusion matrix for SVM is shown in Table 1. Linear SVM is found to be better for text classification since most of the text classification problems are linearly separable (Joachims, 1998). For our model, BOW vector size is 98807.

| | | Actual | |
|---|---|---|---|
| | | OFF | NOT |
| Predicted | OFF | 689 | 618 |
| | NOT | 75 | 1880 |

Table 1: Linear SVM with TF-IDF model

#### 3.3.4 Logistic Regression and RNN Models

We also trained Logistic Regression (Davidson et al., 2017) and RNN + LSTM (Long Short-Term Memory) models (Pitsilis et al., 2018). Table 2 shows macro average F1-scores for the various models developed. These models do not perform well compared to CNN and SVM. We used 75%

| Model Used | F1 (Macro) |
|---|---|
| 1D-CNN with GloVe | **0.751** |
| 2D-CNN with Word2Vec | 0.740 |
| Linear SVM with TF-IDF | 0.722 |
| RNN+LSTM and GloVe | 0.719 |
| Logistic Regression with BoW | 0.703 |
| RNN+LSTM and Word2Vec | 0.702 |

Table 2: Comparison of F1-scores of the Models Used

of the instances for training and 25% for testing the accuracy of the models.

The Logistic Regression using count vectorization model predicts more offensive tweets correctly than the previous models specified here. However, since the model could not classify non-offensive tweets effectively, its performance is low. The reason is that the bag of words model using count vectorization does not take the context or the semantics of the tweet into account and hence contextually non-offensive tweets with an offensive word in them are misclassified. Table 3 shows the confusion matrix for logistic regression model.

| | | Actual | |
|---|---|---|---|
| | | OFF | NOT |
| Predicted | OFF | 831 | 476 |
| | NOT | 250 | 1705 |

Table 3: Logistic Regression using BoW

#### 3.3.5 Second Level Classification for Subtask-A

Most of the models we have described in the paper classify non-offensive tweets more correctly than offensive ones. To overcome the misclassification of offensive tweets, a second level of classification using string comparison model is done. Tweets predicted offensive with a probability less than 0.70 are passed on to a string comparison model which checks for any occurrence of offensive words. As an aid, a dictionary of 1384 words is constructed with the offensive words banned by Google and a list of bad words. This two level classification system proves to be very effective and increases the performance of the system by 5% in the macro average F1-score. The second level of classification is used to find whether the tweet is offensive or not. There is no need for a second level in subtasks B and C since they do not involve this classification.

## 4 Results

Table 4 shows the accuracy (Acc) results for subtask A. Three models were developed and tested for the given dataset. 2D-CNN model with enhanced dataset has better F1 score for both offensive and non-offensive tweets in comparison to the 2D-CNN with given dataset alone and 1D-CNN. 1D-CNN has better accuracy than other models. Since the given dataset is biased, some models classify non-offensive tweets more correctly than offensive tweets. Such a model has better accuracy than other models, but lower F1 score.

| System | F1 (macro) | Acc |
|---|---|---|
| All NOT baseline | 0.4189 | 0.7209 |
| All OFF baseline | 0.2182 | 0.2790 |
| 2D-CNN (28000 Data) | **0.7382** | 0.8058 |
| 2D-CNN (13240 Data) | 0.7351 | 0.8035 |
| 1D-CNN | 0.7281 | **0.8174** |

Table 4: Results for Subtask A

Table 5 shows the results for subtask B. We developed SVM and 2D-CNN model for subtask B. SVM classifies tweets into targeted and untargeted ones more effectively than 2D-CNN model.

| System | F1 (macro) | Accuracy |
|---|---|---|
| All TIN baseline | 0.4702 | 0.8875 |
| All UNT baseline | 0.1011 | 0.1125 |
| SVM with TF-IDF | **0.6602** | **0.8208** |
| 2D-CNN | 0.5588 | 0.775 |

Table 5: Results for Subtask B

Table 6 shows the results for subtask C. SVM with TF-IDF, 1D-CNN with GloVe and 2D-CNN with learned Word2Vec embeddings are used to build models for subtask C. 1D-CNN model has better F1 macro score than other models.

| 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 |
| SVM with TF-IDF | 0.5095 | **0.6808** |
| 1D-CNN | **0.5633** | 0.6714 |
| 2D-CNN | 0.4846 | 0.6479 |

Table 6: Results for Subtask C

The confusion matrix for the best model in subtasks A, B and C are shown in Figure 2, 3 and 4 respectively.

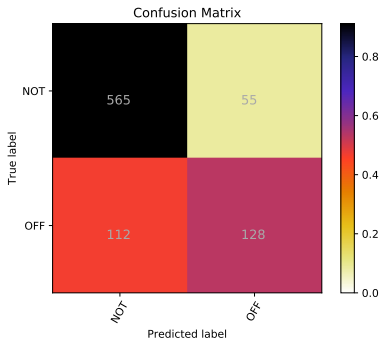


Figure 2: SubTask A: Confusion Matrix for 2D-CNN with Word2Vec embeddings

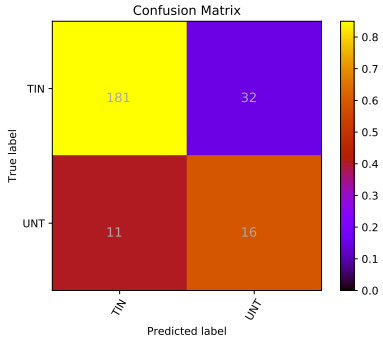


Figure 3: Subtask B: Confusion Matrix for SVM with bag of words model

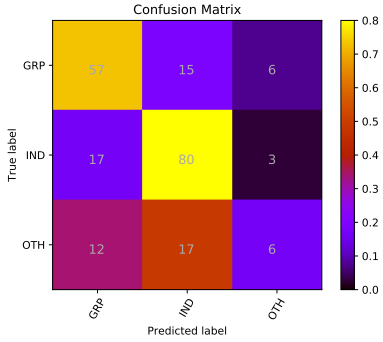


Figure 4: SubTask C: Confusion Matrix for 1D-CNN with GloVe

## 5 Conclusion

We have built three models for the tasks of offensive language detection and classification in tweets. The models are 2D-CNN with Word2Vec learned embeddings, 1D-CNN with GloVe and SVM with TF-IDF. All the models use data preprocessed with NLTK, which we think is an important factor for improved accuracy. In addition, we also made use of a dictionary of offensive and banned words for a second level of classification.

Meaning of a tweet varies with an individual's perception and cannot be judged by simple conventional models. This is one reason for the reduced precision of classification. The concepts of irony, sarcasm, humor and other tones of a conversation are too intuitive and implicit for the models to detect them accurately. We intend to investigate further by adding multiple hidden layers and building complex network structure which will, in parallel, look for the tell-tale signs of the target tone of the tweets.

1D-CNN model achieved less F1-score in the target identification (subtask C) than in subtasks A and B, due to smaller dataset, which can improved by augmenting the dataset.

## References

Hao Che, Susan McKeever, and Sarah Jane Delany. 2017. Abusive text detection using neural networks. In *AICS Conference, Dublin Institute of Technology*.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated Hate Speech Detection and the Problem of Offensive Language. In *Proceedings of ICWSM*.

Karthik Dinakar, Roi Reichart, and Henry Lieberman. 2011. Modeling the detection of textual cyberbullying. In *The Social Mobile Web*, pages 11–17.

Mai ElSherief, Vivek Kulkarni, Dana Nguyen, William Yang Wang, and Elizabeth Belding. 2018. Hate Lingo: A Target-based Linguistic Analysis of Hate Speech in Social Media. *arXiv preprint arXiv:1804.04257*.

Antigoni-Maria Founta, Constantinos Djouvas, Despoina Chatzakou, Ilias Leontiadis, Jeremy Blackburn, Gianluca Stringhini, Athena Vakali, Michael Sirivianos, and Nicolas Kourtellis. 2018. Large Scale Crowdsourcing and Characterization of Twitter Abusive Behavior. *arXiv preprint arXiv:1802.00393*.

Impermium. 2013. Detecting Insults in Social Commentary. https://www.kaggle.com/c/detecting-insults-in-social-comme\ntary/data/.

Thorsten Joachims. 1998. Text categorization with support vector machines: Learning with many relevant features. In *European conference on machine learning*, pages 137–142. Springer.

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 Cyberbulling (TRAC)*, Santa Fe, USA.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196.

Shervin Malmasi and Marcos Zampieri. 2018. Challenges in discriminating profanity from hate speech. *Journal of Experimental & Theoretical Artificial Intelligence*, 30(2):187–202.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics.

Georgios K Pitsilis, Heri Ramampiaro, and Helge Langseth. 2018. Detecting offensive language in tweets using deep learning. *arXiv preprint arXiv:1801.04433*.

Joseph D Prusa and Taghi M Khoshgoftaar. 2017. Deep neural network architecture for character-level learning on short text. In *The Thirtieth International Flairs Conference*.

Amir H Razavi, Diana Inkpen, Sasha Uritsky, and Stan Matwin. 2010. Offensive language detection using multi-level classification. In *Canadian Conference on Artificial Intelligence*, pages 16–27. Springer.

Dominik Stammbach, Azin Zahraei, Polina Stadnikova, and Dietrich Klakow. 2018. Offensive language detection with neural networks for germeval task 2018. In *14th Conference on Natural Language Processing KONVENS 2018*.

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

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019b. SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media (OffensEval). In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.