Classification of Offensive Language on Social Media

Anmol Gupta 2018329 Naman Jain 2018347 Navneet Agarwal 2018348 Sudhir Attri 2018267

Abstract–In this project, we used several machine learning (ML) and built deep learning (DL) models for the *OLID: Offensive Language Identification Dataset*. The data set was annotated on three sub-tasks: to classify tweets as offensive and not offensive (Sub-task A), targeted and untargeted (Sub-task B) and finally, target identification (Sub0task C). We tried different ML models such as Logistic Regression (LR), Support Vector Classifier (SVC) and Random Forest (RF). We also used CNN, BiLSTM and CNN+BiLSTM as the DL models. For the first sub-task, SVC and CNN+BiLSTM had a performance comparable to LR, but LR achieved the best score on all three sub-tasks.

1 Introduction

With the exponential rise of the internet and social media, and the anonymity it provides, toxic online content has become a major issue. Toxic content is usually divided into two categories – offensive language and hate speech, with the latter being a serious concern for organisations. Traditionally, a human moderator would review the reports, following the policies the site has put in place against such content. However, every minute hundreds of thousands of comments are made on a single platform, which makes this method expensive and inefficient. The need arises to have a system which can flag toxic content automatically. In our project, we aim to harness the power of natural language processing and machine learning techniques to create such an automated system.

2 Related Work

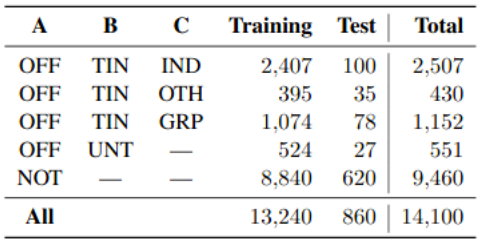
Many studies in this domain have worked on the the problem of toxic speech detection as a classification task, by either performing binary classification to determine whether a comment is toxic, or performing multi-class classification to identify the type of toxic comment. Study [1] uses different models to flag toxic content and distinguish between offensive language and hate speech. On a data set of tweets, it uses n-grams for feature extraction which are then weighted using TF-IDF. It compares the performance of Naive Bayes, Logistic Regression (LR) and Support Vector Machines (SVMs). Study [2] also uses a LR model with similar hyperparameters, but includes more features such as hashtags, mentions, retweets, and URLs, number of characters and syllables in each tweet.

Study [3] implemented a hierarchical structure (classification at three different levels) to identify the type and target of offensive tweets. Based on this approach OLID (Offensive Language Identification Dataset) was created and three models were tested on it: Linear SVM model trained on unigrams, Bidirectional Long Short-Term-Memory (BiLSTM) model with three layers (an input embedding layer, a bidirectional LSTM layer and an average pooling layer of input features) and a Convolutional Neural Network (CNN) based on same multi-channel inputs as the BiLSTM model. For the first two levels, CNN had the best macro-F1 scores and for the third level both BiLSTM and CNN had equal macro-F1 scores.

CNN approach was used in [6] where the feature embedding was generated using word embedding from word2vec and random vectors techniques as well as character n-grams. With random vector word model as the baseline, word2vec performed the best among others except the recall from LR with character n-grams model. The paper suggests the Long Short-Term Memory (LSTM) to utilise the sequential nature of comments. Such a LSTM based model was proposed by [4]–the features for input layers were computed using word-based frequency vectorisation. The ensemble of multiple LSTM classifiers outperformed all single classifiers, and the F_1-scores increased with a user's tendency to write hate-speech.

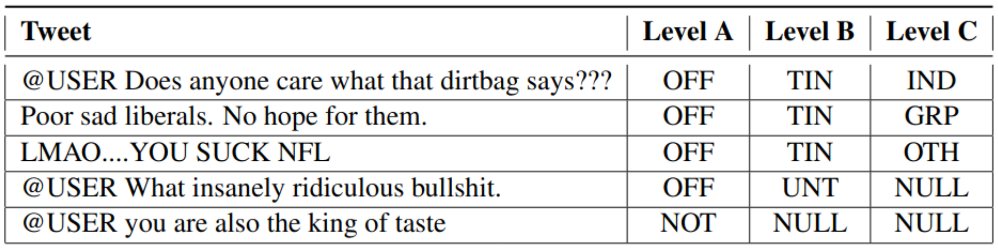
3 Data Set Details

The Offensive Language Identification Dataset (OLID) contains a collection of 14,100 annotated English tweets. It is split into three-level subtasks to identify the type and target of offensive tweets.



A	B	C	Training	Test	Total
OFF	TIN	IND	2,407	100	2,507
OFF	TIN	OTH	395	35	430
OFF	TIN	GRP	1,074	78	1,152
OFF	UNT	—	524	27	551
NOT	—	—	8,840	620	9,460
All			13,240	860	14,100

Fig. 1 Distribution of labels in the data set



Tweet	Level A	Level B	Level C
@USER Does anyone care what that dirtbag says???	OFF	TIN	IND
Poor sad liberals. No hope for them.	OFF	TIN	GRP
LMAO....YOU SUCK NFL	OFF	TIN	OTH
@USER What insanely ridiculous bullshit.	OFF	UNT	NULL
@USER you are also the king of taste	NOT	NULL	NULL

Fig. 2 Some Example Tweets

In *Sub-task A: Offensive Language Identification* the aim is to classify a tweet as offensive or not offensive. In *Sub-task B: Categorization of Offensive Language* the aim is to determine whether an offensive tweet is targeted or untargeted In *Sub-task C: Offensive Language Target Identification* the aim is to determine the target–individual, group or other.

4 Methodology

4.1 Data Preprocessing

Preprocessing of data set text is an important step to clean the data and transform into better usable form for our ML models. Preprocessing was achieved by applying following steps:

- **Usernames and URLs**: These were already anonymized in the dataset itself so that none of the users can be identified.
- **Data cleaning**: Converted every tweet into lower case to maintain uniformity. Punctuation marks, extra symbols and hashtags were removed from sentences.
- **Stopwords removal**: Stop words include common words such as "I", "me", "they" and removing them improves the quality of features.
- **Stemming**: Replaced each word by its corresponding root word to avoid redundant patterns. Porter stemmer was used to perform the word stemming process.

4.2 Features

Feature extraction is one of the most critical factors when performing classifications tasks. Every tweet was represented as a feature vector formed by combining the following features.

- **Word n-grams**: Existence of sequences of length 1, 2, 3 and 4 was obtained from the data. TF-IDF transformation was also performed to give weight to the presence of words that may appear frequently in a few tweets, but not the entire corpus.
- **Hashtags**: Hashtags are often used on Twitter to target or glorify certain individuals or groups, attracting a barrage of offensive comments. We used the count of hashtags.
- **Emoticons**: Unicode emoticons are very common on social media today. We used a dictionary of emojis and obtained the existence of certain sentiment depicting emojis.
- **Upper-case word count**: Upper-case words are usually found in harsh tweets. The number of such words was used a feature.
- **Continuous punctuation**: The number of occurrences of multiple punctuation, e.g. "???", "!!!!". These are commonly used in conjunction with upper-case words to depict strong emotions.
- **VADER sentiment**: It is a technique made for sentiment analysis of posts on social media. It calculates the positive, negative and compound score for every tweet. We incorporated the compound score for each tweet as a feature.

- **Flesch reading-ease score**: High score indicate material that is easier to read; lower numbers mark passages that are more difficult to read. If the text score is high then it is very likely to be engaging for a larger audience. This will help in identifying the tweets which use slangs or sarcasm.
- **Linguistic features**: Following are the lexicon corpora and other resources we used for features engineering. Emotions associated with text often capture the information for the intended intuition behind it.
    - ◇ NRC-10 Expanded Score: Compound scores of aggregate of matching words score for each emotion were extracted for each tweet.
    - ◇ NRC Word-Emotion Score: Counts of the number of words matching each emotion in the lexicon were extracted for each tweet.
    - ◇ SlangSD Word Count: Identifying slang sentiment words can be an advantage to accurately discovering sentiment hidden in tweets and hence number of slang words in each tweet was extracted.
- **Word embedding**: word2vec model provided by gensim's python library was used, it created word embedding trained on our database as corpus using Continuous Bag of Words (CBOW) model.

4.3 Models Used

We used three ML classifiers.

Logistic Regression: It predicts the probability of events using a continuous function. The final output is computed by applying the sigmoid function to the linear regression.

Support Vector Classifier: This model finds the best hyperplane in n-dimensional space, which can be used to classify the points into distinct classes. We used a linear kernel for this model and used balanced class weights to avoid the class imbalance problem.

Random Forest: This is an ensemble method whose base estimator is Decision Trees. It uses bootstrapping mechanism to sub-select some of the features and train each decision trees on those sub-selected features, after this the model assigns the label using majority voting technique.

We further tried three DL models, CNN, BiLSTM and CNN+BiLSTM. Keras with a TensorFlow back-end was used for implementation of these models.
Their outputs were combined with non-sequential features (sentence-level features) and the output was generated using sigmoid and softmax activations for binary or multiple classifications. The optimization was done by cross-entropy loss of the prediction. We used the following architectures.

- **Convolutional Neural Network**: Convolutional filters of size 2 to 5 over one dimension, i.e. along the sequence of words, followed by batch normalization and max.

pooling along the same dimension and same padding. The same padding allowed the outputs of all filters to be merged together.

- **Long Short-Term Memory**: 2 LSTM layers used one after the other, second one in reverse direction to capture both forward and backward relationships.

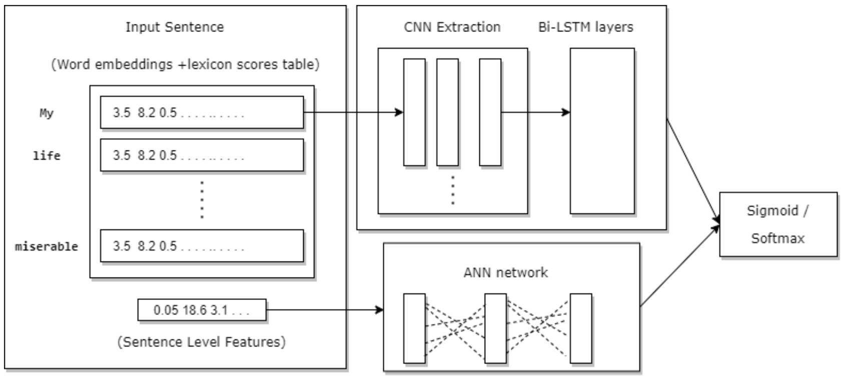


Fig. 3 Model Architecture

5 Novelty

- Most of the related work had only considered emoticons formed using punctuation, letters and numbers in their features, and did not include emojis–the pictographs of faces and symbols. Due to the increase in the usage of mobile devices, emojis are used much more than their keyboard-based predecessor and carry a lot of meaning. We used a python library to construct an emoji vocabulary which was used to check the presence of certain sentiment depicting emojis in a tweet.

- Informal language like slang words is excessively on social media today. They also contain a lot of meaning a tweet which can help in classifying the tweet. We used a slang dictionary to count the number of slang words in a tweet.

- For the neural network models, we distanced words embedding vectors using classes defined by word level lexicons which would help CNN in understanding structure of sentences. This was achieved by combining the word embedding vector with lexicon features and keeping different classes distant from each other on these extra axes, this had a positive impact on performance of the order of 4% F1.

- Our model also incorporated lexicon scores for sentences in the form of sentence-level features being used in the final classification layer along with the CNN-BiLSTM output for final prediction.

- Experimented variant of CNN+BiLSTM models by merging output of 1D convolutions over word embedding for capturing n-grams (1-4) relationships between words, using a variant of CNN inspired from [7]

6 Results and Analysis

We first used the classifiers for sub-task A, the tweets that were classified as offensive were used for sub-task B, and the tweets classified as targeted were finally used for sub-task C to determine the target.

We tried and compared the performances of different ML and DL classifiers. For ML models, all the features mentioned in section 4.2 were used for this model except word embedding. In Table 1, it can be seen that LR outperformed every other model using F1 score as metric. We managed to achieve a macro F1-score better than both BiLSTM and SVM from the official SemEval-2019 Task 6 results [8] for sub-task A, and also CNN for sub-task B and C. However, the BERT models used by the top teams in [8] achieved an even higher score of 0.82, 0.75 and 0.66 in sub-tasks A, B and C respectively.

From our model analysis, we observed that the ML classifiers relies too much on the presence of individual keywords, ignoring the context of these keywords which calls for the need of a more sequential level context capturing. Linguistic features along with word embedding when incorporated in DL architectures contributed to that context.

Sub-task	**Model**	**Macro F1-score**
A	LR	0.76
A	SVC	0.74
A	RF	0.71
A	CNN+BiLSTM	0.75
B	LR	0.69
B	SVC	0.65
B	RF	0.49
B	CNN+BiLSTM	0.63
C	LR	0.63
C	SVC	0.62
C	RF	0.50

Table 1: Results for the sub-tasks

7 Individual Contribution

All the members contributed equally to this project. The literature review, analysis of features to be used and results was done together.

- **Anmol** and **Navneet**: data preprocessing, feature extraction for ML models and tried ML models (LR, SVC and RF).
- **Naman** and **Sudhir**: Extracted the features to be used by DL models, implemented the models (CNN + BiLSTM), created a user interface for demo.

8 References

[1] A. Gaydhani, V. Doma, S. Kendre, and L. Bhagwat, 'Detecting Hate Speech and Offensive Language on Twitter using Machine Learning: An N-gram and TFIDF based Approach', arXiv:1809.08651 [cs], Sep. 2018, Accessed: Oct. 31, 2020. [Online]. Available: http://arxiv.org/abs/1809.08651.

[2] T. Davidson, D. Warmsley, M. Macy, and I. Weber, 'Automated Hate Speech Detection and the Problem of Offensive Language', arXiv:1703.04009 [cs], Mar. 2017, Accessed: Oct. 31, 2020. [Online]. Available: http://arxiv.org/abs/1703.04009.

[3] M. Zampieri, S. Malmasi, P. Nakov, S. Rosenthal, N. Farra, and R. Kumar, 'Predicting the Type and Target of Offensive Posts in Social Media', 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), Minneapolis, Minnesota, Jun. 2019, pp. 1415–1420, doi:10.18653/v1/N19-1144 .

[4] G. K. Pitsilis, H. Ramampiaro, and H. Langseth, 'Effective hate-speech detection in Twitter data using recurrent neural networks', Appl Intell, vol. 48, no. 12, pp. 4730–4742, Dec. 2018, doi: 10.1007/s10489-018-1242-y.

[5] C. Nobata, J. Tetreault, A. Thomas, Y. Mehdad, and Y. Chang, 'Abusive Language Detection in Online User Content', in Proceedings of the 25th International Conference on World Wide Web - WWW '16, Montreal, Quebec, Canada, 2016, pp. 145–153, doi: 10.1145/2872427.2883062.

[6] B. Gambäck and U. K. Sikdar, 'Using Convolutional Neural Networks to Classify Hate-Speech', in Proceedings of the First Workshop on Abusive Language Online, Vancouver, BC, Canada, Aug. 2017, pp. 85–90, doi: 10.18653/v1/W17-3013.

[7] Lopez, M.M. and Kalita, J., 2017. Deep Learning applied to NLP. arXiv preprint arXiv:1703.03091. Available: https://arxiv.org/pdf/1703.03091.pdf

[8] Zampieri, M., Malmasi, S., Nakov, P., Rosenthal, S., Farra, N. and Kumar, R., 2019. SemEval-2019 Task 6:Identifying and Categorizing Offensive Language in Social Media(OffensEval). Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019), pp.75–86. https://www.aclweb.org/anthology/S19-2010.pdf.