Zeyad at SemEval-2019 Task 6: That's Offensive! An All-Out Search For An Ensemble To Identify And Categorize Offense in Tweets

Zeyad El-Zanaty
Faculty of Engineering
Alexandria University
zeyadzanaty@gmail.com

Abstract

The objective of this paper is to provide a description for a classification system built for SemEval-2019 Task 6: OffensEval. This system classifies a tweet as either offensive or not offensive (Sub-task A) and further classifies offensive tweets into categories (Sub-tasks B - C). The system consists of two phases; a brute-force grid search to find the best learners amongst a given set and an ensemble of a subset of these best learners. The system achieved an F1-score of 0.728, ranking in subtask A, an F1-score score of 0.616 in subtask B and an F1-score of 0.509 in subtask C.

1 Introduction

In OffensEval we break down offensive content into three sub-tasks taking the type and target of offenses into account. Sub-task A - Offensive language identification; In this sub-task we are interested in the identification of offensive posts and posts containing any form of (untargeted) profanity. In this sub-task there are 2 categories in which the tweet could be classified Not Offensive - This post does not contain offense or profanity. Non-offensive posts do not include any form of offense or profanity. Sub-task B - Automatic categorization of offense types; In this sub-task we are interested in categorizing offenses. Tweets are labeled from one of the following categories Targeted Insult - A post containing an insult or a threat to an individual, group, or others; Untargeted - A post containing non-targeted profanity and swearing. Posts containing general profanity are not targeted but they contain non-acceptable language. On the other hand, insults and threats are targeted at an individual or group. Sub-task C - Offense target identification. Finally, in sub-task C we are interested in the target of offenses. Only posts which are either insults or threats are included in this sub-task. The three categories included in sub-task C are the following: Individual - The target of the offensive post is an individual: a famous person, named individual or an unnamed person interacting in the conversation. Group - The target of the offensive post is a group of people considered as a unity due to the same ethnicity, gender or sexual orientation, political affiliation, religious belief, or something else. Other The target of the offensive post does not belong to any of the previous two categories (e.g. an organization, a situation, an event, or an issue).(Zampieri et al., 2019b)

To work with such complicated tasks our approach is an exhaustive one. We try combinations of many techniques in pre-processing, feature extraction and classification while tuning their hyper-parameters to find the best models with the leading F1-scores. With the gained information an ensemble of the top three models is formed to get the optimum result. Along side to this approach we try a deep-learning method with a simple 1D - CNN consisting of 3 convolutional layers and a softmax layer just to compare results.

2 Related Work

Offensive language on social media hardly remains unnoticed. Contents involving hateful messages vary from hate speech to group-based racism and could target anyone irrespective of their status, identity, location and so forth. Even when it is not materialized into a hate-motivated crime, the damage is done victims are being labeled, marginalised and exposed to negative stereotyping. The overall consequences of online hate can be the dehumanisation of individuals or groups of individuals. The need for proper strategies to tackle hate speech on social media is unquestionable. The core focus of the thesis is not to find a solution to the challenge, but rather to identify central problems that have contributed to the formation of the existing reality. To unrave the contributing factors, a holistic analysis of both international human rights principles regarding hate speech and the practical application of those standards is necessary.(Schofield and Davidson, 2017)

There have been many studies and publication on the topic of offensive language and hate speech over the last few years. Examples on such studies include (Davidson et al., 2017), (Malmasi and Zampieri, 2017), (ElSherief et al., 2018), (Gambäck and Sikdar, 2017), (Zhang et al., 2018). Also there have been challenges on how to distinguish profanity from hate-speech presented by (Malmasi and Zampieri, 2018).

3 Methodology and Data

The used dataset in this assignment is the one provided in SemEval-2019 task 6. The dataset has been collected from Twitter. It was retrieved by searching offensive terms that could be present in a tweet. It consists of 14,100 tweets in total. It was annotated using crowdsourcing. The gold labels were assigned taking the agreement of three annotators into consideration. No correction has been carried out on the crowdsourcing annotations. The dataset was presented in two phases; Training data: already labeled tweets used to train the classifiers. Each tweet was provided with a binary classification label and an index. Testing data: unlabeled tweets to test the classifiers against. Zampieri et al. (2019a).

The system is a combination of three essential layers. First, pre-processing which is a necessary step in NLP as textual data could and most likely is not clean, thus will affect further stages and create an incoherent model. Second, feature extraction or vectorization, which translates words to a number or a series of numbers with different weights to represent this word. Finally, classification, features extracted from the previous step is fed into a learner and a model is created that could classify tweets.

For our approach we implemented a heap of pre-processors, vectorizers and classifiers and with the help of brute-froce search ranked all the resulting models according to their F1-scores. All implemented techniques are available in Table 1. For the implementation see: `github.com/zeyadzanaty/offenseval`

Phase	Implemented Techniques
Pre-processing	Stopwords Removal Lemmatization - Stemming
Feature Extraction	TFIDF - Count - Word Embedding
Classification	KNN - Naive Bayes - Decision Trees - SVM -Random Forest Logistic Regression - MLP

Table 1: Multiple techniques implemented in our system.

3.1 Pre-processing

A tweet contains many unwanted data that would take extra computational power and decrease the accuracy of the model. So, noise removal and some normalization techniques must be applied to the corpus in-order to generate more consistent models. **Stopword Removal** is a noise removal method by filtering words that dont have significance in the context of the sentence, without them the semantics of the tweet wont be affected. **Lemmatization** is the process of getting the linguistic root of a word. First, words are part-of-speech tagged , then converted to their roots. **Stemming** is the process of stripping a word of it's prefixes and suffixes using the porter-stemmer algorithm (Porter, 1980).

For this step, a list of all combinations of pre-processing techniques is used. For example it would look something like: [(Stopwords Removal), (Stowords Removal, Lemmatization), (Stopwords Removal, Stemming), (Lemmatization), etc..]

3.2 Feature Extraction

Now that we've got our clean, almost noise-free textual data, we cant simply feed a classification model a bunch of text words, most models only work with numerical data. This is where we convert words to numerical features using

one the methods mentioned below to create our classification-ready data.
We use three word embedding models of embedding dimension 100 (which gave adequate results after experimenting with other dimensions) along side to the standard **TFIDF/Count** models.

- Word2Vec model trained on our dataset.(Mikolov et al., 2013)
- fastText model trained on our dataset.(Joulin et al., 2016)
- Pre-trained GloVe model trained on 2 Billion tweets - 27 Billion tokens - 1.2 million vocabulary.(Pennington et al., 2014)

All three models mentioned are zero-padded with the maximum length of a tweet present in the dataset to resolve the uneven dimensionality issue. A list of all techniques is initialized for later usage in the search for the best model.

3.3 Classification and Tuning

This is where all the previous work comes together for the final phase of the system. Seven models where chosen and tuned using sci-kit learns (Pedregosa et al., 2011) GridSearchCV, which does a cross validation search on a list of hyper-parameters for a given model. The parameters grids that were tested are available in Table 2.

Model	**Parameters Grid**
KNN	n_neighbours: [1, 3, 5, 7]
Naive Bayes	fit_prior: [True, False]
SVM	C: [0.1,10,100] kernel: [rbf, poly]
Decision Trees	criterion: [gini, entropy]
Random Forest	n_estimators: [10 - 200]
Logistic Regression	penalty : [l2] solver: [sag, lbfgs, newton]
MLP	activation:[tanh, relu] solver: [sgd,adam, lbfgs]

Table 2: Classification models and their corresponding parameters to tune.

Again, a list of classifiers and their parameters grids is initialized to tune them with a 3-fold cross validation.

3.4 All-Out Search

This is the body of all the work. We try every possible combination of pre-processing, vectorization and classification to ensure the output has the best possible F1-score for the given subtask. We start by cleaning the data using a certain combination of pre-processors, then extracting features using one of the vectorizers and finally to complete the pipeline, tune a classifier's hyper-parameters on the resulting data-matrix. And repeat for the next combination.

```
1: procedure SEARCH($preprocessors$, $vectorizers, classifiers$)
2:     $models \leftarrow \{\}$
3:     for $prp \in preprocessors$ do
4:         clean-data(prp)
5:         for $vec \in vectorizers$ do
6:             vectorize-data(vec)
7:             for $clf \in classifiers$ do
8:                 $models[clf] \leftarrow tune(clf)$
9:     $sort(models)$
```

The resulting set 'models' is a set of each classifier and a list of parameters and their corresponding preprocessors, vectroizers and F1-scores. The results could be plotted to help visualize the performance of each model seen in Figure 1, which

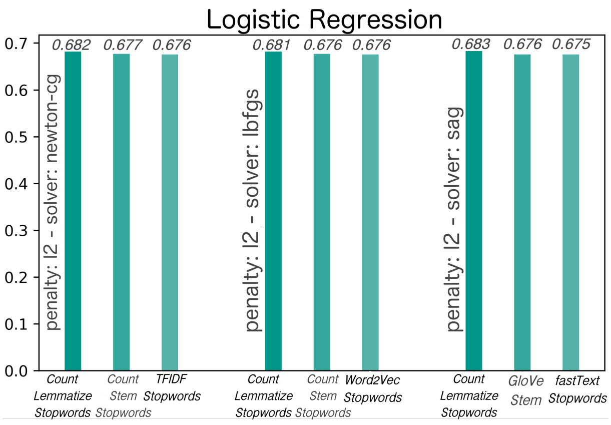

Figure 1: F1-scores of logistic regression model on subtask A, each bar is a model with it's own pre-processing, vectorizer and parameters.

shows the top 3 models for the logistic regression classifier. Each 3 bars represent the hyper-parameters combination and the top 3 combinations of pre-processing and vectorization. The best F1-score (0.683) came from a pre-processing of stopwords removal followed by lemmatization, a count vectroizer and hyper-parameters [penalty: l2, solver: sag]. Following this search, now that

Subtask	Phase	1st	2nd	3rd
A	Pre-processing	Stopwords Removal & Lemmatization	Stopwords Removal & Stemming	Lemmatization
	Vectorization	Count	Count	Count
	Classification	Logistic Regression	Naive Bayes	Random Forest
B	Pre-processing	Lemmatization	Lemmatization	Stopwords-Removal
	Vectorization	TFIDF	TFIDF	GloVe-Embeddings
	Classifiaction	Naive Bayes	Random Forest	MLP
C	Pre-processing	Lemmatization	Stopwords Removal	Stemming
	Vectorization	Count	Count	Count
	Classification	Random Forest	Logistic Regression	Naive Bayes

Table 3: Top 3 models for each subtask, these 3 models will form an ensemble to enhance the performance.

we have the scores of each model, we can model an ensemble of the top 3 models to give us a better overview of the data available in Table 3. And just to add an extra layer we can re-tune the classifier parameters in case of any error that could have appeared in the previous step.

4 Results

We submitted with a couple of models, for subtask A, an ensemble of the three top models mentioned in Table 3, the Random Forest model and a 1-D CNN. The ensemble did its best in subtask A but the Random Forest (RF) model came a very close second. Results can be viewed in Table 4, and confusion matrix Figure 2.

System	F1 (macro)	Accuracy
All NOT baseline	0.4189	0.7209
All OFF baseline	0.2182	0.2790
Ensemble	**0.7289**	**0.8151**
Random Forest	0.7143	0.8128
1D-CNN	0.5506	0.6977

Table 4: Results for Sub-task A. The ensemble approach gave the best results.

As for subtask B, the ensemble submission unfortunately failed, but it didn't look good anyway. The best model was as simple Naive Bayes (NB)-TFIDF model which got a very good F1-score of 0.887. Results can be viewed in Table 5, and confusion matrix Figure 3.

System	F1 (macro)	Accuracy
All TIN baseline	0.4702	0.8875
All UNT baseline	0.1011	0.1125
1D-CNN	0.4436	0.5542
Naive Bayes	**0.6161**	**0.8542**

Table 5: Results for Sub-task B. The best performer was a simple TFIDF - Naive Bayes model.

Finally, for subtask C, we chose to let go of the CNN model as it didn't get an acceptable result, and went for the ensemble, which got the best accuracy but came second for F1-scores and the RF model which also got a good accuracy but a poor F1-score, and the best model was a logistic regression-count model with an F1-score of 0.5093. Results can be viewed in Table 6, and confusion matrix Figure 4.

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
Ensemble	0.4973	0.6479
Random Forest	0.4763	0.6432
Logistic Regression	**0.5093**	**0.6056**

Table 6: Results for Sub-task C. The ensemble got the best accuracy but, LR got a better F1-score.

Looking at these results, we hypothesize that the systems performance can be improved by combining all word embedding features instead of using them individually. It was also remarkable that the for most subtasks a simple Naive Bayes - TFIDF model came close to being the best amongst all others. We also believe better results can be achieved if there was the dataset was more balanced and having more offensive tweets, and if we had sufficient time to perform grammar checking on the tokens and other operations that can reduce noise. The problem of out-of-vocabulary (OOV) words which we unfortunately didn't attempt to solve, could be later be solved by using a character-level embedding model rather than a word embedding one.

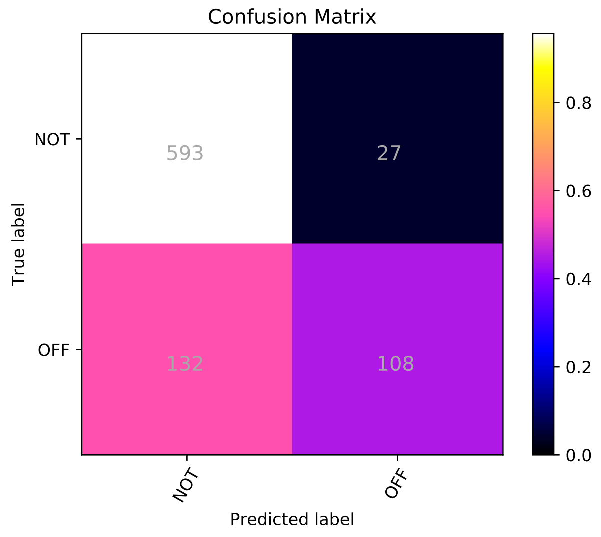

Figure 2: Sub-task A, Ensemble of LR-NB-RF

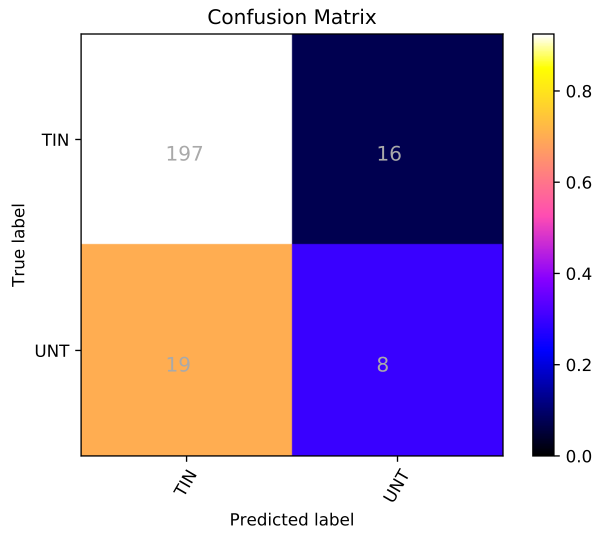

Figure 3: Sub-task B, Naive Bayes - TFIDF - Lemmatization

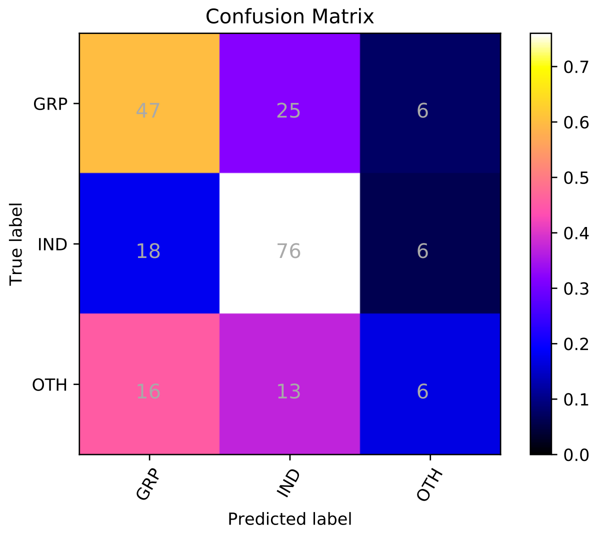

Figure 4: Sub-task C, Logistic Regression - Count - Stopwords Removal

5 Conclusion

This paper describes our offensive tweets identification and categorization system that was built in the framework of SemEval-2019 Task 6. We used a brute-force search technique to find the best model that could be generated from a list of prepocessing techniques, feature extraction models and classifiers and got an F1-sore of 0.728 in subtask A, 0.6161 in subtask B and 0.5093 in subtask C. In future work, we aim to focus more on word embedding features by concatenating all 3 word vector models and experiment with character-level/sentence-level models.

References

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

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*.

Björn Gambäck and Utpal Kumar Sikdar. 2017. Using Convolutional Neural Networks to Classify Hate-speech. In *Proceedings of the First Workshop on Abusive Language Online*, pages 85–90.

Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hérve Jégou, and Tomas Mikolov. 2016. Fasttext.zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651*.

Shervin Malmasi and Marcos Zampieri. 2017. Detecting Hate Speech in Social Media. In *Proceedings of the International Conference Recent Advances in Natural Language Processing (RANLP)*, pages 467–472.

Shervin Malmasi and Marcos Zampieri. 2018. Challenges in Discriminating Profanity from Hate Speech. *Journal of Experimental & Theoretical Artificial Intelligence*, 30:1–16.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. 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 26*, pages 3111–3119. Curran Associates, Inc.

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 *In EMNLP*.

Martin F Porter. 1980. An algorithm for suffix stripping. *Program*, 14(3):130–137.

Alexandra Schofield and Thomas Davidson. 2017. Identifying Hate Speech in Social Media. *XRDS: Crossroads, The ACM Magazine for Students*, 24(2):56–59.

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)*.

Ziqi Zhang, David Robinson, and Jonathan Tepper. 2018. Detecting Hate Speech on Twitter Using a Convolution-GRU Based Deep Neural Network. In *Lecture Notes in Computer Science*. Springer Verlag.