International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Improve Offensive Language Detection with Ensemble Classifiers

Ekin Ekinci*[1], Sevinc Ilhan Omurca[2], Semih Sevim[3]

Submitted: 14/03/2020 Accepted: 22/04/2020

Abstract: Sharing content easily on social media has become an important communication choice in the world we live. However, in addition to the conveniences it provides, some problems have been emerged because content sharing is not bounded by predefined rules. Consequently, offensive language has become a big problem for both social media and its users. In this article, it is aimed to detect offensive language in short text messages on Twitter. Since short texts do not contain sufficient statistical information, they have some drawbacks. To cope with these drawbacks of the short texts, semantic word expansion based on concept and word-embedding vectors are proposed. Then for classification task, decision tree and decision tree based ensemble classifiers such as Adaptive Boosting, Bootstrap Aggregating, Random Forest, Extremely Randomized Decision Tree and Extreme Gradient Boosting algorithms are used. Also the imbalanced dataset problem is solved by oversampling. Experiments on datasets have shown that the extremely randomized trees which takes word-embedding vectors as input are the most successful with an F-score of 85.66%.

Keywords: *BabelNet, ensemble classifiers, offensive language, short text classification, Twitter, Word2Vec*

1. Introduction

As a result of social media being a part of our daily lives, the way people interact with each other has been radically reshaped. Social media has become an important tool to connect the people all over the world. This is because it allows its users to share the content they want quickly, efficiently and in real-time. However, user-created content shared on social media is not always organized by the rules. In fact, nowadays, content written in an offensive language has become widespread on social media. Offensive language is defined as message which contains insulting or threatening expressions written by one person to another. The severity of this problem is increasing each day; consequently, it is very important to deal with this problem in terms of government policy, social media terms and policies and online community plans. At this stage, there is a need for effective methods.

Social media generates a large amount of data daily as mentioned above. Therefore, it is very difficult to manually determine offensive language on the social media even by an expert. Such a "@BreitbartNews Good" language on Twitter, we can easily say that this language is not offensive. However, we cannot say that every content is clear and there are millions of content waiting to be analysed. Classification techniques in machine learning are quite successful in analysing such languages in social media [1].

In the last few years, several studies engaged with prediction of offensive languages in social media especially in Twitter has been realized [2-5]. Although this type of environments includes many raw data from user posts, which is useful for text classification tasks, it brings about some disadvantages. Traditionally for classification task, long texts are used in the literature [6, 7]. In social media texts, the situation is the opposite.

In social media, users write short text messages of 200 characters or less. This type of texts, usually written with the word of mouth behaviour, contains abbreviations, emoticons, misspellings, symbols, slangs and so on, that make classification more difficult [8]. Thus, we face with shortness, sparsity, and misspelling and informal writing as three main drawbacks of short text messages [9, 10]. Short texts contain a small number of words for representation. There are not enough features for statistical modelling and not contain sufficient statistical information [11]. Therefore, the features are very sparse and this leads to low accuracy of the statistical model. Misspelling and informal writing: short texts commonly contain misspellings and noises. These drawbacks pose challenges to achieve superior classification accuracy. Therefore, the short text classification problem is still an important field of study for researchers.

Liu et al. [12] applied part of speech (pos) tagging for feature selection and used HowNet lexicon to improve classification of microblogs in Chinese. Jiang et al. [13] classified sentiments of tweets with Support Vector Machines (SVM) by incorporating target based syntactic features and context of tweets. For short text classification in English and Korean, Kim et al. [14] proposed language independent semantic (LIS) kernels. These kernels can capture the similarity between texts successfully and do not need grammatical tags and lexical databases. Wang et al. [15] devised semantic clustering and convolutional neural network (CNN) based short text classification model. In their model, semantic clustering was realized with fast clustering algorithm and obtained semantic units from word-embeddings were given as input to the CNN. In another work, Wang et al. [16] classified short texts by using word-embedding clustering and CNN. Sotthisopha and Vateekul [17] used fast semantic expansion on multichannel CNN. Semantic expansion was obtained by clustering word-embeddings with mini batch K-Means++ algorithm. When these studies and more are examined

[1]*Dogus University, Faculty of Engineering / Software Engineering ORCID ID: 0000-0003-0658-592X*
[2]*Kocaeli University, Faculty of Engineering, Computer Engineering ORCID ID: 0000-0003-1214-9235*
[3]*Kocaeli University, Faculty of Engineering, Computer Engineering ORCID ID: 0000-0002-2486-7704*
** Corresponding Author Email: ekinekinci.61@gmail.com*

in detailed, it is observed that for short text classification task, single classifiers achieve successful results. However, improving these results is always a research problem.

Based on the assumption that "Using a single classifier is not enough to represent whole problem space." ensemble classifiers, combine the output of individual base classifiers (SVM, Decision Tree, and so on) [18, 19]. The success of the method depends on both the ability to combine accurate predictions and the ability to reduce errors from the base classifiers. Because all of these, ensemble classifiers have been frequently used for short text classification task [8, 20-23].

In this study, for classifying offensive language in short texts, we compare semantic word expansion methods by using decision tree and decision tree based ensemble classifiers. For the experiments, the Offensive Language Identification Dataset (published for SemEval-2019 Task 6), which is about Identifying and Categorizing Offensive Language in Social Media (OffensEval) is used. We concentrate on sub-task A; it is about discrimination between offensive (OFF) and non-offensive (NOT) languages. The highest achievement for this task is the work of Liu et al. [24] with F-score of 0.8286.

In ensemble classification, creating diversity among classifiers is very critical for accuracy. Since offensive language classification is a challenging task, to create diversity two semantic enhancement way is provided. In the step of expansion, word-embedding vectors and BabelNet concepts are used. BabelNet concepts are used for the first time for this task in the literature. Also the dataset is imbalanced and classification models cope with the problem of between class imbalance frequently. In this study, this problem is also addressed and over-sampling is applied to the dataset. Classification models are compared over raw, concept based and word-embedding based expanded datasets. The experimental results show that while similar accuracy has been observed for raw and concept expanded datasets, word-embedding based ensemble models outperform existing classification methods.

The rest of paper is organized as follows. In section 2, literature about classification of offensive language in short texts is summarized. In Section 3, pre-processing steps, semantic expansion methods and text representation and oversampling are explained. In Section 4, ensemble classification algorithms are mentioned. In Section 5, conduction of experiments and experimental results are given in detailed. Finally, discussions and conclusions for the future work are summarized in Section 6.

2. Literature Review

In this section we present a brief summary of the previous works in the context of offensive language classification on official dataset (sub-task A) of the shared task SemEval 2019 Task 6: Identifying and Categorizing Offensive Language in Social Media [3].

Zampieri et al. [3] utilized SVM, bidirectional Long Short-Term-Memory (BiLSTM) and CNN to realize sub-task A. CNN achieved best results with F-score of 0.80. Mitrovic et al. [5] proposed C-BiGRU which is combination of CNN with a bidirectional Recurrent Neural Network (RNN). In their model, Word2Vec was used for capturing similarity between words. C-BiGRU was capable of classifying tweets based on long-term dependencies. With this model they got F-score of 79.40%. Kebriaei et al. [25] experienced traditional machine learning methods, deep learning methods, combination of them and an augmentation method for sub-task A. They used different features such as content-based, sentiment-based, TF-IDF and hate-based to improve classification performance. They achieved best results using augmentation method with F-score of 0.76. Aggarwal et al. [26] represented posting space with word-embeddings and applied Multi-Layer Perceptron (MLP) and BERT for classification. Best results were obtained by using BERT with F-score of 0.798. Rani and Ojha [27] trained SVM with unigram, bigram, trigram and 4-gram and provided F-score of 78.58%. Patraş et al. [28] applied rule based approaches to determine whether the language was offensive or not and achieved a maximum F-score of 0.6446. Kapil et al. [29] used CNN-BiLSTM-Attention and achieved an F-score of 0.7594. Bansel et al. [30] tried LSTM with pre-processing, LSTM with pre-processing and lexicon and LSTM with hashtag parsing methods for sub-task A and obtained F-score of 0.7327 with LSTM+ hashtag parsing. Balasubramanian et al. [31] carried out 2D-CNN with Word2Vec Learned Embeddings and 1D-CNN with GloVe. 2D-CNN had better F-score of 0.7382. Thenmozhi et al. [32] applied LSTM with Normed Bahdanau and Scaled Luong attentions. They got best F-score of 0.5341 with Scaled Luong. Sarracen and Rosso [33] utilized BiLSTM-RNN, CNN and ensemble of these models and these models took word-embeddings as input. When the classification results were compared, it was seen that CNN achieved better F-score of 0.66. Han et al. [34] proposed RNN with Gated Recurrent Unit (GRU) and modified sentence offensiveness calculation (MSOC) for classifying offensive language. The best F-score of 0.6899 was obtained with RNN. Wang et al. [35] built K-max pooling CNN with meta embedding and global learning rate (CLR) to improve classification performance. When the results were examined, it was seen that proposed method achieved F-score of 0.8024. Liu et al. [24] achieved the most successful classification results for this task with F-score of 0.8286. They trained BERT with only 3 epochs and classified offensive language as offensive or not offensive. Indurthi et al. [36] applied three different sentence-embedding methods to discriminate offensive and not offensive content. Among these methods, Deep Contextualized Word Representations (ELMo) provided F-score of 0.6436. Oberstrass et al. [37] also used ELMo with LSTM and obtained F-score of 0.767. Swamy et al. [38] ensembled vote of L1-regularised Logistic Regression, L2-regularised Logistic Regression, Linear Support Vector Classification (SVC), Stochastic Gradient Descent (SGD), and Passive Aggressive (PA) classifier and ensemble model achieved F-score of 0.7434.

3. Feature Engineering

3.1. Pre-processing

Twitter data has structural anomalies involving shortness of tweets, abbreviations, special characters, symbols and emoticons, character repetitions, capitalized words, white spaces and typos. These anomalies complicate obtaining useful information from data. Therefore, we need more preprocessing steps to clean raw data.

First, to translate tweets to standard English text we use the dictionary which was prepared by The University of Texas Computer Science Department. This dictionary comprises of abbreviations, typos-that are often used on Twitter-and their expanded and correct forms. In addition, we remove urls, emoticons, numbers, punctuations and Twitter notations like hashtags (#), retweets (RT) and user mentions that are accepted as noise from texts. After we convert all dataset to lowercase, we realize stemming process by using Stanford CoreNLP framework

[39]. We take an original tweet from dataset like the following: “@USER Liberals donâ€™t give a sh!t. They have no souls.”, it is converted to “liberal do give shit they have no soul” after pre-processing. Another tweet is look like this: “@USER @USER Go home youâ€™re drunk!!! @USER #MAGA #Trump2020 ğŸ‘ŠğŸ‡ºğŸ‡¸ğŸ‘Š” and after pre-processing it takes the following form: “go home you be drunk maya trump by”.

3.2. Concept Generation

Word sense ambiguity (WSA) is a major challenge need to be solved for Natural Language Processing (NLP) tasks. For a computer system, there is no difference between banks in the following sentences: “His bank account is rarely over one million.” and “He is sitting on the river bank across the forest”. However, while in the first sentence bank is defined as financial intuition and money, in the second sentence bank is defined as beach and coast. These words which are used to define the real sense, are called as concept. Concept is the smallest semantic unit with a unique meaning that defines the real sense of a word. So concept extraction is required for disambiguation.
BabelNet, which uses WordNet, Wikipedia, OmegaWiki, Open Multilingual WordNet and Wiktionary infrastructures for extracting million concepts and named entities for 50 languages, is a multilingual semantic network [40]. BabelNet provides an important semantic network for its users for disambiguation.
In this study, to handle the real sense of the words for accurate classification and short text problem effectively we expand each tweet with its concepts by using BabelNet. For example, 5concepts of “Trump” is “current President of the United States”. We change tweet “Trump” into “Trump current President of the United States” hence the tweet is expanded.

3.3. Word-Embedding Vector Generation

It is very vital for NLP tasks to learn the Word2Vec which is a dense, low-dimensional and real-valued vector for a word [41]. Apart from this, the basic assumption behind Word2Vec is words that appear in similar contexts tend to be semantically similar [42]. Overall, word vector is presented as a solution to WSA problem as well as BabelNet.
In our study, we model every tweet with its context vector by using skip-gram because of its ability to achieve powerful training and give accurate results for large datasets. Skip-gram predicts the neighbor words of input word within a certain window size. On the other hand, to create vector representation of tweets we utilize pre-trained word-embeddings, which is trained on Google News dataset. For each word in tweets, the window size is set to five, considering the number of features will be too large. For example, 5-dimensional word vector of “Trump” is “Donald, Baker, Rockefeller, Rogers, Larry”. We change tweet “Trump” into “Trump Donald Baker Rockefeller Rogers Larry” hence the tweet is expanded.

3.4. Term Weighting

Term Frequency-Inverse Document Frequency (TF-IDF) is one of the most widely used weighting schemes in the vector space model. It is the most preferred model in the literature because of its simplicity and powerful structure. TF-IDF weights each term in the corpus based on its inverse document frequency. This means that the more documents a term occurs in, the less likely it is discriminative, and less weight is assigned to this term. Formula for TF-IDF is depicted as in (1) below.

$$d_{i,j} = tf_{i,j} \times \log \frac{N}{n_i} \quad (1)$$

The left side of the equation above is weight of the term i in document j. While the first term in the right side of the equation expresses the frequency of term i in document j, second term expresses the idf value of term i. In the second term, N in the nominator is number of documents and n_i in the denominator is the number of documents term i occurs.

3.5. Oversampling

In classification tasks, quality of the training data is always an important factor for improving classification accuracy. However, class imbalance distribution is a common problem and an emerging issue in many real world data [43]. Class imbalance occurs when the most of the data belongs to the majority class, while a small part of it belongs to the minority class. This means, if the whole training data is used for classification task, the task is results with misclassification of unseen data in terms of minority class. In such a case, reducing the imbalance in training data can be a solution.
Oversampling reduces the imbalance in training data by duplicating or creating samples from minority class [43]. With synthetic minority oversampling technique (SMOTE), number of samples are increased by using one of the k nearest instances of sampled instance and random interpolation of both of them [44].

4. Ensemble Construction Techniques

In recent years, multiple classifier systems also called ensemble classifiers have become increasingly important for machine learning and computational intelligence community due to their robustness and effectiveness in real-world applications. The underlying idea behind these classifiers is to create a stronger classifier from multiple weak classifiers [45].
Ensemble creation is generally performed in four steps as given below [46].

- A labeled dataset is given as input to ensemble classifiers.
- Base classifier is used to learn relation between input features and output feature and form a model from relation.
- Diverse classifiers are generated.
- Finally results of diverse classifiers are combined.

The most common approaches in ensemble classifiers are Adaptive Boosting (AdaBoost), Bootstrap Aggregating (Bagging), Random Forest, Extremely Randomized Decision Tree (Extra-trees) and Extreme Gradient Boosting (XGboost) which are implemented as decision tree.
The proposed ensemble model is shown in Fig 1 below.

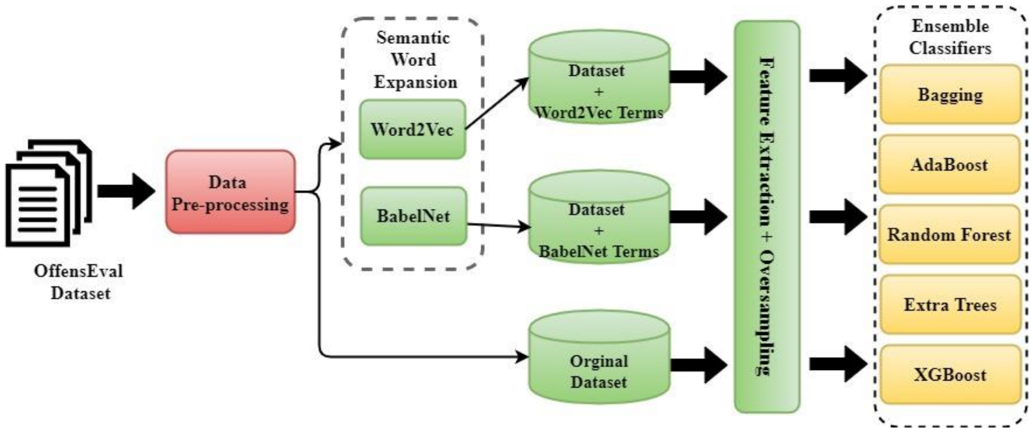

Fig 1. Images showing the visual symptoms cause by fungal disease

4.1. AdaBoost

AdaBoost, which was first introduced by Freund and Schapire, is a popular ensemble algorithm based on reweighting of training data [47]. The main idea behind AdaBoost is to concentrate on samples that are harder to classify. The algorithm starts by assigning the same weight to each sample, and then increases the weights of the misclassified samples, while decreasing the weights of the correctly classified samples, serially, at each iteration [48].

4.2. Bagging

Bagging, which was developed by Breiman is based on resampling of training data [49]. In this model, resampling is realized with selection of a set of samples randomly from the training set to comprise new training sets. Then base classifiers are applied to resampled datasets in parallel and obtained results are aggregated by using majority voting approach.

4.3. Random Forest

Random Forest is another ensemble method devised by Breiman that utilizes ensemble of several unpruned decision trees [50]. Random Forest, one of the most successful ensemble methods in the literature, trains decision tree classifiers on bootstrapped samples like bagging [51].

4.4. Extra-trees

Extra-trees were proposed by Geurts et al. that is based on randomization and construct totally randomized trees [52]. As in RF, also extra-trees construct ensemble of unpruned decision trees based on top-down strategy. This ensemble model chooses cut-points randomly and builds the trees by using all training data.

4.5. XGBoost

Chen and Guestrin developed XGboost as gradient boosted based decision tree ensembles for sparse data [53]. XGBoost algorithm with its successful results has become very popular for real world-applications. The main reason why this algorithm is so successful is its scalability, generalization performance and faster learning features. This greedy algorithm ranks the data according to eigenvalues [54].

5. Experimental Setup

5.1. Dataset

In realized experiments, we use subset of the Offensive Language Identification Dataset which was provided in SemEval 2019 Task 6, to evaluate our models. This dataset contains 13240 tweets which are tagged as offensive or not offensive. The number of tweets are increased to 14000 by oversampling. Offensive tagged tweets contain profanity, offensive and hate speeches. 4640 tweets are labeled as offensive and the rest of the tweets are labeled as not offensive. After preprocessing step input data is expanded with concepts obtained from BabelNet and word-embeddings obtained from Word2Vec separately. Thus, we create three different dataset representations to build and test our classification models. The first dataset is established with only raw data while the second and the third one are separately established by adding concepts and word-embedding vectors. The varying number of features in the established datasets are summarized in Table 1.

Table 1. Summary of datasets

Dataset	*Number of features*
Raw	12156
Raw+BabelNet	65100
Raw+Word2Vec	39693

5.2. Evaluation Measures

In the experiments, the F-score measure is used to evaluate classification models. F-score is harmonic mean of precision and recall values. Precision (p) is used to measure quality of positive predictions and is obtained by dividing number of correctly classified positive samples (tp) to sum of correctly (tp) and incorrectly (fp) classified positive samples. Recall (r) is used to measure quantity of correct positive predictions and is obtained by dividing number of correctly classified positive samples (tp) to sum of correctly classified positive samples (tp) and incorrectly classified negative samples (fn). Precision does not contain number of fn and likewise recall does not contain number of fp. That's why we used F-score to take more accurate measurements from our models.

$$\mathrm{F-score} = 2 \times \frac{p \times r}{p + r} \tag{2}$$

5.3. Conduction of Experiments and Results

In our study, we develop decision tree and decision tree based ensemble classifiers to classify offensive tweets. Besides traditional bagging and bosting models we also use RF, extra-trees and XGBoost classifiers. In the implementation of classification and oversampling processes we utilize scikit-learn and imbalanced-learn libraries in python. We split three input datasets into training and test set using a ratio of 80% for training data and 20% for test data. We use Gini index for node split criterion to overcome overfitting problem in decision tree learning. We stop splitting a node to reduce overfitting if impurity decreases lower than 10^{-4} in decision tree, Bagging and AdaBoost. The reason why we choose 10^{-4} is sparsity. Because of sparsity, impurity decreases slightly at each node.

Bagging and AdaBoost are established with 100 base classifiers. Sub-datasets are half the size of the entire dataset and are selected with bootstrapping. In here bootstrap is used to reduce variance.

Random forest and extra-trees are established with 200 base classifiers. While splitting nodes, feature space is partitioned randomly into sub-spaces because of overfitting. Size of each subspace is equal to square root of the size of the original feature space.

Same as RF and Extra-trees, we establish XGBoost with 200 base classifiers. On XGBoost, tree depth is limited with 30 and the learning rate is selected as 10^{-1}. Results are combined by averaging probabilistic prediction of each single classifier. Based on these parameter values, the evaluation results of whole models based on F-score are given in Table 2.

As shown in the Table 2 ensemble models affect the success of classification algorithms in a positive way. Tree based ensembles are superior to decision tree model. Extra-trees is the most successful model among all models and extended dataset with Word2Vec increases the performances of all models. BabelNet extended datasets don’t influence over model performance a lot and their performance are close to trained models with raw datasets.

Performances of BabelNet and Word2Vec extended datasets over classification models are examined and we have seen that context based extension is very effective concept based extension in terms of offensive language detection. Because of high number of features in BabelNet extended dataset decreases model performance very much.

Table 2. Classification results (F-score)

Ensemble techniques	***Datasets***		
	Raw	***Raw+BabelNet***	***Raw+Word2Vec***
Adaboost	73.85%	73.06%	80.29%
Bagging	76.02%	75.55%	81.63%
Random Forest	81.53%	81.42%	84.31%
Extra-trees	84.62%	83.74%	**85.66%**
XGBoost	74.33%	75.03%	81.76%
Decision Tree	67.28%	67.56%	77.77%

6. Conclusion

The use of offensive language is one of the major social media problems of today. To address this problem effectively many methods have been tried in the literature and some of them have been concluded accurately. However, in terms of researchers there is always a quest for the better. Therefore, we aim to improve offensive language detection in Twitter and to achieve this we use ensemble methods.

Short texts are problematic for classification tasks. Herewith, the original data set has been expanded in two different ways to solve the short text problem and to create different views of data. We expanded original dataset with concepts that obtained from BabelNet and word-embedding vectors from Word2Vec. Three datasets are given as input data to decision tree and decision tree based ensemble models which are bagging, boosting, RF, extra-trees and XGBoost.

When the experimental results are analyzed, it can be concluded that the most successful classification is obtained with extra-trees classifier where word-embedding vectors are given as the input to classifier with F-score of 85.66%.

References

[1] K. Denecke and W. Nejdl, “How valuable is medical social media data? Content analysis of the medical web,” *Inform Sciences*, vol. 179, no. 12, pp. 1870–1880, May. 2009.

[2] M. Wiegand, M. Siegel, and J. Ruppenhofer, “Overview of the GermEval 2018 Shared Task on the Identification of Offensive Language,” in *Proc. KONVENS,* Vienna, Austria, 2018, pp. 1–10.

[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 *Proc. NAACL-HLT,* Minneapolis, Minnesota, USA, 2019, pp. 1415–1420.

[4] M. Zampieri, S. Malmasi, P. Nakov, S. Rosenthal, N. Farra, and R. Kumar, “SemEval-2019 Task6: Identifying and Categorizing Offensive Language in Social Media (OffensEval),” in *Proc. SemEval,* Minneapolis, Minnesota, USA, 2019, pp. 75–86.

[5] J. Mitrovic, B. Birkeneder, and M. Garnitzer, “nlpUP at SemEval-2019 Task6: A Deep Neural Language Model for Offensive Language Detection,” in *Proc. SemEval,* Minneapolis, Minnesota, USA, 2019, pp. 722–726.

[6] B. Pang, L. Lee, and S. Vaithyanathan, “Thumbs up? Sentiment classification using machine learning techniques,” in *Proc. EMNLP*, Philadelphia, USA, 2002, pp. 79–86.

[7] P-W. Liang and B-R. Dai, “Opinion Mining on Social Media Data,” in *Proc. MDM,* Italy, 2013, pp. 91–96.

[8] S. Sevim, S. İlhan Omurca, and E. Ekinci, “An Ensemble Model using a BabelNet Enriched Document Space for Twitter Sentiment Classification,” *IJISAE*, vol. 10, no. 1, pp. 24–31, Oct. 2018.

[9] Z. Faguo, Z. Fan, Y. Bingru, and Y. Xingang, “Research on Short Text Classification Algorithm Based on Statistics and Rules,” in *Proc. ISECS,* NW Washington, DC, USA, 2010, pp. 3–7.

[10] I. Alsmadi and K. H. Gan, “Review of short-text classification,” *IJWIS*, vol. 15, no. 2, pp. 155–182, June. 2019.

[11] J. Tang, X. Wang, H. Gao, X. Hu, and H. Liu, “Enriching short text representation in microblog for clustering,” *Front Comput Sci*, vol. 6, no. 1, pp. 88–101, Jan. 2012.

[12] Z. Liu, W. Yu, W. Chen, S. Wang, and F. Wu, “Short Text Feature Selection for Micro-blog Mining,” in *Proc. ACL-HLT,* Portland, Oregon, USA, 2011, pp. 151–160.

[13] L. Jiang, M. Yu, M. Zhou, X. Liu, and T. Zhao, "Target-dependent Twitter Sentiment Classification," in *Proc. CiSE,* Wuhan, China, 2010, pp. 1–4.

[14] K. Kim, B-S. Chung, Y. Choi, S. Lee, and J-Y. Juang, "Language independent semantic kernels for short-text classification," *Expert Syst Appl*, vol. 41, no. 2, pp. 735–743, Feb. 2014.

[15] P. Wang, J. Xu, B. Xu, C-H. Liu, H. Zhang, F. Wang, and H. Hao, "Semantic Clustering and Convolutional Neural Network for Short Text Categorization," in *Proc. ACL- IJCNLP,* Beijing, China, 2015, pp. 352–357.

[16] B. Wang, J. Xu, B. Xu, G. Tian, C-L. Liu and H. Hao, "Semantic expansion using word embedding clustering and convolutional neural network for improving short text classification," *Neurocomputing*, vol. 174, no. B, pp. 806–814, Feb. 2016.

[17] N. Sotthisopha and P. Vateekul, "Improving Short Text Classification Using Fast Semantic Expansion on Multichannel Convolutional Neural Network," in *Proc. IEEE/ACIS SNPD,* Busan Gwang'yeogsi · South Korea, 2018, pp. 182–187.

[18] B. V. Dasarathy and B. V. Sheela, "A Composite Classifier System Design: Concepts and Methodology," *Proceedings of the IEEE*, vol. 67, no. 5, pp. 708–713, May. 1979.

[19] E. Ekinci and H. Takçı, "Comparing Ensemble Classifiers: Forensic Analysis of Electronic Mails," in *Proc. INSODE,* Antalya, Turkey, 2013, pp. 167–173.

[20] R. Xia, C. Zong, and S. Li, "Ensemble of feature sets and classification algorithms for sentiment classification," *Inform Sciences*, vol. 181, no. 6, pp. 1138–1152, Mar. 2011.

[21] M. Tutek, I. Sekulic, P. Gombar, I. Paljak, P. Culinovic, F. Boltuzic, M. Karan, D. Alagic, and J. Snajder, "TakeLab at SemEval-2016 Task 6: Stance Classification in Tweets Using a Genetic Algorithm Based Ensemble," in *Proc. SemEval,* San Diego, California, 2016, pp. 464–468.

[22] Z. H. Kilimci and S. İlhan Omurca, "A Comparison of Extended Space Forests for Classifier Ensembles on Short Turkish Texts," in *Proc. AC-EITAI,* Prague, Czech Republic, 2017, pp. 96–104.

[23] Z. H. Kilimci and S. İlhan Omurca, "Extended Feature Spaces Based Classifier Ensembles for Sentiment Analysis of Short Texts," *Inf Technol Control*, vol. 47, no. 3, pp. 457–470, Sep. 2018.

[24] P. Liu, W. Li, and L. Zou, "NULI at SemEval-2019 Task6: Transfer Learning for Offensive Language Detection using Bidirectional Transformer," in *Proc. SemEval,* Minneapolis, Minnesota, USA, 2019, pp. 87–91.

[25] E. Kebriaei, S. Karimi, N. Sabri, and A. Shakery, "Emad at SemEval-2019 Task 6: Offensive Language Identification using Traditional Machine Learning and Deep Learning approaches," in *Proc. SemEval*, Minneapolis, Minnesota, USA, 2019, pp. 600–603.

[26] P. Aggarwal, T. Horsmann, M. Wojatzki, and T. Zesch, "LTL-UDE at SemEval-2019 Task6: BERT and Two-Vote Classification for Categorizing Offensiveness," in *Proc. SemEval*, Minneapolis, Minnesota, USA, 2019, pp. 678–682.

[27] P. Rani and A. Kr. Ojha, "KMI−Coling a tSemEval-2019 Task6: Exploring N-grams for Offensive Language detection," in *Proc. SemEval,* Minneapolis, Minnesota, USA, 2019, pp. 668–671.

[28] G-V. Patraş, D-F. Lungu, D. Gifu, and D. Trandabat, "Hope at SemEval-2019 Task 6: Mining social media language to discover offensive language," in *Proc. SemEval,* Minneapolis, Minnesota, USA, 2019, pp. 635–638.

[29] P. Kapil, A. Ekbal, and D. Das, "NLP at SemEval-2019 Task6: Detecting Offensive language using Neural Networks," in *Proc. SemEval*, Minneapolis, Minnesota, USA, 2019, pp. 587–592.

[30] H. Bansal, D. Nagel, and A. Soloveva, "HAD-Tübingen at SemEval-2019 Task6: Deep Learning Analysis of Offensive Language on Twitter: Identification and Categorization," in *Proc. SemEval,* Minneapolis, Minnesota, USA, 2019, pp. 622–627.

[31] L. Balasubramanian, H. S. Kumar, G. Bandlamudi, D. Sivasankaran, R. Sivanaiah, A. D. Suseelan, S. M. Rajendram, and M. T. N. Thanagathai, "TECHSSN at SemEval-2019 Task6: Identifying and Categorizing Offensive Language in Tweets using Deep Neural Networks," in *Proc. SemEval*, Minneapolis, Minnesota, USA, 2019, pp. 753–758.

[32] D. Thenmozhi, B. S. Kumar, C. Aravindan and S. Srinethe, "SSN_NLP at SemEval-2019 Task6: Offensive Language Identification in Social Media using Traditional and Deep Machine Learning Approaches," in *Proc. SemEval,* Minneapolis, Minnesota, USA, 2019, pp. 739–744.

[33] G. L. De la P. Sarracen and P. Rosso, "Deep Analyzer at SemEval-2019 Task6: A deep learning-based ensemble method for identifying offensive tweets," in *Proc. SemEval,* Minneapolis, Minnesota, USA, 2019, pp. 582–586.

[34] J. Han, S. Wu, and X. Liu, "jhan014 at SemEval-2019 Task6: Identifying and Categorizing Offensive Language in Social Media," in *Proc. SemEval,* Minneapolis, Minnesota, USA, 2019, pp. 652–656.

[35] B. Wang, X. Zhou, and X. Zhang, "YNUWB at SemEval-2019 Task6: K-max pooling CNN with average meta-embedding for identifying offensive language," in *Proc. SemEval*, Minneapolis, Minnesota, USA, 2019, pp. 818–822.

[36] V. Indurthi, B. Syed, M. Shrivastava, M. Gupta, and V. Varma, "Fermi at SemEval-2019 Task6: Identifying and Categorizing Offensive Language in Social Media using Sentence Embeddings," in *Proc. SemEval,* Minneapolis, Minnesota, USA, 2019, pp. 611–616.

[37] A. Oberstrass, J. Romberg, A. Stoll, and S. Conrad, "HHU at SemEval-2019 Task6: Context Does Matter-Tackling Offensive Language Identification and Categorization with ELMo," in *Proc. SemEval,* Minneapolis, Minnesota, USA, 2019, pp. 628–634.

[38] S. D. Swamy, A. Jamatia, B. Gamback, and A. Das, "NIT_Agartala_NLP_Team at SemEval-2019 Task6: An Ensemble Approach to Identifying and Categorizing Offensive Language in Twitter Social Media Corpora," in *Proc. SemEval,* Minneapolis, Minnesota, USA, 2019, pp. 696–703.

[39] C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky, "The Stanford Corenlp Natural Language Processing Toolkit," in *Proc. ACL,* Baltimore, Maryland, 2014, pp. 55–60.

[40] E. Ekinci and S. İlhan Omurca, "Concept-LDA: Incorporating Babelfy into LDA for aspect extraction", *J. Inf. Sci.*, Accessed on: Apr., 29, 2019, DOI: 10.1177/0165551519845854, [Online].

[41] D. Tang, F. Wei, N. Yang, M. Zhou, T. Liu, and B. Qin, "Learning Sentiment-Specific Word Embedding for Twitter Sentiment Classification," in *Proc. ACL,* Baltimore, Maryland, 2014, pp. 1555–1565.

[42] H. K. Kim, H. Kim, and S. Cho, "Bag-of-concepts: Comprehending document representation through clustering words in distributed representation," *Neurocomputing*, vol. 266, pp. 336–352, Nov. 2017.

[43] A. Ali, S. M. Shamsuddin, and A. L. Ralescu, "Classification with class imbalance problem: A Review," *IJASCA*, vol. 7, no. 3, pp. 176–204, Nov. 2015.

[44] M. Galar, A. Fernandez, E. Barrenechea, H. Bustince, and F. Herrera, "A Review on Ensembles for the Class Imbalance Problem: Bagging, Boosting-, and Hybrid-Based Approaches," *IEEE Trans. Syst. Man Cybern. B. Cybern.*, vol. 42, no. 4, pp. 463–484, Aug. 2011.

[45] A. D. McDonald, J. D. Lee, C. Schwarz, and T. L. Brown, "Steering in a Random Forest: Ensemble Learning for Detecting Drowsiness-Related Lane Departures," *Hum. Factors*, vol. 56, no. 5, pp. 986–998, Aug. 2014.

[46] L. Rokach, "Ensemble-based classifiers," *Artif. Intell. Rev.*, vol. 33, pp. 1–39, Feb. 2014.

[47] Y. Freund and R. E. Schapire, "Experiments with a New Boosting

Algorithm," in *Proc. ICML,* Bari, Italy, 1996, pp. 325–332.

[48] E. Ekinci, S. İlhan Omurca, and N. Acun, "A Comparative Study on Machine Learning Techniques using Titanic Dataset," in *Proc. ICAT,* Antalya, Turkey, 2018, pp. 411–416.

[49] L. Breiman, "Bagging Predictors," *Mach.Learn.*, vol. 24, pp. 123–140, Sep. 1994.

[50] L. Breiman, "Random Forests," *Mach.Learn.*, vol. 45, pp. 5–32, Sep. 2001.

[51] S. İlhan Omurca, E. Ekinci, B. Çakmak, and S. G. Özkan, "Using Machine Learning Approaches for Prediction of the Types of Asthmatic Allergy across the Turkey," *DataSCI.*, vol. 2, no. 2, pp. 8–12, Dec. 2019.

[52] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Mach.Learn.*, vol. 63, pp. 3–42, Sep. 2006.

[53] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in *Proc. KDD,* San Francisco, CA, USA, 2016, pp. 785–794.

[54] Z. Chen, F. Jiang, Y. Cheng, X. Gu, W. Liu, and J. Peng, "XGBoost Classifier for DDoS Attack Detection and Analysis in SDN-based Cloud," in *Proc. BigComp,* Shanghai, China, 2018, pp. 251–256.