SAKARYA UNIVERSITY JOURNAL OF COMPUTER AND INFORMATION SCIENCES
VOL. 5, NO. 1, APRIL 2022
DOI: 10.35377/saucis.05.01. 1070822

Research/Review Article

Classification of Imbalanced Offensive Dataset – Sentence Generation for Minority Class with LSTM

Ekin Ekinci[1]
[1]Sakarya University of Applied Sciences, Faculty of Technology, Computer Engineering Department;
ekinekinci@subu.edu.tr; ORCID: 0000-0003-0658-592X

Received 10 February 2022; Revised 16 April 2022; Accepted 18 April 2022; Published online 30 April 2022

Abstract

The classification of documents is one of the problems studied since ancient times and still continues to be studied. With social media becoming a part of daily life and its misuse, the importance of text classification has started to increase. This paper investigates the effect of data augmentation with sentence generation on classification performance in an imbalanced dataset. We propose an LSTM based sentence generation method, Term Frequency-Inverse Document Frequency (TF-IDF) and Word2vec and apply Logistic Regression (LR), Support Vector Machine (SVM), K Nearest Neighbor (KNN), Multilayer Perceptron (MLP), Extremely Randomized Trees (Extra tree), Random Forest, eXtreme Gradient Boosting (Xgboost), Adaptive Boosting (AdaBoost) and Bagging. Our experiment results on an imbalanced Offensive Language Identification Dataset (OLID) that machine learning with sentence generation significantly outperforms.

Keywords: sentence generation, imbalance classification, offensive language, deep learning, machine learning

1. Introduction

Social media is an indispensable part of daily life and the way people communicate with each other is now shaped by this environment. Communication via social media is not only limited to people we know but is also frequently used in communication with people we do not know. This can be accomplished in a variety of ways, such as writing a comment under a photo, answering a question, or commenting on a topic. Although the free environment that social media offers to its users creates a situation where we can express all kinds of ideas in any way as if there are no restrictions, it is absolutely necessary to stay within certain limits. However, nowadays, it is seen that social media is used in the wrong ways that infringe on people's rights. Violation of human rights over social media is usually carried out with the use of offensive language. Offensive language use is a big problem in social media, leading to a large amount of research in detecting content against cyberbullying and hate speech [1]. Therefore, it is very critical to discover offensive language. Over the last ten years, a lot of progress has been done in the field of automatic detection and classification of offensive language and other related phenomena. [2-5]. The determination of whether a language is offensive or not is carried out by supervised methods. When dealing with problems involving imbalanced classification, however, supervised approaches encounter challenges. Imbalanced classification is one of the most important research topics to be handled in machine learning [6, 7]. Because the performance criteria for classification algorithms are designed to reduce classification error, while classifiers perform well on the majority class, they perform poorly on the minority class, which is the focus of attention [8]. To overcome class imbalance, oversampling techniques have an important place in the literature. Oversampling is defined as adding samples to a minority class and carried out in many different ways such as Random Oversampling (ROS), synthetic minority over-sampling technique (SMOTE), BorderLine SMOTE (B-SMOTE), K-Means SMOTE, Safe Level SMOTE (SL-SMOTE), SMOTE-Nominal and Continuous (SMOTE-NC), Adaptive Synthetic Sampling Approach (ADASYN) [9, 10]. ROS equalizes class membership by selecting and multiplicating samples from minority classes until their balance is reached [11]. SMOTE is one of the most widely used oversampling techniques. SMOTE realizes oversampling by taking samples of each minority class and creates new synthetic examples by

using the sample and some of its KNN from the minority class [12]. B-SMOTE creates new samples by using minority borderline and other minority samples to determine the new samples [13]. K-Means SMOTE is the hybridization of the k-means clustering and SMOTE [14]. SL-SMOTE generates synthetic examples along line segments as in SMOTE but locates them near the largest safe level [15]. SMOTE-NC is a variant of SMOTE that supports discrete values [16]. ADASYN proposes sample distribution learning efficiently [17]. As a consequence, the aim of this article is to provide a new solution to imbalanced data using a deep learning algorithm to provide an effective solution for classifying offensive language. In the experiments, Offensive Language Identification Dataset (OLID)-sub-task A, which consists of English tweets annotated for offensive language, is used. The goal of this sub-task is to classify offensive language as offensive (OFF) and not-offensive (NOT). The data augmentation is realized by using LSTM, for feature representation two different methods namely TF-IDF and Word2vec are used. Then for both representation classification is realized by using base classifiers which are LR, SVM, KNN, MLP, and ensemble classifiers which are Extra-tree, Random Forest, Xgboost, AdaBoost and Bagging. Experiments are conducted on original and augmented datasets and the best results are achieved with TF-IDF represented augmented dataset by LR and SVM with an average 82% F-score.

The rest of the paper is organized as follows. Section 2 reviews studies in the literature on offensive language classification. In Section 3 pre-processing, feature representation, LSTM and the classification algorithms are explained. In Section 4 proposed data augmentation model is explained. In Section 5 experiments and results are given in detail. In the last section, the paper is concluded.

2. Related Works

Here, in chronological order, we provide a brief summary of the studies from literature in which sub-task A of OLID from the shared task SemEval-2019 Task 6 on Identifying and Categorizing Offensive Language in Social Media (OffensEval) was used [18].

Rozental et al. [19] proposed a Multiple Choice Convolutional Neural Network (MC-CNN) which was fed into contextual embedding generated from Twitter and achieved a 0.7868 F-score. In [20] Temporal Convolutional Neural Network with an attention layer was applied and the authors received a 0.4682 F-score. Also, they claimed that preserving as many even numbers of samples for each class as possible could increase classification accuracy when the dataset had a class imbalance ratio. Kumar et al. extracted uni-gram and bi-gram features and then used Linear SVM for classification [21]. The F-score obtained was 0.5282. Wu et al. applied an uncased BERT model pre-trained on model files with a 0.8057 F-score [22]. In [23], multi-layer RNN extracted features -benefited from ELMo embeddings, self-attention, character n-grams and node2vec- were given to gradient-boosted decision trees (GBDT) for classification and a 78.79% F1-score was obtained. Zang et al. proposed a 2-layer residual connected BiDirectional LSTM (BiLSTM) with double attention and their F1-score was 0.768 [24]. This architecture was designed as BiLSTM for contextual feature extraction, residual connected layer for deep feature synthesis and one attention mechanism for the extraction of semantic information of emojis and the other for output. Pavlopoulos et al. compared Perspective which is an API and BERT for offensive language classification and with a 0.7933 F1-Score Perspective had the upper hand over BERT [25]. To cope with class imbalance Modha et al. used pre-trained word vectors and applied LSTM, BiLSTM, CNN, and Stacked CNN [26]. However, they concluded that the proposed solution was not successful in classification according to the 0.7833 F-score achieved. In another study, an ensemble of CNN and RNN was used and low performance with a 0.5925 F-score was realized [27]. Pedersen trained logistic regression, data augmentation and logistic regression and a rule-based black–list approach [28]. Among these three approaches, the rule-based black–list was the best with a 0.73 F-score. Kebriaei et al. proposed machine learning algorithms, deep learning algorithms and the ensemble of these algorithms, and also, realized data augmentation from different sources [29]. The most successful F-score (0.76) was obtained with the application of SVM to expanded data. Pelicon et al. achieved a 0.8078 F-score with BERT [30]. In [31] with machine learning algorithms that were fed into multiple sentence embeddings 64.40% F-score was obtained. Results showed that the deep model dominated the SVM models with a 0.7793 F-score. LSTM classifiers with SVM predictions were used in three different ways

in Bansal et al.'s study [32]. In the first experiment LSTM classifier with SVM, predictions were used directly, in the second and third ones, classification was applied with the help of lookup list and hashtag parsing, respectively. The best F-score (0.7327) was achieved with the hashtag parsing approach. Oberstrass et al. proposed 6 different LSTM models and among these models LSTM trained with chars, words, stems and LSTM outputs gave the best F-score (0.767) [33]. Pătraș et al. aimed at three different models one of them was rule-based, the second one was lexicon-based and the last and best one with a 0.6446 F-score was external sources and offensive words in the training data [34]. Graff et al. applied B4MSA, FastText, and EvoMSA for classification and obtained a 0.774 F-score with EvoMSA [35]. In [36], 6 different architectures were trained and the most successful among them was RNN with an F-score of 0.74. In this RNN architecture, uni-gram and bi-gram features were used and in a fully connected (FC) layer, seven different machine learning models were performed. Pal et al. applied different machine learning and deep learning algorithms [37]. While the best among machine learning algorithms was LR-tri-gram with 0.7231 F-score, among deep learning algorithms the best was CNN-Glove with 0.7844 F-score. Rani and Ojha aimed to learn the impact of n-grams on offensive language detection [38]. It was observed that successful results were obtained (79.76%) with 1 gram from 1-4 grams. In [39], BiLSTM-Att -BiLSTM with attention layer- was proposed to extract semantic features and a 0.7682 F-score was obtained. In another study, BERT achieved state-of-the-art performance with a 0.798 F-score [40]. In Doostmohammadi et al.'s study comparison was made between SVM which took the word and character n-grams and BERT representations as input separately and the character-based deep model [41]. To cope with imbalanced class distribution, Ekinci et al. used oversampling, concept and word-embedding vectors based on expansion [4]. Then, applied weak and ensemble classifiers to the dataset and achieved an 85.66% F-score. Oswal made several experiments based on classification algorithms, feature representation techniques and sampling methods [42]. The best results were obtained with the original dataset by using LSTM-glove and achieved a 0.72 F-score. In [43] the authors devised fBert which was trained on 1.4 million offensive data from the SOLID dataset to deal with the imbalanced class problem. Compared to benchmarks, fBert outperformed with a 0.813 F-score. Muslim et al. improved fine-tuned BERT using a cost-sensitive and ensemble model and obtained an F1 score of 0.8207 [44].

3. Methodology and Applied Techniques

3.1 Pre-processing

Data pre-processing is a must, necessitating the use of a technique known as data cleaning, which transforms raw data into a machine-readable format while separating noise from data [45]. In particular, social media data, such as Twitter data, must be subjected to a data cleaning step due to the special and non-ASCII characters, punctuation, numbers, misspellings, URLs, links, and @-mentions, retweets (RT), hashtags and emoticons it contains. To correct misspelled words autoencoder library of Python is used. So, all of these noises are removed from the dataset at hand. Then, other noises such as stopwords namely articles, prepositions and conjunctions are removed. For stopwords removing the stopword list in Python is used. All tweets have been converted to lowercase due to case insensitivity. As a final step, stemming is used to present each word on a stem or root base. Then, to give dataset as input to model some model specific pre-processing is realized. At first, a vocabulary is created which contains distinct words in the dataset. Every distinct word in the vocabulary is tokenized and every token is associated with a specific number. These token-numbers are used to represent sentences. For each sentence, number of token lengts (T) minus one word n-grams (2 n-grams, 3 n-grams, …, T n-grams) are created and these created n-grams are padded to be of equal size. The last element of this obtained sequence is the label and the remaining elements are the feature. Finally, the labels are encoded with one-hot encoding. To achieve these steps Python's Natural Language Toolkit (NLTK) is used.

3.2 Text Representation Models

3.2.1 TF-IDF

The less a term appears in a document, the less informative that term provides and the less weight it is given. TF-IDF algorithm is widely used feature weighting techniques in Natural Language Processing (NLP) tasks. The reason why it is preferred so much is that it is simple and yet it is a very powerful model. The statistical method of multiplying TF and IDF to determine the significance of a word is known as TF-IDF. The formula for TF-IDF is given with Equation 1 below.

$$\boldsymbol{d_{i,j} = tf_{i,j} \times log \frac{N}{n_i}} \tag{1}$$

The weight of the term i in the document j is represented by the equation above $d_{i,j}$ represents. The frequency of the term i in document j is $tf_{i,j}$ and IDF of the term i is $log\, N/n_i$. In the IDF, while N represents the number of documents, n_i shows how many documents the term i occurs in.

3.2.2 Word2vec

Word2vec is one of the word embedding algorithms that maps each word to vectors that represent the semantic meaning of the word [46]. Word2vec, built on artificial neural network (ANN) architecture, is an unsupervised learning algorithm. Word2vec takes a big corpus of texts as input and generates a real-vector space with hundreds of dimensions by exploiting word context information. A vector in space represents each word in the corpus. In this vector space, distances between words can be calculated and semantically similar ones are also close to each other. Thanks to this method, some semantic conclusions can also be drawn for realizing NLP tasks.

Continuous Bag of Words (CBOW) and Skip-gram are two learning models in Word2vec. These two models were built over the n-gram model. While the CBOW model predicts a target word by using context words, the Skip-gram model predicts the context words of a particular word [47]. The input, projection, and output layers are the three layers of the CBOW and skip-gram models. The neural network (NN) architectures of CBOW and Skip-gram models are given in Figure 1.

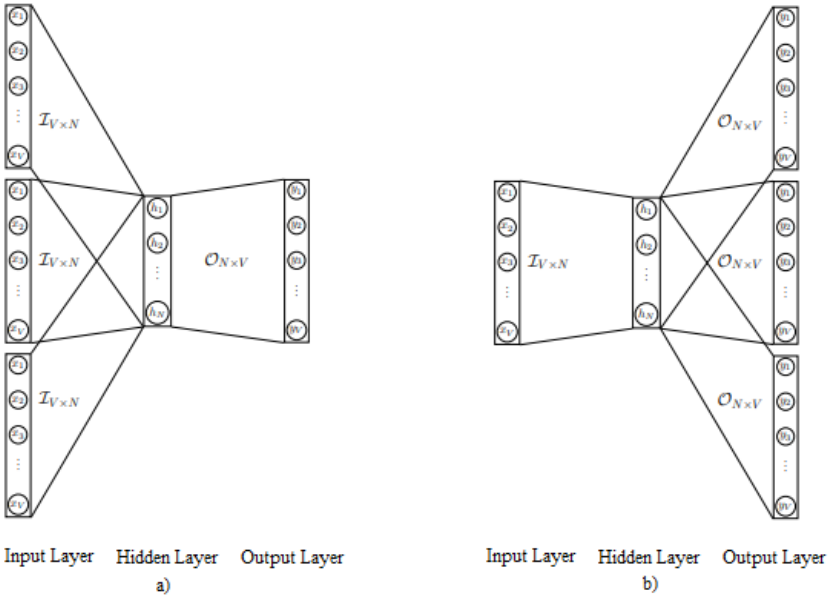

Figure 1 a) NN Architecture of CBOW b) NN Architecture of Skip-gram [48]

In the CBOW model, context words are represented with word vectors of one-hot encoded and each the size of the vocabulary (V). The input layer consists of as many neurons as one-hot encoded context

words. The input layer is fully connected with the hidden layer. So connections must be stored in a weight matrix ($I_{V\times N}$). N is the number of neurons in the hidden layer. The hidden layer and the output layer are also fully connected. And weight matrix between them is represented with $O_{N\times V}$. The output layer consists of V neurons and the output of this layer is the one-hot encoded target word.

In the Skip-gram model, the input layer consists V neurons which represent the one-hot-encoded word. As in the CBOW model, the input layer and hidden layer and hidden layer and output layer are fully connected in this model. $I_{V\times N}$ represents the weight matrix between the input and hidden layer. There are N neurons in the hidden layer. The weight matrix between the hidden layer and output layer is represented with $O_{N\times V}$. The output layer consists of as many neurons as one-hot encoded context words each the size of the vocabulary.

Both CBOW and Skip-gram models find the output by using working logic of ANN.

3.3 LSTM

RNN is the name given to specialized neural networks for processing sequential data. In traditional ANN, the output is directly obtained from inputs. RNN is devised to overcome this drawback by adding cyclic connections to hidden neurons. In this way, between input and output a sequence-to-sequence mapping is realized, that is, the output is obtained based on the previous computation. However, the insufficient memory capacity of the RNN -single layer neural network with feedback loop- for long time steps requires a new architecture that has strong memorization ability.

LSTM with the repetitive module which has four operations (3 sigmoid and 1 tanh) provides memorization of information for these long time steps [49]. LSTM is a special RNN architecture with its hidden neurons composed of a memory cell and three gates namely the forget gate, input gate and output gate [50, 51].

The forget gate selects which information is taken off from the cell state (memory) based on the previous hidden state (h_{t-1}) and the current input (x_t). If the forget gate's output is 0, all information is forgotten, if it is 1, all information is taken into account. The mathematical expression of the forget gate is given with Equation 2:

$$\boldsymbol{f_t = \sigma\left(w^f x_t + w^f h_{t-1} + b^f\right)} \tag{2}$$

where f_t is the output of forget gate, σ represents the sigmoid function. w^f indicates weight and b^f indicates bias of the forget gate.

The input gate determines which information is updated and added to the cell state. The output of input gate i_t is given with Equation 3:

$$\boldsymbol{i_t = \sigma\left(w^i x_t + w^i h_{t-1} + b^i\right)} \tag{3}$$

where w^i indicates weight and b^i indicates bias.

The cell state of the network, and is updated by using Equations 4 and 5:

$$\boldsymbol{\widetilde{C}_t = tanh(w^c x_t + w^c h_{t-1} + b^c)} \tag{4}$$

$$\boldsymbol{C_t = i_t \odot \widetilde{C}_t + f_t \odot C_{t-1}} \tag{5}$$

where $\tilde{C}_t$ represents the candidate values for cell state C_t. w^c indicates weight and b^c indicates bias of the cell state. Element-wise multiplication is represented with $\odot$.

What information in the cell state will be used is determined with the output gate. The output of this gate is given with Equation 6 and hidden state (h_t) for time t Equation 7:

$$\boldsymbol{o_t = \sigma\left(w^o x_t + w^0 h_{t-1} + b^0\right)} \tag{6}$$

$$\boldsymbol{h_t = o_t \odot tanh(C_t)} \tag{7}$$

where w^o and b^o are the weight and bias of the output gate, respectively.

3.4 Classification Algorithms

3.4.1 Logistic Regression

LR is a statistical binary classification method that uses regression analysis to predict a categorical variable based on one or more predictive variables [52]. LR is built on the principle that the value of the dependent variable is estimated using independent variables. In the model, while the class label Y is the dependent variable, X $(x_1, x_2, \dots, x_n)$ is the set of independent variables which corresponds to attributes and used to predict Y.

3.4.2 Support Vector Machines

SVM is a powerful classification algorithm grounded on statistical learning theory and has the capacity to classify the data that is both linear and nonlinear. The core idea behind this algorithm is to separate the two classes by finding the optimal separating hyperplane, that is, the decision boundary [53]. The SVM uses support vectors and margins to find this hyperplane. If the data is linearly separable, a separation hyperplane can easily separate data in the two-dimensional space. However, to separate non-linear data converting this data to a higher dimension is necessary. At this stage, SVM uses non-linear mapping functions for this purpose. Training can be slow, but classification accuracy is high with its success in modelling complex and nonlinear decision boundaries.

3.4.3 K Nearest Neighbor

The most important disadvantage of parametric methods is that they need prior knowledge about the distribution of the data. KNN is an easy-to-implement and non-parametric algorithm used when there is fewer or no prior knowledge about data distribution. There is no need to iterate this algorithm for tuning the parameters [54]. To assign a class label to a test instance, at first, the algorithm finds KNNs to that instance in training samples. Then, it assigns the class label that is most common in the k closest training instances to the test instance. In general, to calculate the distance between instances KNN uses Euclidean distance.

3.4.4 Multilayer Perceptron

MLP is a layered feed-forward ANN trained with a backpropagation algorithm used for separating non-linear data [55]. The input layer, hidden layer(s), and output layer form the layered structure. In the input layer, the number of neurons is equal to the number of features. The number of hidden layers and neurons in each layer are determined through trial and error. In the output layer, there are as many neurons as there are classes. Nodes in each layer are fully interconnected with nodes in the next layer. All connections have weights. Initially, these weights are randomly determined. After every iteration, the resulting error propagates backwards until the error falls below a certain value or a certain iteration occurs to adjust the weights.

3.4.5 Extremely Randomized Trees

Extra tree is an ensemble learning algorithm that makes classification by combining results of independent decision trees into a forest [56]. Although it is a type of random forest, decision trees are created differently in this method. Each decision tree in extra trees is built using the whole dataset with feature subsets that are assembled at random.

3.4.6 Random Forest

Random forest is an ensemble classifier that employs a large number of unpruned decision trees. Each decision tree that composes the forest is created with selected samples from the training dataset with the bootstrapping technique [57]. A randomly selected subset of features is used to separate the data according to the heterogeneity measure.

3.4.7 eXtreme Gradient Boosting

Xgboost is a faster, scalable and effective version of the gradient boosting decision tree (GBDT). The scalability of the Xgboost is so named because it can handle missing values without preprocessing [58], can process weights of instances and can work in parallel and distributed [59]. In Xgboost Classification and Regression Tree (CART) is used.

3.4.8 Adaptive Boosting

AdaBoost is widely used ensemble learning algorithms. Adaboost follows the logic of combining weak classifiers to create a strong classifier [60]. In this algorithm, at first, all training samples have the same weight and in the training process reweighting is realized. While correctly classified samples have a lower weight, erroneously classified samples have a higher weight. Thus, misclassified samples will become more important in the next iteration. This process will continue for all iterations and all samples will be reclassified at each iteration. It determines the classes of data samples by majority voting of decisions from weak classifiers applied in series.

3.4.9 Bagging

Bagging, which suggests a resampling mechanism is the oldest ensemble classifier [61]. Instead of making decisions according to the classification results obtained on a single dataset, a weak classifier is applied in parallel with the resampled training datasets in Bagging. Then, it determines the classes of data samples by majority voting of decisions from each training model.

4. Proposed Data Augmentation Method

In the data augmentation process, LSTM architecture is used. The input layer is designed as an embedding layer that takes the tokens. In this layer input dimension is the number of total words in OFF labelled tweets. The output dimension, which defines the size of the output embedded vectors for each token, is 10. Input dimension is equal to maximum sequence length minus one. We define only one hidden LSTM layer with 100 memory units. To prevent overfitting, the proposed network uses dropout with a probability of 10. The output layer is designed as a Dense layer with a softmax activation function and as an output, determines the probability of the best probable next word. And the number of iterations is equal to 100. The architecture of the model is After the model has been trained, the tokenized sequence given as input to the model. The architecture is given in Figure 2.

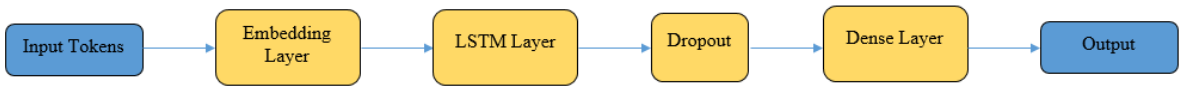

Figure 2 The Proposed Architecture

After each iteration, a new word is predicted as output of the model depending on the context. This output is added to previous input and combined input is given to model as a next input. If the input of the architecture is like “what the hell be”, one of the outputs of the architecture is “what the hell be like one see moron get drop”. For data augmentation, the Keras library of Python is used. In the experiments, the used model parameters are selected by using a grid search over multiple initialization seeds. From input to output, the generation is represented in Figure 3.

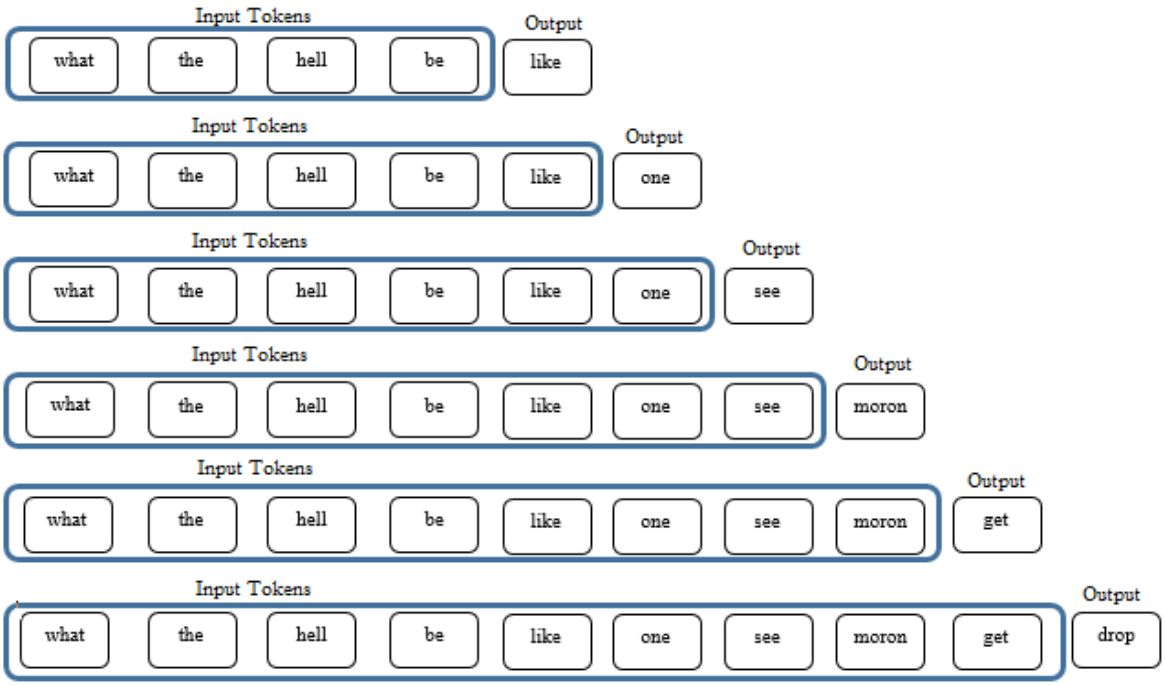

Figure 3 Sentence Generation

5. Experimental Study

5.1 Dataset

In the experiments Offensive Language Identification Dataset (OLID)- sub-task A is used. There are 13240 tweets in the dataset, with 8840 being labelled as not-offensive (NOT) and 4400 as offensive (OFF). The statistics of the imabalanced dataset are given in Table 1. As it is understood, the available dataset is an imbalanced dataset and an LSTM-based oversampling is applied to OFF labelled data within the scope of the study to increase the classification accuracy. With the application of LSTM based oversampling, the number of OFF tweets is 8840 and the total data is 17680.

Table 1 Statistics of the Imbalanced Dataset

		# of Term	**# of Average Terms Per Tweet**	**Average Term Length**
Label	NOT	160.719	18.18	4.26
	OFF	91.504	20.8	4.24

5.2 Performance Metrics

In order to assess the proposed models, we preferred F-score as the main evaluation metric. F-score is the harmonic mean of precision and recall. Precision is (p) the ratio of correctly classified not offensive tweets (tp) to all not offensive tweets (tp+fp). Recall (r) is the ratio of correctly classified not offensive tweets (tp) to all tweets that are classified as not offensive (tp+fn). The confusion matrix to calculate performance metrics is given in Table 1. The formula of the F-score is given with Equation 9.

Table 2 Confusion Matrix

		Actual	
		NOT	OFF
Predicted	NOT	tp	fp
	OFF	fp	tn

$$F - Score = \frac{2 \times p \times r}{p + r} \tag{9}$$

5.3 Realization of Experiments and Performance Results

In this study, we apply basic and ensemble classifiers to classify offensive tweets. For the implementation of classification algorithms scikit-learn library of Python is used. We divide the original and augmented datasets into train and test sets. As a train set, we take 80% of the dataset, as a test set, we take 20% of the dataset. While, in the training and test sets of the original dataset there are 10592 and 2648 samples, respectively, in the training and test sets of the augmented dataset there are 14144 and 3536 samples, respectively.

In the experiments, the l2-regularized LR algorithm with settings inverse regularization parameter C = 10 and the liblinear optimization algorithm are used. The reason why we set the 10 to C parameter is that the smaller C provides better regularization. In the SVM, regularization parameter c = 1 with a squared L2 penalty and RBF kernel are used. For the KNN algorithm, the selection of the k is important. So, to make a decision on k we use the Elbow method and see that the best value is 2 for both datasets. All points in each neighbourhood have equal weight and the Euclidean metric is preferred as a distance measure. While designing MLP 100 hidden layers are used, the activation function is relu and the solver is adam. 0.001 is assigned to the learning rate and 300 is assigned to the maximum iteration count.

The number of base classifiers is set to 100 when creating an extra tree classifier. As a split criteria Gini index is used. Minimum impurity decrease is defined as 10^{-3}. The number of base trees in the Random forest is assigned to 100. As in Extra-tree, Gini is used as the split criterion for this algorithm. The tree's maximum depth has been set to 2. The number of base classifiers is set to 100 in Xgboost, as in other extra tree and random forest. The maximum depth of the tree is selected as 6 and the learning rate is selected as 0.1. In AdaBoost, 50 base estimators are used. In Bagging, the number of base estimators is 10 and bags are composed by replacement of samples.

The results for the original dataset are given in Table 3. When the results are examined for the original dataset it is clearly seen that TF-IDF representation is better than Word2vec representation in most cases in terms of performance achieved. Among the algorithms, while LR is the best for TF-IDF representation, KNN, Extra tree and Bagging are the best for Word2vec representation.

Table 3 Classification Results for Original Dataset (F-score)

	Text Representations	
Classifiers	**TF-IDF**	**Word2vec**
LR	0.76	0.53
SVM	0.73	0.52
KNN	0.56	0.58
MLP	0.71	0.52
Extra tree	0.76	0.58
Random Forest	0.52	0.52
Xgboost	0.72	0.53
AdaBoost	0.75	0.53
Bagging	0.74	0.58

The results for the augmented dataset are given in Table 4. As seen in the original dataset, TF-IDF representation performed better than Word2vec representation in most cases in augmented data. In addition, it is clearly seen that success has increased in the results obtained with both representations with data augmentation. Among the algorithms, while LR and SVM are the best for TF-IDF representation, MLP and Xgboost are the best for Word2vec representation.

Table 4 Classification Results for Augmented Dataset (F-score)

	Text Representations	
Classifiers	**TF-IDF**	**Word2vec**
LR	0.82	0.75
SVM	0.82	0.75
KNN	0.75	0.75
MLP	0.76	0.77
Extra tree	0.81	0.76
Random Forest	0.71	0.74
Xgboost	0.80	0.77
AdaBoost	0.80	0.75
Bagging	0.81	0.75

6. Conclusions

In this paper, we investigate the problem of how to improve classification performance on the imbalanced dataset. To achieve this, we conduct a comparative study of imbalanced data classification using LSTM based sentence generation method. For text representation, TF-IDF and Word2vec are used and LR, SVM, KNN, MLP, and ensemble classifiers which are Extra-tree, Random Forest, Xgboost, AdaBoost and Bagging algorithms are used to train the classifiers. Experimental results on the augmented dataset reveal that the TF-IDF based LR and SVM increase the F-score by 6% and 9%, respectively. We also investigate that classification with TF-IDF-based text representation often performs well on both original and augmented datasets compared to classification with Word2vec.

In the future, we will apply different sentence generation methods with different text representation methods to improve performance.

References

[1] S. Rosenthal, P. Atanasova, G. Karadzhov, M. Zampieri, and P. Nakov, "OLID: A Large-Scale Semi-Supervised Dataset for Offensive Language Identification," *arXiv preprint arXiv:2004.14454*, 2020.

[2] G. Wiedemann, E. Ruppert, R. Jindal and C. Biemann, "Transfer Learning from LDA to BiLSTM-CNN for Offensive Language Detection in Twitter," *arXiv preprint arXiv:1811.02906v1*, 2018.

[3] H. Mubarak and K. Darwish K., "Arabic Offensive Language Classification on Twitter," *Lecture Notes in Computer Science*. Springer, Cham, 2019.

[4] E. Ekinci, S. İlhan Omurca and S. Sevim, "Improve Offensive Language Detection with Ensemble Classifiers," *IJISAE*, vol. 8, no. 2, pp. 109–115, 2020.

[5] M. Djandji, F. Baly, W. Antoun and H. Hajj, "Multi-Task Learning using AraBert for Offensive Language Detection," *Proc. - 4th Workshop on Open-Source Arabic Corpora and Processing Tools, with a Shared Task on Offensive Language Detection*, pp. 97–101, 2020.

[6] Y. Tung and Y. Q. Zhang, "Granular SVM with Repetitive Undersampling for Highly Imbalanced Protein Homology Prediction," *Proc. - 2006 IEEE International Conference on Granular Computing*, pp. 457–460, 2006.

[7] J. Brownlee, *Imbalanced Classification with Python.* Machine Learning Mastery, 2020.

[8] Q. Zou, S. Xie, Z. Lin, M. Wu and Y. Ju, "Imbalanced classification is one of most popular topics in the field of machine learning," *Big Data Res.*, vol. 5, pp. 2–8, 2016.

[9] L. Wang, H. Cheng, Z. Zheng, A. Yang and X. Zhu, " Ponzi scheme detection via oversampling-based Long Short-Term Memory for smart contracts," *Knowl Based Syst.*, vol. 228, pp.1–12, 2021.

[10] A. Gosain and S. Sardana, "Handling Class Imbalance Problem using Oversampling Techniques: A Review," *Proc. - 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pp. 79–85, 2017.

[11] E. L. Iglesias, A. S. Vieira and L. Borrajo, " An HMM-based over-sampling technique to improve text classification," *Expert Syst. Appl.*, 465, pp. 1–20, 2013.

[12] N. V. Chawla, K. W. Bowyer, L. O. Hall, W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-

sampling Technique," *J Artif Intell Res.*, vol. 16, pp. 321–357, 2002.

[13] H. A. Majzoub, I. Elgedawy, Ö. Akaydın and M. Köse Ulukök, " HCAB-SMOTE: A Hybrid Clustered Afnitive Borderline SMOTE Approach for Imbalanced Data Binary Classifcation," *Arab. J. Sci. Eng.*, vol. 45, pp. 3205–3222, 2020.

[14] G. Douzas, F. Bacao and F. Last, "Improving imbalanced learning through a heuristic over-sampling method based on k-means and SMOTE," Inf. Sci., vol. 465, pp. 1–20, 2018.

[15] C. Bunkhumpornpat, K. Sinapiromsaran and C. Lursinsap, "Safe-Level-SMOTE: Safe-Level-Synthetic Minority Over-Sampling TEchnique for Handling the Class Imbalanced Problem," *Proc. - Pacific-Asia conference on knowledge discovery and data mining*, pp. 475–482, 2009.

[16] S. Darabi and Y. Elor, "AE-SMOTE: A Multi-Modal Minority Oversampling Framework," pp. 1–19, 2020.

[17] A. Amin, S. Anwar, A. Adnan, M. Nawaz, N. Howard, J. Qadir, A. Hawalah and A. Hussain, "Comparing Oversampling Techniques to Handle the Class Imbalance Problem: A Customer Churn Prediction Case Study," *IEEE Access*, vol. 4, pp. 7940–7957, 2016.

[18] M. Zampieri, S. Malmasi, P. Nakov, S. Rosenthal, N. Farra and R. Kumar, " Predicting the Type and Target of Offensive Posts in Social Media," *Proc. - NAACL-HLT*, pp. 1415–1420, 2019.

[19] A. Rozental and D. Biton, "Amobee at SemEval-2019 Tasks 5 and 6: Multiple choice over contextual embedding," *arXiv preprint arXiv:1904.08292.*, 2019.

[20] M. Sridharan and T. R. Swapna, "Amrita School of Engineering-CSE at SemEval-2019 Task 6: Manipulating attention with temporal convolutional neural network for offense identification and classification," *Proc. - 13th International Workshop on SemEval*, pp. 540–546, 2019.

[21] R. Kumar, G. Bhanodai, R. Pamula, and M. R. Chennuru, "bhanodaig at SemEval-2019 Task 6: Categorizing offensive language in social media," *Proc. - 13th International Workshop on SemEval*, pp. 547–550, 2019.

[22] Z. Wu, H. Zheng, J. Wang, W. Su and J. Fong, "Bnu-hkbu uic nlp team 2 at semeval-2019 task 6: Detecting offensive language using bert model," *Proc. - 13th International Workshop on SemEval*, pp. 551–555, 2019.

[23] G. Aglionby, C. Davis, P. Mishra, A. Caines, H. Yannakoudakis, M. Rei, E. Shutova and P. Buttery, "CAMsterdam at SemEval-2019 Task 6: Neural and graph-based feature extraction for the identification of offensive tweets," *Proc. - 13th International Workshop on SemEval*, pp. 556–563, 2019.

[24] Y. Zhang, B. Xu and T. Zhao, "CN-HIT-MI. T at SemEval-2019 Task 6: Offensive Language Identification Based on BiLSTM with Double Attention," *Proc. - 13th International Workshop on SemEval*, pp. 564–570, 2019.

[25] J. Pavlopoulos, N. Thain, L. Dixon and I. Androutsopoulos, "Convai at semeval-2019 task 6: Offensive language identification and categorization with perspective and bert," *Proc. - 13th International Workshop on SemEval*, pp. 571–576, 2019.

[26] S. Modha, P. Majumder, D. Patel, "DA-LD-Hildesheim at SemEval-2019 task 6: tracking offensive content with deep learning using shallow representation," *Proc. - 13th International Workshop on SemEval*, pp. 577–581, 2019.

[27] G. L. De la Peña and P. Rosso, "DeepAnalyzer at SemEval-2019 Task 6: A deep learning-based ensemble method for identifying offensive tweets," *Proc. - 13th International Workshop on SemEval*, pp. 582–586, 2019.

[28] T. Pedersen, "Duluth at SemEval-2019 task 6: Lexical approaches to identify and categorize offensive tweets," *arXiv preprint arXiv:2007.12949*, 2019.

[29] 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," *Proc. - 13th International Workshop on SemEval*, pp. 600–603, 2019.

[30] A. Pelicon, M. Martinc and P. K. Novak, "Embeddia at semeval-2019 task 6: Detecting hate with neural network and transfer learning approaches," *Proc. - 13th International Workshop on SemEval*, pp. 604–610, 2019.

[31] V. Indurthi, B. Syed, M. Shrivastava, M. Gupta and V. Varma, "Fermi at SemEval-2019 Task 6: Identifying and categorizing offensive language in social media using sentence embeddings," *Proc. - 13th International Workshop on SemEval*, pp. 611–616, 2019.

[32] H. Bansal, D. Nagel and A. Soloveva, "HAD-Tübingen at SemEval-2019 Task 6: Deep learning analysis of offensive language on Twitter: Identification and categorization," *Proc. - 13th International Workshop on SemEval*, pp. 622–627, 2019.

[33] A. Oberstrass, J. Romberg, A. Stoll and S. Conrad, "HHU at SemEval-2019 Task 6: Context does matter-tackling offensive language identification and categorization with ELMo," *Proc. - 13th International Workshop on SemEval*, pp. 628–634, 2019.

[34] G. F. Patras, D. F. Lungu, D. Gifu and D. Trandabat, "Hope at SemEval-2019 Task 6: Mining social media language to discover offensive language," *Proc. - 13th International Workshop on SemEval*, pp. 635–638, 2019.

[35] M. Graff, S. Miranda-Jiménez, E. Tellez and D. A. Ochoa, "INGEOTEC at SemEval-2019 task 5 and task 6: A genetic programming approach for text classification," *Proc. - 13th International Workshop on SemEval*, pp. 639–644, 2019.

[36] Y. HaCohen-Kerner, Z. Ben-David, G. Didi, E. Cahn, S. Rochman and E. Shayovitz, "JCTICOL at SemEval-2019 Task 6: Classifying offensive language in social media using deep learning methods, word/character n-gram features, and preprocessing methods," *Proc. - 13th International Workshop on SemEval*, pp. 645–651, 2019.

[37] P. Mukherjee, M. Pal, S. Banerjee and S. K. Naskar, "JU_ETCE_17_21 at SemEval-2019 Task 6: Efficient Machine Learning and Neural Network Approaches for Identifying and Categorizing Offensive Language in Tweets," *Proc. - 13th International Workshop on SemEval*, pp. 662–667, 2019.

[38] P. Rani and A. K. Ojha, "KMI-coling at SemEval-2019 task 6: exploring N-grams for offensive language detection," *Proc. - 13th International Workshop on SemEval*, pp. 668–671, 2019.

[39] L. S. M. Altın, À. B. Serrano and H. Saggion, "Lastus/taln at semeval-2019 task 6: Identification and categorization of offensive language in social media with attention-based bi-lstm model," *Proc. - 13th International Workshop on SemEval*, pp. 672–677, 2019.

[40] P. Aggarwal, T. Horsmann, M. Wojatzki and T. Zesch, "LTL-UDE at SemEval-2019 Task 6: BERT and two-vote classification for categorizing offensiveness," *Proc. - 13th International Workshop on SemEval*, pp. 678–682, 2019.

[41] E. Doostmohammadi, H. Sameti and A. Saffar, "Ghmerti at SemEval-2019 task 6: a deep word- and character-based approach to offensive language identification," *arXiv preprint arXiv:2009.10792*, 2020.

[42] N. Oswal, "SemEval-2019 (OffensEval): Identifying and Categorizing Offensive Language in Social Media," *arXiv preprint arXiv: 2104.04871v1*, 2021.

[43] D. Sarkar, M. Zampieri, T. Ranasinghe and A. Orarbia, "fBERT: A Neural Transformer for Identifying Offensive Content," *arXiv preprint arXiv: 2109.05074v1*, 2021.

[44] F. Muslim, A. Purwarianti and F. Z. Ruskanda, "Cost-Sensitive Learning and Ensemble BERT for Identifying and Categorizing Offensive Language in Social Media," *Proc. - ICAICTA*, pp. 1–6, 2021.

[45] A. S. Neogi, K. A. Garg, R. K. Mishra and Y. K. Dwivedi, "Sentiment analysis and classification of Indian farmers' protest using twitter data," *Int. J. Inf. Manage.*, vol. 1, no. 2, pp. 100019, 2021.

[46] E. M. Dharma, F. L. Gaol, H. L. H. S. Warnars and B. Soewito, "The Accuracy Comparison Among Word2vec, Glove, And Fasttext Towards Convolution Neural Network (CNN) Text Classification," *J. Theor. Appl. Inf.*, vol. 100, no. 2, pp. 349–359, 2022.

[47] M. S. Başarslan and F. Kayaalp, " Sentiment Analysis on Social Media Reviews Datasets with Deep Learning Approach," *SAUCIS*, vol. 4, no. 1, pp. 35–49, 2021.

[48] J. V. Lochter, P. R. Pires, C. Bossolani, A. Yamakami and T. A. Almeida, " Evaluating the impact of corpora used to train distributed text representation models for noisy and short texts," *Proc. - 2018 International Joint Conference on Neural Networks*, pp. 1–8, 2018.

[49] A. Zhao, L. Qi, J. Dong and H. Yu, "Dual channel LSTM based multi-feature extraction in gait for diagnosis of Neurodegenerative diseases," *Knowl. Based Syst.*, vol. 145, pp. 91–97, 2018.

[50] B. Kaya and A. Günay, "Twitter Sentiment Analysis Based on Daily Covid-19 Table in Turkey," *SAUCIS*, vol. 4, no. 3, pp. 302–311, 2021.

[51] Y. Canbay, A. İsmetoğlu and P. Canbay, " Deep Learning and Data Privacy in Diagnosis of Covid-19," *J. Eng. Sci. Technol.*, vol. 9, no. 2, pp. 701–715, 2021.

[52] E. Ekinci, S. İlhan Omurca and N. Acun, "A Comparative Study on Machine Learning Techniques using Titanic Dataset," *Proc. - 7th International Conference on Advanced Technologies*, pp. 411–416, 2018.

[53] D. Chen, H. Bourlard and J. P. Thiran, "Text identification in complex background using SVM," *Proc. - 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 621–626, 2001.

[54] M. Jogin, M. S. Madhulika, G. D. Divya, R. K. Meghana, and S. Apoorva, "Feature Extraction using Convolution Neural Networks (CNN) and Deep Learning," *Proc. - 3rd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology*, pp. 2319–2323, 2018.

[55] S. Lallahem, J. Mania, A. Hani and Y. Najjar, "On the use of neural networks to evaluate groundwater levels in fractured media," *J. Hydrol.*, vol. 307, no. 1-4, pp. 92–111, 2005.

[56] K. Kaur and S. K. Mittal, "Classification of mammography image with CNN-RNN based semantic features and extra tree classifier approach using LSTM," *Mater. Today.*, pp. 1–7, 2020.

[57] S. Sevim, E. Ekinci and S. İlhan Omurca, "Multi-view Document Classification with Co-training," *Proc. - 28th IEEE Conference on Signal Processing and Communications Applications*, pp. 1–4, 2020.

[58] D. A. Rusdah and H. Murfi, "XGBoost in handling missing values for life insurance risk prediction," *SN Appl. Sci.*, vol. 2, no. 8, pp. 1–10, 2020.

[59] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," *arXiv preprint arXiv: 1603.02754v3*.

[60] E. Ekinci and H. Takçı, "Comparing ensemble classifiers: Forensic analysis of electronic mails," *Global Journal on Technology*, vol. 4, no. 2, pp. 167–173, 2013.

[61] G. Liang, X. Zhu, and C. Zhang, "An empirical stıdy of bagging predictors for different learning algorithms," *Proc. - AAAI'11*, pp. 1802–1803, 2011.