Ratio Mathematica Volume 42, 2022

A Deep Learning Model for Classifying the Hate and Offensive Language in Social Media Text

Nidhi Bhandari[1]
Rachna Navlakhe[2]
G.L. Prajapati[3]

Abstract

Social media is a big data source for analyzing content. It requires a significant amount of computational cost. Recently introduced model for identifying and removing toxic content from Twitter, using an Information Retrieval (IR) model is working accurately but with small amount of data. Therefore, in this paper, a deep learning technique is used to process large-scale social media text data. First, it uses Natural Language Processing (NLP) based feature extraction to create four different sets of training samples i.e. TF-IDF-based features, POS Tagged Features, a reduced feature vector of POS and the combined vector of TF-IDF and POS tagged features. The deep Convolutional Neural Networks (CNN) is used to train the model and to classify hate and offensive language. The dataset has been obtained from Kaggle. The performance in terms of training accuracy, validation accuracy, training loss and validation loss has been measured with the time complexity. In addition, the class-wise Precision, Recall, F1-score and Mean accuracy have also been investigated. From experimental results, we found TF-IDF and POS-based combined features provide 82% and 83% of accuracy, respectively.

Keywords: Text mining, social media, semantic knowledge, sentiment analysis, deep learning, hate and offensive language.[*]

[1]Department of Applied Mathematics, IET, DAVV, Indore. nidhi.bhandari1@gmail.com.
[2]Department of Applied Mathematics & Compu. Sci., SGSITS, Indore, sgsits.rachna@gmail.com.
[3]Department. of Comp. Engi., IET, DAVV, Indore, glprajapati1@gmail.com.
[*]Received on January 21st, 2022. Accepted on June 20th, 2022. Published on June 30th, 2022. doi: 10.23755/rm.v39i0.705. ISSN: 1592-7415. eISSN: 2282-8214.

1. Introduction

Information Retrieval (IR) is a technique used to locate precise information and supports different data formats, i.e. text, image, video, etc. Among these data formats, the text has a significant contribution. The text IR model uses text mining techniques. In IR techniques, text mining or data mining algorithms are employed to recover user query relevance information [1].The IR model contains three key components: (1) User query (2) Query processing (3) Generation of outcomes[2]. However, the deficiency of these components can impact the performance of IR model such as lack of specific user query keywords, inappropriate keyword selection, lack of similar data, ranking of results etc. [3].

Our study is initiated with an IR model named SOIR (Semantic query Optimization-based Information Retrieval). This IR model incorporates query optimization and an FCM clustering technique. The results indicate the performance of the SOIR is better than previous models [4]. However, there are various applications of the IR systems. Beyond these applications, the IR model can also be applied for pattern recognition. In this context, the proposed work is extended in order to be used with the social media toxic content filtering. This paper is an extension of the SOIR model, which will be used for classifying the toxic contents from the social media posts. This model is a promising technique for handling various negative tweets from social media by using lexical as well as semantic pattern analysis.

In this paper, we are going to evaluate the SOIR and DGM based toxic content classification system on a large dataset. Additionally, we do an investigation of a deep learning technique for classifying the different NLP based text features. We organize the contents in the following manner:

1. Background: This section discusses the previously introduced SOIR and Directed Graph Model (DGM) for text IR and toxic content classification.
2. Proposed work: This Section provides the understanding of the proposed Deep learning model for classifying hate speech, offensive language and normal text.
3. Results analysis: This section contains the experimental analysis of the proposed model and the different parameters have been provided.
4. Conclusion: In this section, finally the work is concluded and future extension of work has been suggested.

2. Background

This section provides insight into SOIR model and previously developed DGM classification model.

A. SOIR Model

This model is aimed to improve the IR model for running time and relevancy. Due to the large amount of data, a significant amount of time is required to locate the precise information. In the documents database, a number of documents are present which do not belong to the similar category and content. This nature of database increases search

space. Therefore, some improvements have been made to enhance the user query representation and subjective data categorization. The query optimization involves a semantic model to recognize similar words to optimize the query.
First, we pre-process the data to improve the quality of data and exclusion of noise by removing stop words and special characters. Next, feature selection techniques are used to reduce data dimensions and speed up the search process. Thus, the TF-IDF is used [5], to compute the weight W for identifying important tokens. The fixed size of the feature i.e. 30 tokens is considered. Further, FCM clustering is being used to categorize feature vectors [6]. The clustering results are organized as a list of features:

$$F =< F_n, k_{1,2,\dots n}, C >$$

where, F is the feature set, F_n is the file name or index, $k_{1,2,\dots n}$ is the list of keywords, and C is the class name or subject. The training feature vector F is stored in a database.
The categorized features are helpful for efficient data retrieval. On the other hand, the user query is transformed into a vector Q as a set of keywords:

$$Q = \{q_1, q_2, \dots, q_k\}$$

In order to optimize the query, we initialize a set of queries by using synonyms. Additionally, a map is prepared that contains the keywords and the synonyms. In this algorithm, a single keyword is twisted multiple times to generate new queries using similar words. The different search query increases the chances of finding accurate data. The search process is developed on the basis of the k-NN (k-nearest neighbour) algorithm [7]. The distance between the query and data less than 0.25 is counted as the result. The developed SOIR model has been compared with the Cosine similarity-based and k-NN-based IR models. The precision, Recall and F-Score are calculated and visualized in figure 1.

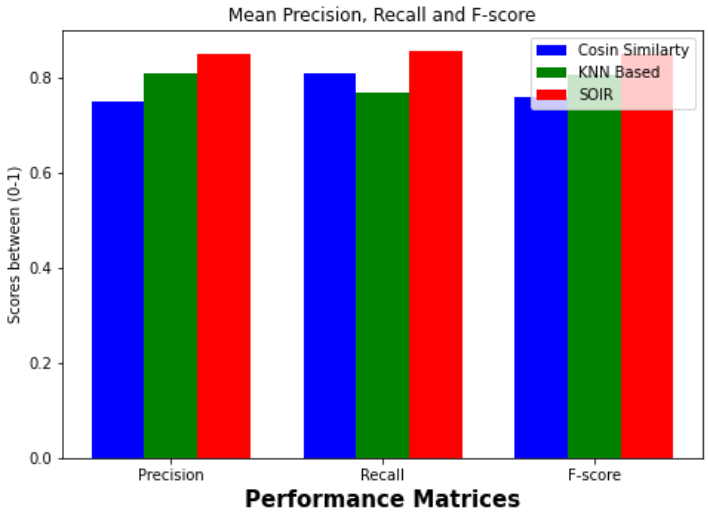

Figure 1. Mean Performance of SOIR Model

In this diagram, X-axis shows the measured matrix, and the Y-axis shows the precision, Recall and F-Score. According to the results, we found that the technique improves the precision, recall and f-score as we utilizing a large size of learning data.

B. SOIR for toxic content classification

In order to keep clean social media, we need an accurate model to identify malicious and toxic posts. We used an SOIR model for this task [4]. We summarized the negative emotions identified in table 1 below, which consists of Emotion classes and the relevant flow of emotions [8]. There are five main classes, and each class consists of its own subclasses. Thus, we need a multiclass classification system. So, we solve this problem by using the SOIR-based model.

Table 1. Emotion Classes

Emotions	**Content**
Distressed	Sad, Disappointed, Guilty, Missed
Surprised	Surprised
Fearful	Panic, Frightened, Shy
Angry	Angry
Disgusted	Dissatisfied, Annoyed, Doubtful, Hateful

In order to train the model, we have collected more than 3329 tweets and categorized them. Then we considered 2002 tweets for the experiment. The pre-processing has been applied for removing tags, special characters, and stop words. Further, part of speech tagging (POS) is used to understand the lexical structure [9]. The POS tags and sentiment classes are used to prepare a new dataset. On the other hand, tweets are also processed using the TF-IDF weights. The top 20 weighted tokens are picked and used. In the next process, we prepared thresholds using the tweets' POS tags. In the first step, we compute the mean feature of each feature group using:

$$M_F = \frac{1}{N}\sum_{i=1}^{N} F_i$$

Where M_F is the mean of the feature, N is the total number of samples in the groups. After measuring the mean value, we compute the distance from each point as the limit.

$$L_F = \frac{1}{N}\sum_{i=1}^{N} |F_i - M_F|$$

Thus the threshold of the feature can be defined as:

$$T_F{}^{Group} = M_F \pm L_F$$

Using this equation, we calculate the feature map FM for all the POS features. After that, we append features of the TF-IDF. Here the FCM clustering is used for creating the dictionary learning. We are just using the membership function of FCM for preparing the dictionary. The membership between data instance i and centroid j is measured using:

$$\mu_{i,j} = \frac{1}{\sum_{k=1}^{c} \left(\frac{d_{i,j}}{d_{i,k}}\right)^{\frac{2}{m-1}}}$$

According to this process, the emotion-labeled data is being used from each group. A tweet is tokenized and then inserted into the dictionary D with the TF-IDF weight. If a token exists in the dictionary, then we update the weight of the token and if the token is not available, then we simply insert the token. In order to compute the updated weight, we used:

$$NewWeight = 0.5 * Oldweight + 0.5 * \mu_{i,j}$$

where, $\mu_{i,j}$ is the membership between previous and new weight for the token.

The dictionaries are preserved in a database. Now we classify new tweets. For the classification, 50% of training samples and 50% of new tweets from Tweeter have been used. Then, we perform the POS tagging of test data. The tagged feature vector T_v is used to compare with the thresholds. For both pattern matching and decision making here the mean value M and limit L are used for computing upper threshold $L_{max}$and lower threshold L_{min}.The patterns between these two limits are used to compute the distance among queried tweets. POS tag feature and all the features threshold are used from feature map FM. Finally the higher matched value based class label is predicted as decision. After lexical pattern based decision making, we use the semantics for categorizing a tweet as final class label. Thus, the tweets are tokenized to regenerate it, as in SOIR. Let us have a tweet such that:

$$TW = \{k_1, k_2, \dots, k_n\}$$

After regeneration, we get:

$$TW_{m,n} = \begin{cases} k_{1,1}, k_{1,s}, \dots\dots\dots\dots, k_{1,n} \\ k_{2,1}, k_{2,2}, \dots\dots\dots\dots, k_{2,n} \\ k_{m,1}, k_{m,2}, \dots\dots\dots\dots, k_{m,n} \end{cases}$$

After preparing the set of similar keywords, we train the model using keywords and relevant weights for all the classes. The trained model can be defined as:

$$D_{o,p} = \begin{cases} D_1 = \{[T_{1,1}, W_{1,1}], [T_{1,2}, W_{1,2}] \dots\dots, [T_{1,p}, W_{1,p}]\} \\ D_2 = \{[T_{2,1}, W_{2,1}], [T_{2,2}, W_{2,2}] \dots\dots, [T_{2,p}, W_{2,p}]\} \\ D_o = \{[T_{o,1}, W_{o,1}], [T_{o,2}, W_{o,2}] \dots\dots, [T_{o,p}, W_{o,p}]\} \end{cases}$$

This is a function that generates a virtual directed graph using a weight matrix for decision-making. The matrix in the form of a graph model describes the association of a tweet with the given sentiment dictionary. An algorithm is developed to get the class label of the tweet, as given in table 2.

Table 2. DGM based Classification

Input: set of semantically similar twits $TW_{m,n}$, trained dictionary model $D_{o,p}$
Output: Class label C

Process:

1. $for(i = 1; i < m; i++)$
    a. $for(j = 1; j < n; j++)$
        i. $if\left(D_i.contains\left(TW_{i,j}\right)\right)$
            1. $W_i = W_{i-1} + W_{i,j}$
        ii. $end\ if$
    b. $end\ for$
    c. $DW_i = \frac{W_i}{m}$
2. End for
3. $C = getMaxVal(DW_i).classlabel$
4. Return C

The above algorithm searches each word in the dictionary and the relevant weights are aggregated. A greater value of weights is used as the final sentiment label. In order to make final decision, we use a function as:

$$f(B,C) = \begin{cases} B = C & then\ \ C \\ B! = C \quad is\ B_{i-1} == C & then\ C \end{cases}$$

Finally, if we find the Class label C belongs to shy, panic, sad and guilty then we label the tweet as nontoxic else C returns the toxic labels. This model is named as Directional Graph Model. Figure 2 below shows the precision, recall and f-score of different techniques. The results indicate the DGM shows better accuracy as compared to other models. However, the DGM performance is higher but the SOIR is consistent as compared to DGM.

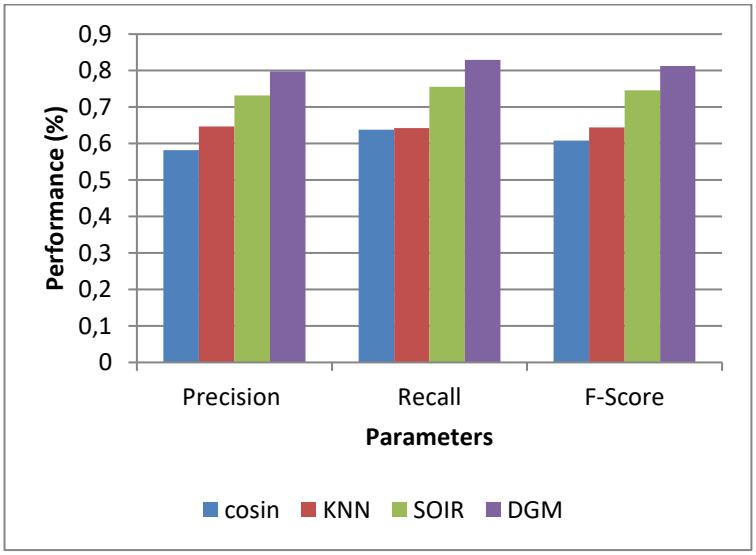

Figure 2. Performance summaries of classification models

Secondly, we measured the required time for training and reported it in figure 3. According to the results, the cosine-based and k-NN based techniques are winners. The main advantage of this model is that we can preserve the previously trained model for future use. Additionally, new data will also learn continuously. The model's performance replicates the efficient and accurate identification of toxic tweets.

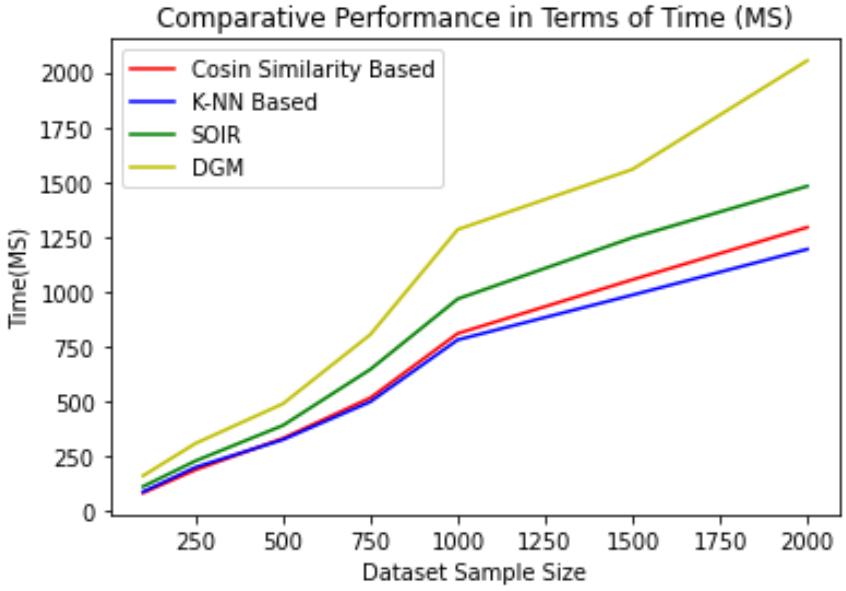

Figure 3. Time consumption

3. The proposed work

Emotion classification has become a common and classical domain of research and development but still there are many opportunities available for discovering something new. Initially, we started our work with the study of text mining and information retrieval. But due to new kinds of problems associated with text mining attracted us to classify social media text for finding toxic content. In order to design an effective method, we have implemented a DGM approach which is accurate and efficient for modelling negative emotions. But during the investigation, we found the following issues we tried to resolve in this study.

1. Social media content these days is full of offensive language even during formal communication, comments and others. Therefore, there is a need of distinguishing between the content as normal and toxic.
2. The previous experiments have been done with a small size of samples, but social media involve big data problems. Therefore, a previously defined model is reframed using the Deep learning technique to deal with a large amount of data.

In this context, we have obtained a dataset of Twitter from Kaggle. The dataset is hate speech and offensive language dataset. The dataset is distributed in three class labels hate speech, offensive and neither. The dataset involves 1430 instances as hate speech post of twitter, 19190 tweets as the offensive language and 4163 instances as the normal text. The dataset consists of 24783 instances of data. Among them, we make a split of 18587 instances for training and 6196 for testing. However, this dataset contains more data than the dataset on which we trained our previously designed models.

Therefore, initially, we have employed our DGM model for extracting features. During this, we found a significant drawback of this method, which we tried to demonstrate in figure 4. Figure 4 shows the time consumed during the experiment with only 50% of entire dataset for constructing the required features for DGM technique. However, the DGM based technique has accurate results but having huge time complexity for training with a large amount of data.

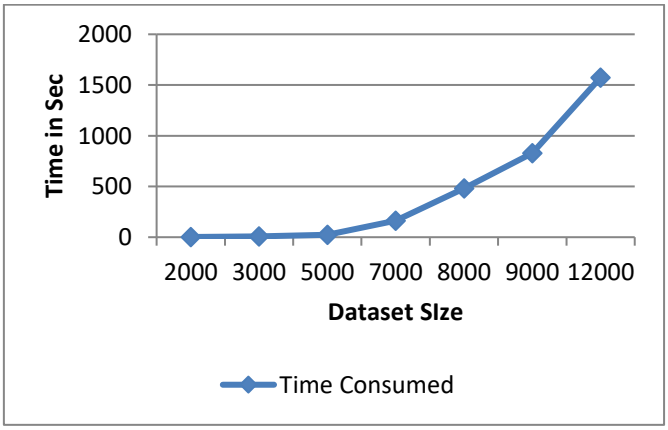

Figure 4. Time consumed with DGM

Therefore we drop the idea of utilizing the traditional classification technique for processing such a huge amount of data. The main reason behind the time consumption is computation of TF-IDF features and then computing the feature representation, which increases the time consumption significantly. Additionally, the measured features that need to train with the machine learning algorithms consume a significant amount of time for training. Therefore, the idea of working with the large amount of data with the classical and traditional feature construction method is becoming expensive. Finally, we proposed to use the Deep Learning technique for classifying the hate speech and offensive text to deal with the large amount of data. However, the deep learning models are mainly developed for image classification, but these techniques can also be used for other relevant classification and prediction tasks. The proposed model consists of the following steps to conduct the experiments.

Data preparation

In any machine learning model, data preparation is an essential step of data analysis. We have placed the obtained data from Kaggle into our Google drive and use the Google Colab to process data using the different techniques of text mining and machine learning.

	Unnamed: 0	count	hate_speech	offensive_language	neither	class	tweet
0	0	3	0	0	3	2	!!! RT @mayasolovely: As a woman you shouldn't...
1	1	3	0	3	0	1	!!!!! RT @mleew17: boy dats cold...tyga dwn ba...
2	2	3	0	3	0	1	!!!!!!! RT @UrKindOfBrand Dawg!!!! RT @80sbaby...
3	3	3	0	2	1	1	!!!!!!!!! RT @C_G_Anderson: @viva_based she lo...
4	4	6	0	6	0	1	!!!!!!!!!!!! RT @ShenikaRoberts: The shit you...

Figure 5. Dataset samples

In the first step, we load the dataset, the initial dataset samples are demonstrated in figure 5 with their relevant attributes. However, the dataset contains various attributes, but we are just utilizing the class labels and tweets here for data analysis.

Data pre-processing

In basic terminology, the data pre-processing is a technique by which we prepare the data to apply to the learning algorithm. Therefore in this context, we studied various NLP based text pre-processing techniques. From them, some essential techniques are applied for pre-processing of data. Here we applied the following pre-processing techniques:

a. Punctuation Removal: In this phase, we removed all the Punctuations from the input dataset instances by analyzing each character of the input strings.

b. Tokenization: In this step, the aim is to divide the tweet strings into chunks of words.

c. Stop-word Removal: In this step, the aim is to remove English stop words. The stop words are those words which frequently occur in the text but not essential for domain identification such as, "this', "that', "is', "'am' so on. In this context, we apply the Natural Language Toolkit (NLTK) library for stop word removal.

d. Stemming and Lemmatization: Both are generated form of the same words. The difference is that stem might not be an actual word and lemma is an actual language word. Stemming usage an algorithm to which makes word processing faster. On the other hand, in lemmatization, we used WordNet to produce lemma, which is slower than stemming.

	Unnamed: 0	count	hate_speech	offensive_language	neither	class	tweet
0	0	3	0	0	3	2	[rt mayasolovely as a woman you shouldnt comp...
1	1	3	0	3	0	1	[rt mleew17 boy dats coldtyga dwn bad for cuf...
2	2	3	0	3	0	1	[rt urkindofbrand dawg rt 80sbaby4life you ev...
3	3	3	0	2	1	1	[rt cganderson vivabased she look like a tranni]
4	4	6	0	6	0	1	[rt shenikaroberts the shit you hear about me...

Figure 6. Pre-processed data

Features analysis

In this phase, we transform the tweets into a structured manner. So, we have used the following different ways to prepare four different set of feature vector.

a. POS Tagging: In this phase, we utilized the NLTK based POS Tagger to process each string of the dataset. After that we prepared the vector for making it training and testing set. In this process we get 37 tags or features for training data. The transformed data is demonstrated in figure 7.

	NN	RB	IN	DT	PRP	VBP	VB	VBG	RP	PRP$	MD	NNS	JJ	CD	VBZ	VBD	CC	TO	VBN	JJR	WP	WDT	WRB	RBS	PDT
0	6.0	2.0	3.0	3.0	2.0	1.0	2.0	1.0	2.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	7.0	0.0	2.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	2.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	5.0	1.0	1.0	1.0	3.0	1.0	2.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
3	3.0	0.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	3.0	0.0	2.0	2.0	4.0	1.0	3.0	0.0	0.0	0.0	2.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0

Figure 7. Sample tagged data

b. POS with reduced features: In this phase, the 37 features of NLP-POS tags are used for experimentation. In order to reduce the amount of features, we have used Principle Component Analysis (PCA) and reduced the features, finally we have getting only 20 features for applying the algorithms.

c. Term Frequency – Inverted Document Frequency (TF-IDF): In this feature vector, we utilized TF-IDF Vectorizer to extract weighed keywords from the data. The dimensions of data are kept maximum 5000 keywords. The extracted features are demonstrated in figure 8.

	00sexilexi00	05professor	10	100	1000	10000	100046729	1008465039	100d	1017sqquad	1041204	10412041041204	1043359	11	12	1
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	

5 rows × 5000 columns

Figure 8. extracted TF-IDF features

d. Combined features: In this phase, we just combined TF-IDF features, and POS Tagged Features in a vertical manner to construct total of 5037 columns and 24783 rows of data.

Applying CNN

Here, we trained the high-dimensional data using Convolutional neural network (CNN). Therefore, we have developed a simple three-layer sequential model. First layer is configured with the total number of attributes in training features, and activation function "'ReLu' is used. The second hidden layer consists of 10 neurons and the activation function "'Sigmoid' has been used. The final layer contains three output neurons and 'Soft-'max' activation function. The model takes very less amount of time to train the CNN model. The training time of the input training vectors has given in figure 9. The model will be trained in a few times even the combined features, which consist of 5037 columns and 24783 data. Therefore, we can use the large data for learning with deep learning techniques.

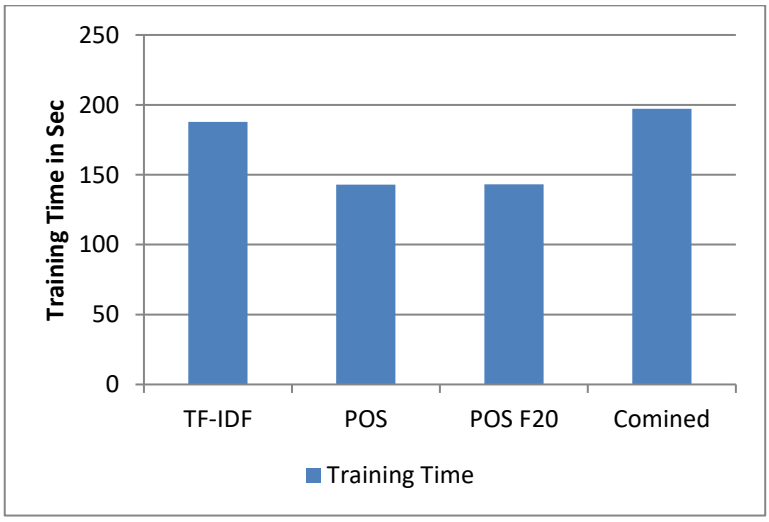

Figure 9. Training time

Performance analysis

In this phase, we evaluated our model for obtaining the performance matrix for learning as well as the validation of the model. Therefore, the training accuracy, validation accuracy, training loss and validation loss of the model have been measured. These performance parameters are discussed in next section. In addition, the class-wise performance report has also been prepared.

4. Results

This section details the conducted experiments and the obtained performance results. In deep learning, the accuracy metric is used to measure the 'algorithm's performance in an interpretable manner. The accuracy is determined in the form of a percentage. It measures how accurate our model's prediction is with the true data.

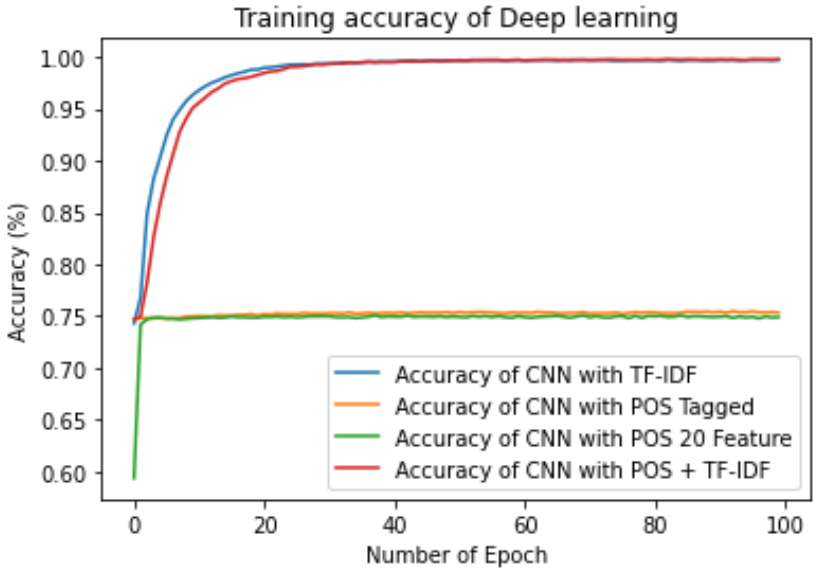

Figure 10. (A) Training Accuracy (%)

In this experiment, we have used four different set of feature vectors namely TF-IDF based weighted feature vector, second the POS tagged features, third we have reduce the size of POS tagged data for classification and finally the combined features of POS and TF-IDF has been employed with the CNN model. The performance of the model with the different sets of selected features in terms of accuracy (%) has been reported using figure 10(A). Additionally, the accuracy for validation or test dataset has been demonstrated in 10(B). According to the obtained consequences in terms of these two parameters, we found that the TF-IDF based and the combination of TF-IDF and POS

produces a higher performance than the other two algorithms. During training, we found the POS tagged data shows approximately 74.5 to 75.2% performance.

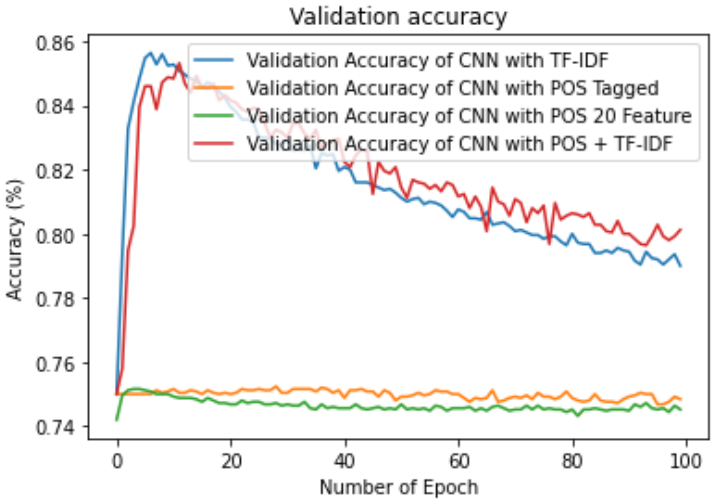

Figure 10. (B) Validation Accuracy (%)

On the other hand, when we used the reduced 20 features with the POS tagged data then we found a small improvement and we get an accuracy of 75.5%, which is higher than the POS tagged complete dataset, which has 36 features.

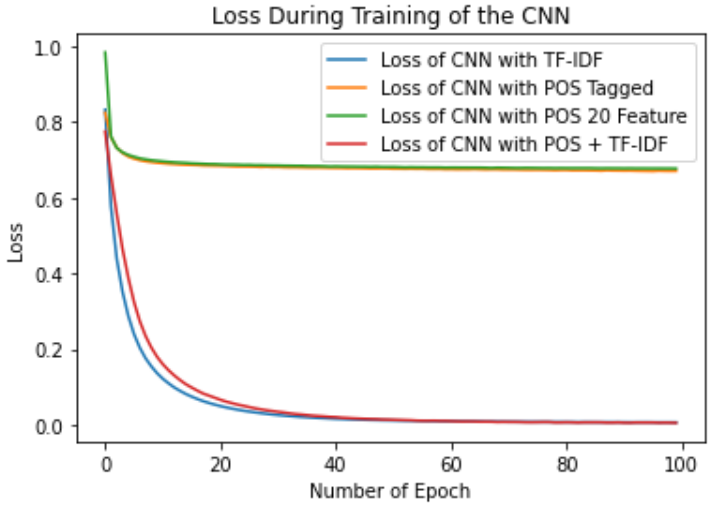

Figure 10. (C) Training Loss

On the other hand, when we train the model with the same data but TF-IDF features, model returns the total of 98.8% accurate results in consistent manner. And when we apply the combination of POS and TF-IDF features, we get improved results up to

99.8%. However, when working with the validation set, the performance of algorithm has dropped continuously. In figure 10(B), we can see the performance of all the feature based technique has been decreased significantly.

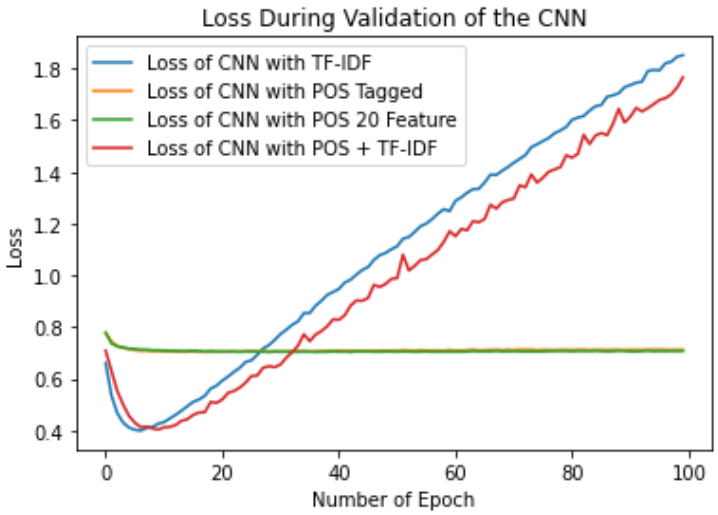

Figure 10. (D) Validation Loss

In addition, we have also computed the loss for training and validation. The loss function is used to optimize a machine learning algorithm. The loss is calculated on the training set and validation set and its interpretation is based on how well the model is doing in these two sets. It is the sum of errors made for each sample in training or test sets. Loss value implies how poorly a model behaves after each round of optimization. A loss is a number indicating how bad the 'model's prediction was on a sample. If the 'model's prediction is perfect, the loss is zero; otherwise, the loss is greater. The goal of a model is to find the set of weights and biases that have low loss. Higher loss is worse for any model. A loss is not a percentage. It is a sum of the errors made for each sample in training or validation.

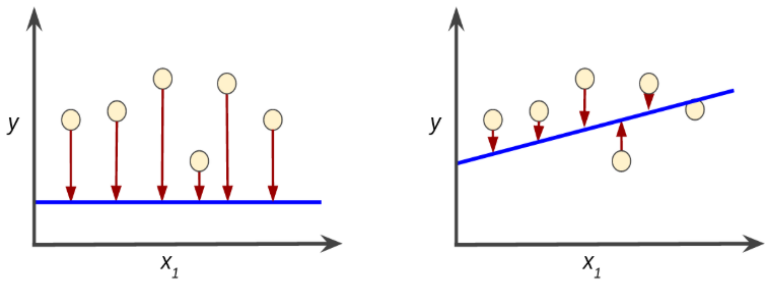

Figure 11. shows the loss measuring example

Figure 11 shows two graphs demonstrating the losses of two different models, the left graph has a high loss and the right graph has a low loss.

- The arrows represent a loss.
- The blue lines represent predictions.

The loss of the proposed deep learning model has demonstrated in figure 10(C) and 10(D). The figure 10(C) shows the training loss and 10(D) shows the validation loss of the model. In the diagram 10(C), we found that the loss of training reduced and become consistent. On the other hand, 10(D) demonstrates the loss during the validation of the model. The performance indicates when number of epoch has increases the loss of validation has also been increases in the case of TF-IDF based features learning, as well as also for the combined features, but the for combined features the loss function is less incline.
On the other hand for POS tagged data we can see the model produces similar or consistent performance in all the cases. Finally, for more simple understanding we have also measured the classification report.

Table 3. Performance summary of the models

	POS with 20 feature			POS Tagged			TF-IDF			Combined (TF-IDF + POS)		
	Precision	Recall	F1-sscore	Precision	Recall	F1-sscore	Precision	Recall	F1-sscore	Precision	Recall	F1-sscore
Class 0	0. 0	0.0	0.0	0.0	0.0	0.0	0.25	0.31	0.28	0.23	0.29	0.26
Class 1	0.78	0.99	0.87	0.77	0.99	0.87	0.90	0.89	0.89	0.90	0.89	0.90
Class 2	0.51	0.06	0.51	0.42	0.05	0.09	0.70	0.70	0.70	0.74	0.72	0.73
Accuracy	0.77			0.77			0.82			0.83		
Marco avg	0.43	0.35	0.43	0.40	0.35	0.32	0.62	0.63	0.62	0.62	0.63	0.63
Weighted avg	0.69	0.77	0.69	0.67	0.77	0.68	0.83	0.82	0.83	0.84	0. 83	0.83

The classification report for the implemented techniques has been given in table 3.The class-wise performance analysis is demonstrated in the table 3. That demonstrates the precision, recall and F1-score. Additionally, shows the accuracy, Marco avg, and weighted avg. according to these values we can say the combined and TF-IDF based features are providing higher accurate results. The accuracy has also visualized in figure 12. According to the given results of the implemented techniques, we found that the accuracy of POS based both the techniques i.e. entire features and top 20 features based approach demonstrate the similar accuracy of 77%. On the other hand the TF-IDF and combined features based technique provide the 82% and 83% of accuracy. Thus we can say the combined features based classification approach provide higher accurate results.

5. Conclusions

Social media is one of the user-centric and user-generated data source. The data may have a significant amount of knowledge, challenges and opportunity for diverse areas of

applications. However, there are a number of opportunities hidden in the social media data, but toxic intension based data contains can create the biggest issues in healthy social media surroundings. In this context, a DGM model is analysed, which is limited to working with a small amount of data. Therefore, in this paper, a deep learning Model is proposed for classifying hate speech, offensive language and the normal social media posts. The finding says the deep learning techniques are better for classifying large datasets.

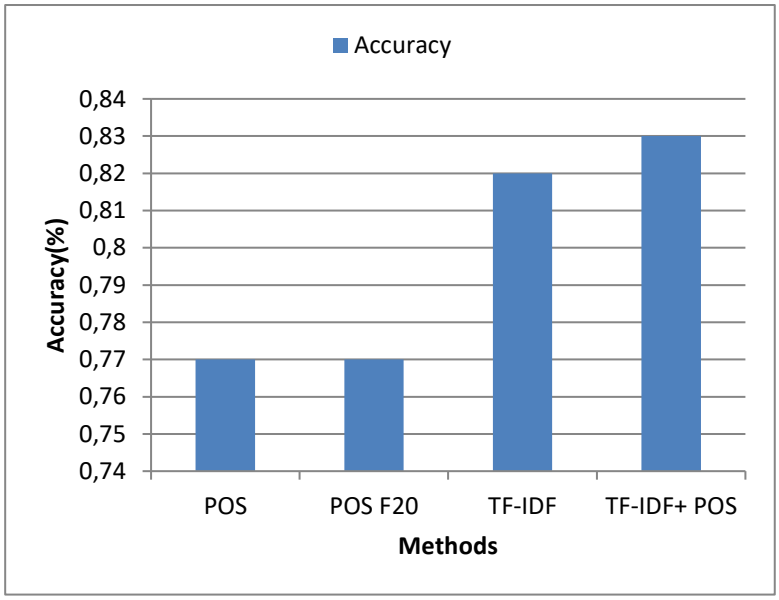

Figure 12. Comparison of Mean Accuracy

The DGM model utilizes the classical techniques for accurate classification of text but a limitation of the model motivates us to work with deep learning models. The proposed model includes the text feature analysis techniques and CNN algorithm for providing an accurate text classification technique. This model aims to differentiate between hate speech, offensive text and normal text by only considering text in social media post. During the experiments we conclude the following observations:

1. DGM is accurate but is limited to work with the less dimensional data on the other hand the deep learning is effective for learning with large scale data
2. Deep learning technique provides relevant or higher accuracy in less amount of time which is proved in the obtained performance.
3. In classification of offensive text and hate speech text, the key words are more essential then the sentence structures or sentence formation.
4. POS tagger is less effective for representing the text features to model difference between offensive and hate speech text.
5. The combination of keyword based features and POS tagged data can improve the performance significantly.

This study is helpful as a tutorial to demonstrate how the classical machine learning techniques are accurate but can be expensive to solve real-world problems. On the other hand, this study also tried to demonstrate how we can make use the deep learning techniques for processing the text data and get higher classification accuracy. In the near future, the following directions can be helpful for the extension of this study:

1. The deep learning techniques involve more complex data models, which will provide more accurate data analysis. Among them the LSTM has become much more popular. For extension, we can use them in future implementations.
2. The combination of two different kinds of features increases the dimension of the data to be trained. At the same time, in the selected TF-IDF features, various non-meaningful keywords are available. Therefore, the need is to design an enhanced semantic keyword selection technique for feature representation.

References

[1] H. Wang, Q. Zhang, & J. Yuan, *"Semantically Enhanced Medical Information Retrieval System: A Tensor Factorization Based Approach"*, IEEE, 2169-3536, 2017.

[2] S. Bergamaschi, E. Domnor, F. Guerra, M. Orsini, R. T. Lado, Y. Velegrakis, *"Keymantic: Semantic Keyword-based Searching in Data Integration Systems"*, Proceedings of the VLDB Endowment, ACM, Vol. 3, No. 2, 2010.

[3] M. Chahal, "Information Retrieval using Jaccard Similarity Coefficient", International Journal of Computer Trends and Technology, Volume 36, Number 3, June 2016.

[4] N. Bhandari, R. Navlakhe, G. L. Prajapati, "Semantic Query Optimization based Information Retrieval Technique", The Journal of Oriental Research Madras, ISSN:0022-3301,Vol. XCII-LXXVII, 2021

[5] P. Bafna, D. Pramod, A. Vaidya, "Document Clustering: TF-IDF approach", International Conference on Electrical, Electronics, and Optimization Techniques, IEEE, 2016.

[6] R. K. Roul, J. K. Sahoo, K. Arora, "Modified TF-IDF Term Weighting Strategies for Text Categorization", IEEE, 2017.

[7] J. Chen, X. Tang, "Ensemble Of Multiple K-NN Classifiers For Societal Risk classification", J Syst Sci Syst Eng, 26(4): 433-447, Aug 2017.

[8] H. Xu, W. Yang, J. Wang, "Hierarchical emotion classification and emotion component analysis on chinese micro-blog posts", Expert Systems With Applications, 42, 8745–8752, 2015.

[9] C. Pasquier, C. da Costa Pereira, A. G. B. Tettamanzi, "Extending a Fuzzy Polarity Propagation Method for Multi-Domain Sentiment Analysis with Word Embedding and POS Tagging", ECAI, The IOS Press, 2020.