CE807 – Assignment 2 - Final Practical Text Analytics and Report

Student id: 2200735

Abstract

This study focuses on training a model for offensive tweet classification using two suitable methods: Support Vector Machines and Convolutional Neural Networks with keras library. Additionally, the study explores the use of glove pre-trained vectors as a separate method to understand the data behavior of existing tweets. The decision to use these methods is based on requirements and prior experience gained from the existing text classification machine learning algorithms. Before going ahead and implementing the model, the data is cleaned and performance metrics are defined. The selected methods behaved differently with the data when trained and tested.

1 Materials

The resources of the assignment can be referred from the following links:

- Code
- Google Drive Folder
- Presentation
- Zoom Recording

2 Model Selection (Task 1)

Model selection in text classification tasks depends on various factors like the size of the data, dimensionality of the feature space and the type of classification task at hand. For the given task, there are mainly two algorithms referred for model building mainly Support Vector Machines and Convolutional Neural Networks. As an extension to Convolutional Neural Network deep learning algorithm, a third model is also created using an **GloVe** (Global Vectors for Word Representation) to train on existing twitter tweets word vectors. (Pennington et al., 2014)

- **Model 1**: Support vector machines (SVMs) are really good at handling problems where there are a lot of different features to consider, especially when it comes to text classification. This is because in text, each word can be a feature, and there are usually a lot of words to consider. SVMs are really good at handling all of these different features and making sense of them.

- **Model 2**: A deep learning technique called Convolutional Neural Network (CNN), which is specifically designed for text classification tasks. The model architecture is made up of several layers, namely an embedding layer, a convolutional layer, a global max pooling layer, and a dense layer. Before training the model, it is preprocessed through a process known as tokenisation and padding. During training, the model uses binary cross-entropy loss to evaluate its performance, while the Adam optimiser is used to optimise the model parameters.

- **Model 3**: To further check the model's performance, pre-trained word embeddings are implemented in the **Model 2** using set of pre-trained word vectors called Global Vectors for Word Representation (GloVe), that were trained on twitter tweets data. These pre-trained embeddings can be used as initial weights for the embedding layer in the model, instead of randomly initialising them. By using pre-trained word vectors, the model can make use of the semantic relationships between words and improve its ability to understand the tweets. Furthermore, using pre-trained word embeddings can help reduce overfitting and improve the model's generalisation ability.

2.1 Summary of 3 selected Models

The selected models are among the other popular text classifican models. (Manek et al., 2015) describes that a SVM supervised learning algorithm constructs a model that distinguishes between samples belonging to one class or the other. The SVM model represents samples as points in space and maps them in a way that maximises the separation between samples of different classes, creating a wider gap between them. As a result, when new samples are introduced and mapped into the same space, they are classified based on which side of the gap they fall on. The CNN model is a neural network text classification model and CNNs are particularly suitable for tasks that involve processing sequences of text, because they can effectively capture local patterns and relationships between close words.

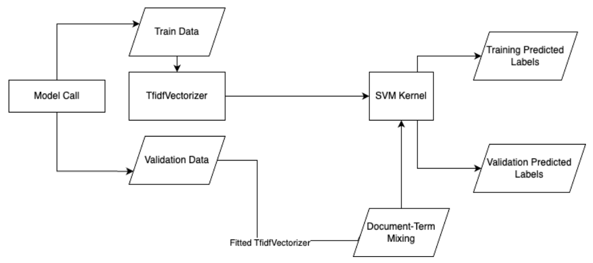
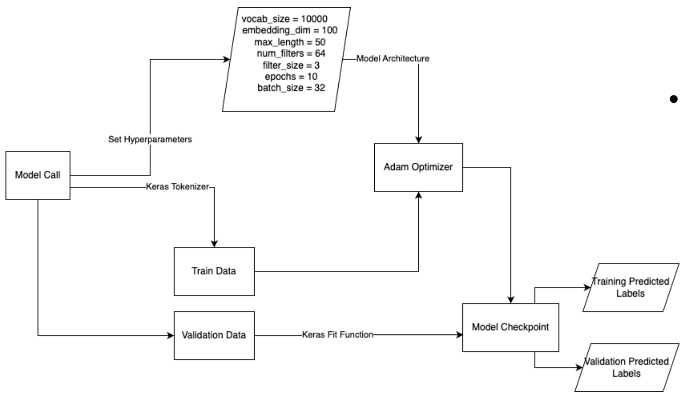


Figure 1: Understanding SVM Approach



```
vocab_size = 10000
embedding_dim = 100
max_length = 50
num_filters = 64
filter_size = 3
epochs = 10
batch_size = 32
```

Figure 2: Understanding CNN Approach

2.2 Critical discussion and justification of model selection

Each model has its own advantages and disadvantages. The selection of the best model depends on the particular task that needs to be accomplished. We will now talk about the strong points and limitations of each model and explain why one may be chosen over the other.

- To use **Model 1** as shown in Figure 1, the first step is to transform each message into a numerical feature vector. TF-IDF is commonly used, more recent work suggests that the "bag of words" technique worked better (Souza and Costa-Abreu, 2020). Offensive and normal text examples were used to create a table with scores assigned to each phase for offensive and normal language. This resulted in two features for each phase indicating the degree to which it belonged to offensive and normal language sets. The code is easy to understand and implement, making it a good choice for beginners in machine learning. The TfidfVectorizer is a useful tool for text feature extraction that can handle various text formats. The linear SVM model is generally fast and effective for text classification tasks, particularly for binary classification problems. The code includes a function to save the trained model and vectorizer for later use, which can save time and effort in future tasks. The TfidfVectorizer does not capture semantic relationships between words, which may result in suboptimal performance in tasks where meaning matters. The linear SVM model assumes that the data is linearly separable, which may not always be the case. This could result in underfitting or overfitting of the model, reducing its accuracy.

- Convolutional Neural Networks (CNNs) classify tweets as shown in Figure 2, which is a technique that can capture important features of text data. The use of the Keras library and its high-level API makes it easier to define and train the neural network model. The use of the **ModelCheckpoint** callback allows the best performing model to be saved automatically during training, which saves time and effort. CNNs may not perform well if the dataset is too small because it typically require large amounts of data to train effectively.

- An extension to check improvement in the **Model 2** using the embedded file of **GloVe**. The model uses pre-trained word embeddings, which can improve the performance of the model. Hence it can help tackling the situation if small dataset is being trained using the Model 2.

3 Design and implementation of Classifiers (Task 2)

Before understanding each model parameters in detail, the tweet data is cleaned based on various factors which include duplicate rows, null rows, unwanted regex such as @user or emoticons and other special characters. After that, stopword and lemmatizer is used on the 'tweet' feature, along with converting the 'label' column to binary data type.

- The Table 1 show the rows present in the train, valid and test datasets. The features include 'id', 'tweet' and 'label'.
- The hyperparameters used in the Model 1 are discussed as follows:
  1. TfidfVectorizer(): By default, the TfidfVectorizer uses the L2 normalisation and the ngram_range parameter is set to (1, 1), which means it only considers unigrams. The max_df parameter is set to 1.0, which means it considers all the words in the documents. The min_df parameter is set to 1, which means it only includes words that appear at least once in the document. The default settings of TfidfVectorizer are a good starting point, but they can be tuned to improve the performance of the model.
  2. svm.SVC(kernel='linear'): The SVC (Support Vector Classifier) model is used with a linear kernel. This means that the decision boundary between the two classes will be a straight line in the feature space.
- The hyperparameters used in the Model 1 are discussed as follows:
  1. vocab_size: This hyperparameter determines the maximum number of words to keep in the tokenizer's vocabulary based on word frequency. The default value used here is 10,000, meaning that only the 10,000 most frequent words in the training data will be kept.
  2. max_length: This hyperparameter determines the maximum length of the input sequence, and any sequences longer than this value will be truncated. The default value used here is 50.
  3. num_filters: This hyperparameter determines the number of filters used in the convolutional layer of the model. The default value used here is 64.
  4. filter_size: This hyperparameter determines the size of the filters used in the convolutional layer. The default value used here is 3.
  5. epochs: This hyperparameter determines the number of training epochs to run. The default value used here is 10.
  6. batch_size: This hyperparameter determines the number of samples used in each mini-batch during training. The default value used here is 32.

Dataset	Total	% OFF	% NOT
Train	12313	33.23	66.76
Valid	927	33.2	66.8
Test	860	27.9	72.1

Table 1: Dataset Details

Model	%F1 Score
Model 1	78
Model 2	49.25
Model 3	48.5

Table 2: Model Performance

4 Data Size Effect (Task 3)

The size of the dataset has a significant impact on model training, as the performance of the model is highly dependent on the dataset size. To ensure that the class distribution is preserved while splitting the data, the train-test split method is used with the stratify parameter. The train dataset is split into four parts, comprising 25%, 50%, 75%, and 100%.

- The Table 3 indicates the subset division of the training data used for the model building.
- The hyperparameter for **Model 3** which is the vocab_size is changed to 50000 to include the words from the embedded file in the model for training.
- Model 1 Performance: The average accuracy across the four runs is approximately 80.07%. This means that on average, the SVM model correctly classified 80.07% of the samples in

the test set.The validation accuracy scores for the same SVM model. These scores were obtained during the training phase of the model using a separate validation set. The validation accuracy scores range between approximately 73.89% and 76.81%, which is slightly lower than the average test accuracy. Overall, these accuracy scores indicate as shown in Figure 3 that the SVM model is performing reasonably well on the classification task, but there may be some overfitting to the training data as evidenced by the higher test accuracy compared to the validation accuracy.

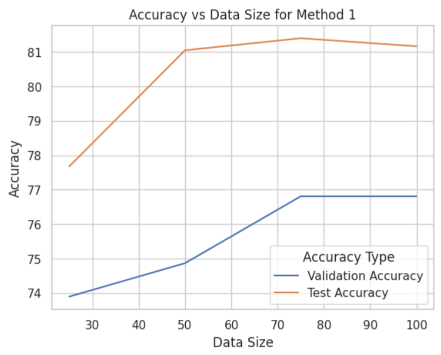


Figure 3: Model 1 Accuracy and Data Size

- Model 2 Performance: The test accuracy shows the test accuracy of the CNN model for four different runs. The values range from approximately 56.63% to 60.00%. The validation accuracy shows the validation accuracy of the CNN model for the same four runs. The values range as shown in Figure 4 from approximately 70.77% to 74.22%. Overall, the test accuracy scores for the CNN model are lower than those of the SVM model, but the validation accuracy scores are higher, which suggests that the CNN model may generalise better to new data than the SVM model.

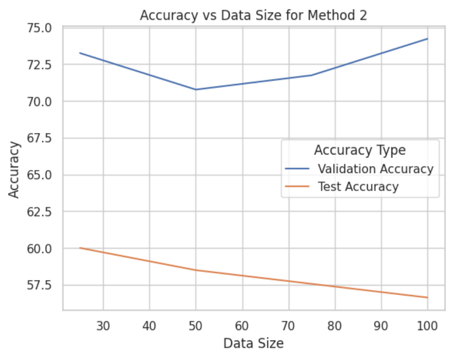


Figure 4: Model 2 Accuracy and Data Size

- Model 3 Performance: As seen in the Figure 5, the test accuracy shows the accuracy of the model on the test set for each epoch during training. The accuracy ranges from 55.9% to 59.6%, which is relatively low, indicating that the model might not be performing well on the test set. The validation accuracy, on the other hand, shows the accuracy of the model on the validation set for each epoch during training. The validation accuracy ranges from 72.7% to 73.6%, which is slightly better than the test accuracy but still relatively low.

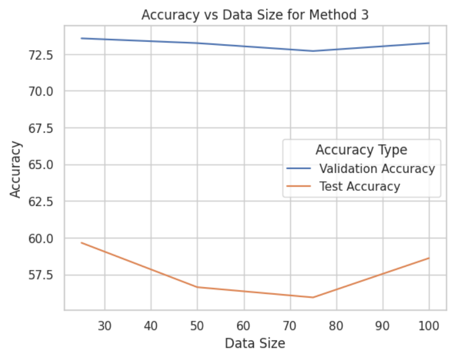


Figure 5: Model 3 Accuracy and Data Size

- Table 5 and Table 6 provide evidence that the performance accuracy of any model is significantly influenced by the size of the dataset, regardless of how strong the model is. Additionally, the test file revealed a few words that occurred so infrequently that the model was predicting the wrong class for tweets containing these words, as the model was not able to adequately learn their significance due to their limited presence in the data.

Data %	Total	% OFF	% NOT
25%	3078	33.23	66.76
50%	6156	33.23	66.76
75%	9234	33.23	66.76
100%	12313	33.23	66.76

Table 3: Train Dataset Statistics of Different Size

5 Summary (Task 4)

The objective of this report is to identify models that excel in text classification tasks. The quality of the dataset played a significant role in the models' performance. Despite the data cleaning and pre-processing, some tweets remained redundant,

Example %	GT	M1(100%)	M2(100%)	M3(100%)
nigga ware da hit	NOT	NOT	NOT	OFF
bitch url	NOT	NOT	OFF	NOT
liar like rest url	NOT	NOT	OFF	OFF
take nigga name bio url	NOT	NOT	OFF	OFF
get feeling kissing behind humiliate later	OFF	OFF	OFF	OFF

Table 4: Comparing two Model's using 100% data: Sample Examples and model output using Model 1 & 2. GT (Ground Truth) is provided in the test.csv file.

Example %	GT	M1(25%)	M1(50%)	M1(75%)	M1(100%)
school shooting arent controversial wanting gun control	OFF	OFF	OFF	OFF	OFF
party low taxation url	OFF	OFF	OFF	OFF	OFF
always smack url	OFF	OFF	OFF	OFF	OFF
please ban cheating scum literally invisible url	NOT	OFF	NOT	NOT	NOT
know u aint going work tomorrow url	OFF	OFF	OFF	OFF	OFF

Table 5: Comparing Model Size: Sample Examples and model output using Model 1 with different Data Size

Example %	GT	M2(25%)	M2(50%)	M2(75%)	M2(100%)
pic awesome	NOT	OFF	OFF	OFF	OFF
k bitch	NOT	OFF	OFF	NOT	OFF
everything liberal touch turn absolute shit	NOT	OFF	OFF	OFF	OFF
fucking year url	NOT	OFF	OFF	NOT	OFF
many cum fan	NOT	OFF	OFF	OFF	OFF

Table 6: Comparing Model Size: Sample Examples and model output using Model 2 with different Data Size

having only one word or mismatched ground truth labels, resulting in inaccurate predictions. However, even after these issues were addressed, the models achieved an accuracy of around 75%, which is considered commendable given the quality of the data provided.

5.1 Discussion of work carried out

Based on the provided test and validation scores, it appears that the model based on SVM has the highest validation accuracy (76.8%), followed by the CNN model with embedded Glove word vector file (73.2%), and then the CNN model with no word embeddings (71.7%). However, it's important to note that other factors such as training time and computational resources required may also need to be considered when determining the best model for a specific use case.

5.2 Lessons Learned

There are few important things that could be understood from the models built and are listed as follows:

1. SVM is a linear model that works well when the data is linearly separable, but it may not be as effective when dealing with complex, non-linear data.
2. CNN, on the other hand, is a neural network model that is more capable of capturing complex, non-linear relationships between features in the data.
3. Pre-processing the data is crucial for improving the performance of the models, including cleaning the data, removing noise, and converting text to numerical features.
4. Training the models on larger datasets and exploring other models, such as recurrent neural networks and transformers, may further improve the performance of text classification tasks.

References

Asha S Manek, P Deepa Shenoy, M Chandra Mohan, and Venugopal K R. 2015. Aspect term extraction for sentiment analysis in large movie reviews using gini index feature selection method and svm classifier.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation.

Gabriel Araújo De Souza and Márjory Da Costa-Abreu. 2020. Automatic offensive language detection from twitter data using machine learning and feature selection of metadata.