Abstract

Online Social Networks (OSN) have become an important source for the exchange of the data and knowledge around the world. Twitter is one of the social networking platform which is widely used across the globe for this. But as the popularity grows, it evolves as the prime target for the spammers to spread fake news and attack the user's personal information. Spammers obtain Internet users' attention by repeatedly disseminating unsolicited content on social networking sites. Their conduct has made uncomfortable to true Internet users. In order to identify and remove social media spammers, spam detection techniques are necessary. The goal of spam detection is to classify reviews into spam or non-spam. Past studies have shown that the success of spam detection techniques was built by numerous types of machine learning and deep learning methods. The deep learning models such as LSTM, CNN, BiLSTM and Transformer are popularly used for spam detection on Twitter dataset. On the dataset, strategic text processing techniques were applied to produce refined dataset for the experiment. We investigated a number of word embedding methods, including the Word2Vec model, pre-trained GloVe vectors, random normal weight initialization, random Glorot initialization, and random uniform weight initialization. Finally, the classification results of LSTM, CNN, BiLSTM and Transformer were evaluated using standard performance measures. According to experimental result, the best spam accuracy and precision scores were attained by LSTM with uniform random embedding weight initialization, scoring 80% and 79%, respectively. The model has assigned the embedding weights to all vocabulary words, including unknown words for the tweets. As a result, this will cause better generalization of model to new test data.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Through the use of virtual organisations and networks, online social networking platforms are a digital invention that promotes the exchange of ideas, opinions, and data. The social media networks have a quick access to users to discuss on latest topics. Social media platforms emphasise on collaboration, sharing of content, localised information, and correspondence. Twitter [5] is one of the popular social networking platform for these purposes. Without a question, social media platforms have become a part into our daily life. Cybercriminals use social media platforms to disseminate spam, misleading facts, fake news, and malicious links. The more time spent on social media can lead to cyberbullying, social anxiety, depression, and exposure to content that is not age appropriate.
Spam is best defined as unnecessary or unwanted messages delivered over the internet. These are typically sent to a huge number of users for variety of scenarios, such as advertising, phishing, distributing malware, and so forth. Spammers [6] from all around the world are from spamming organisations and are multiplying the amount of spam to genuine users. Spam on social media platforms, which originates from spam accounts and postings, creates unnecessary noise that drowns out genuine commitments and information. However, a great quantity active user and the convenient conditions for publishing content have attracted a lot of spammers on OSNs. The release of spam by spammers has caused great troubles to normal users and platforms.
The high traffic volume of twitter spam message was close to 10 million spam accounts reported in 2018 [7]. Given that only a small proportion of people create the majority

of Twitter posts, Twitter is an extremely tweeting organisation. The popularity of Twitter attracts spammers, which has led to an increase in spam. Spamming on social media is clearly more effective than spamming via email or other more standard techniques. Currently, fake reviews or spam are growing and becoming a significant problem. According to studies, 15% of Twitter users are automated, and one in every 20 tweets is often spam [8].

Even though social media platforms are useful for accelerating the spread of news and enabling users to discuss topics and notify their status, they also present new opportunities for spam. As an instance, the latest trends, which are the subjects that are most popular on Twitter at any given moment, have been viewed as a way to drive customers and make money. Many individuals tweet about important events, which causes them to move to the top of trending topics quickly. These issues find up on the radar of spammers, who tweet about them using the usual buzzwords, but with URLs that take viewers to entirely different websites. It is challenging for users to determine the URL information without opening the webpage because tweets are frequently made with abbreviated URLs.

Spammers swiftly spread fake information and unwanted content to several Twitter accounts. Trending words are used in this spam content to draw the interest of viewers. Additionally, shorter URLs are used to direct people to harmful websites that could infect their computers with malware [9]. If these adverse impacts are not addressed seriously, crime rates may spike dramatically. To address this issue, an effective spam detection system is needed.

1.1 Objective and Scope

The goal is to use deep learning models in order to perform spam detection on tweets. In addition, several text processing methods are investigated in order to improve the accuracy of deep learning models for spam detection.

The goal of this thesis is to analyse how well deep learning models perform at spam detection using the public Twitter dataset. Non-English tweets are removed through a data cleaning procedure. The next phase in the data pre-processing process is to extract word features and turn words into dense vectors for computational purposes. Different

word embeddings techniques will be studied and experimented. To perform spam categorization on tweets, various deep learning models including CNN [10], LSTM [11], BiLSTM [3], and Transformer [12] are explored. Deep learning models are hyper-parameter tuned to enhance classification outputs. Deep learning model classification performance are contrasted with the latest machine learning algorithms like SVM [13], RF [14] classifier, and other relevant works. Various text feature processing methods will be investigated on deep learning models.

Contributions of the Proposed Work. The following are the contributions of the proposed work:

- Firstly, we pre-processed the tweets from the dataset by applying several techniques, i.e., tokenization, stemming, lemmatization, stop-words removal. After that, we applied several word-embeddings techniques (Word2Vec model, GloVe vectors, random normal weight initialization, Glorot initialization, and random uniform weight initialization) to get the feature vectors.
- These word embedding vectors are applied on different deep learning models (LSTM, CNN, BiLSTM and Transformer) which can fully consider the context and semantics to capture the core of tweets text.
- In the proposed work, the LSTM attained a classification accuracy of 78.8%. The tweets' classification performance for the Transformer and biLSTM models was 77.9% and 77.1%, respectively.

The rest of the content is organized as follows. The next chapter covers past works related to machine learning and deep learning techniques on spam detection. Chapter 3 covers the step-by-step procedure from data pre-processing to selection of word embedding techniques. Chapter 4 describes the different types of deep learning models parameter, and the evaluation metrics used in experiment. Additionally, dataset, and experimental results are presented in figures and tables format for ease of visualization. Chapter 5 offers conclusions and directions for future work.

Chapter 2

Background and Literature Review

This chapter covers the background work required to convert the text into the format required by the embedding layers to extract the feature vectors. We also reviewed the research work done in the past by the researchers.

2.1 Text-Processing Techniques

In this section, we discussed about the pre-processing techniques required to convert raw tweet data into the required input format and word embeddings.

2.1.1 Stemming & Lemmatization

There are numerous grammatical forms for words, including noun, verb, singular, plural, and various tenses. Machines cannot comprehend words with multiple variations, unlike humans. Machine learning and prediction may become ambiguous as a result of these variances. In addition, a lot of computer power will probably be needed to handle the enormous variety of words in languages. To reduce word complexity, computational time, and space, text normalisation is a crucial step in machine learning applications. Two often used text normalisation techniques in NLP tasks to change a term to its base form are stemming and lemmatization. The NLTK [15] package can be used to implement these two techniques.
Stemming is a method for eliminating affixes from words to reveal their basic form. It is

analogous to trimming a tree's branches back to the trunk. For instance, the word *eat* is the root of the verbs *eating*, *eats*, and *eaten*. Stemming is used by search engines to index the words. Because of this, a search engine can only record the stems of a word rather than all of its variations. Stemming does this by reducing the overall dimension of the index and improving retrieval precision. Stemming is a technique used to extract the base form of the words by removing affixes from them. It is just like cutting down the branches of a tree to its stems. For example, the stem of the words *eating*, *eats*, *eaten* is *eat*. Search engines use stemming for indexing the words. That's why rather than storing all forms of a word, a search engine can store only the stems. In this way, stemming reduces the size of the index and increases retrieval accuracy.

Lemmatization, as opposed to stemming, accurately distils a word to its dictionary-found root form. The root word in lemmatization is referred to as a lemma, which is a group of words in their dictionary form. Examples of the word carry variations include carries, carried, and carrying. Therefore, the lemma of those words is carry. Lemmatization uses Parts-of-Speech (POS) tagging to give words context, which results in excellent word conversion accuracy. Based on the POS tag supplied, the word lemmatizer changes a word to its root form. Lemmatization's standard POS tag is noun. Lemmatization, while its excellent accuracy, takes longer to complete since it needs to search the WordNet [16] corpus for the right root word before converting the term.

2.2 Word Embedding

Word embeddings are a crucial machine learning approach for converting words to dense vectors of real numbers. These word vectors were created using a text corpus for word embedding model training. The model studies the text corpus and uses cosine similarity to map semantically related words close to one another on the embedding space. As a result, words with similar semantic relationships have nearly identical vectors.

Word2Vec used a two-layered neural network to generate a set of vectors after training on a huge text corpus. Word2vec uses the skip-gram and CBOW algorithms to gather word context data. These algorithms are not ideal because the semantics of every given word are determined by the words around it within a predetermined window size. The

key distinction between CBOW and skip-gram is that while skip-gram utilises a word to forecast target context, CBOW uses context to anticipate a target word.
The Word2Vec model was used to train the pre-trained dense vector known as GloVe [17]. The GloVe-Twitter vector, which has been trained on 2 billion tweets and 1.2 million vocabulary terms, is specifically employed in this research. The co-occurrence matrix, which calculates how frequently words occur alongside one another in a corpus, was used to create the GloVe vector. This calculation has the ability to encode some kind of word context information. For instance, *ice* and *solids* are more likely to occur together than *ice* and *gas*.
Random embedding weight initialization in the embedding layer of the neural network is another approach to word embedding. This approach learns word vector representations within the same network rather than a separate network, which is the key distinction between it and Word2Vec. The back-propagation approach is used to update the embedding weights during model training once they are randomly initialised using a uniform distribution [18].

2.3 Literature Review

In early implementation of spam detection model, statistical data such as user-based and content-based features were extracted from Twitter user profiles and tweets, respectively. Examples of user-based features are number of followings, number of followers, length of username, number of tweets posted etc. and content-based features are number of words, number of URLs, number of mentions (@username) etc. Traditional machine learning methods such as Naive Bayes (NB) [19], K-Nearest Neighbor (KNN) [20], Support Vector Machine (SVM), Decision Tree (DT) [21], Logistic Regression (LR) [22] and Random Forest (RF) were experimented on these features to create Twitter spam detection model. Benevenuto *et. al* [23] utilized Twitter API to crawl tweets which contain trending hashtags back in year 2009. Tweets were collected and manually labelled into spam and non-spam to create a labelled dataset for machine learning. SVM, a supervised machine learning model, was used in the experiment to learn user-based and content-based features. Wang *et. al* [6] employed several machine learning methods to learn user-based,

content-based, word n-grams, and sentiment features. The combination of user-based and content-based features achieved best f-score using RF classifier. However, RF classifier did not achieve good f-score value on word n-grams (text-based) feature alone. Ashour *et. al* [24] employed n-grams models to create character n-grams feature. N-gram models use a fixed window size to extract local context of a character based on its surrounding characters. Several machine learning methods such as SVM, LR, and RF classifiers were experimented to learn n-grams features. The best method was achieved by LR classifier, which has a slight improvement in f-score compared to [6].

In recent years, deep learning has gained popularity in spam detection. Deep learning models incorporate word embedding techniques to convert words into numerical vectors based on semantic word similarities [25]. Most commonly used word embedding techniques are Word2Vec [25] model and pre-trained word vectors, GloVe [17]. Deep learning method enables models to learn text-based feature in the form of word vectors and raise their effectiveness in spam detection.

Ban *et al.* [9] employed Word2Vec [25], and BiLSTM network to extract features. BiLSTM consists of two LSTMs to learn past and future context of each sequence data (tweets). Word2Vec model was used to convert textual tweets into high dimensional word vectors called Word2Vec features. Likewise, BiLSTM network was used to learn tokenized textual tweets and create deep-learnt features. Spam detection results based on the two feature sets were compared using state-of-the-art machine learning algorithms such as NB, KNN, SVM, DT, and RF. Similarly, Martinelli *et al.* [26] employed Word2Vec model by using CBOW [25] approach to learn efficient word embedding representation. On the contrary to [9], word vectors were loaded to Multilayer Perceptron (MLP) [27] classifier to perform text classification. The best result was obtained by MLP with 3 hidden layers. On the contrary, Wei et al. [28] adopted GloVe to map vocabulary words found in the dataset to higher dimensional matrix (up to 300d) using its pre-trained word vectors. Instead of using random weight initialisation, this high dimensional matrix was assigned as weights in the embedding layer of BiLSTM network.

The combination of user-based, content-based, and text-based features were also experimented in deep learning models. Founta *et al.* [29] employed two neural networks to learn metadata (user-based and content-based features) and text data. Metadata classifier

comprises multiple dense layers, whereas text classifier used RNN with attention layer to focus on specific parts of the text data. GloVe [17] word vectors were used as pre-trained embedding weights in the neural network to reduce model training time. The outputs of these two networks were merged and passed to classification layer.

Similar to [29], Alom *et al.* [30] implemented a dual network to learn user-based, content-based, and text-based features. Text-based classifier was built by Word2Vec algorithm to transform tweets into higher dimensional vectors. This was followed by CNN model to learn the high dimensional matrix. Meta-data classifier utilized user-based and content-based features as inputs to second network. These features were normalized before passing to a five-layered dense network. The outputs of these two classifiers were merged and passed to classification layer.

Based on the mentioned related works, it can be inferred that pre-trained word embeddings paired with CNN and LSTM networks has strong effectiveness in spam detection. Apart from Twitter spam detection, these approaches were experimented in other applications such as sentiment analysis, sentence classification, and email spam detection.

Archchitha *et al.* [31] employed a CNN model with three parallel convolution layers with different filter sizes for opinion spam detection in online reviews. Since online reviews are usually longer than tweets, review text sequences were truncated to a maximum length of 1000 words. Word tokens were mapped to high dimensional vectors using pre-trained GloVe word vectors before passing to CNN model as inputs. Wang *et al.* [32] adopted a method which uses Word2Vec skip-gram model and LSTM to create a sentiment classification model. In the experiment, LSTM outperformed NB and ELM [33] in short text classification. In addition, classification performance of LSTM improves with the increase in training data size.

Kumar *et al.* [34] employed LSTM network for spam message detection. Instead of using pre-trained word embedding models, this approach used random weight initialization in the embedding layer to learn the word tokens. Those weights were updated through back propagation during model training. A summary of related works is tabulated in Table 1.

Related Works	Features			Learning Methods		Word Embedding
	TF	UF	CF	ML	DL	
[6], [23]	x	✓	✓	✓	x	Word2Vec
[24]	✓	x	x	✓	x	Word2Vec
[9]	✓	x	x	✓	✓	Word2Vec
[26], [28]	✓	x	x	x	x	GloVe
[30], [29]	✓	✓	✓	x	✓	GloVe
[31], [32], [34]	✓	x	x	x	✓	Random Weight Initialization

Table 1: Summary of Related Works. TF: Text-based Feature, UF: User-based Feature, CF: Content-based Feature, ML: Machine Learning, DL: Deep Learning.

Chapter 3

Proposed Work

In the proposed work, various deep learning models for text classification, including CNN, LSTM, biLSTM and Transformer were tested. Steps for data pre-processing, such as feature extraction and data cleaning, were completed before model training. The *"content_polluters_tweets.txt"* and *"legitimate_users_tweets.txt"* files contained information that was relevant and significant. Data cleaning to eliminate outliers was done after feature extraction. Regular expressions (regex) method in NLTK packages were used to do text processing to remove unnecessary stuff from tweets. Using pre-trained word vectors from the Word2Vec model and GloVe, processed tweets were normalised to equal length and mapped to vector representation. The final step was to feed the normalised training dataset into the machine learning text classifiers. The accuracy, precision, recall, and f-score metrics were used for evaluating model performance using test data, or tweets that were not used in training. Figure 1 depicts the proposed model architecture.
Deep learning networks allow the learning of the deep structure of data. A deep learning-based model for spam detection was built in this work by combining Word Embedding techniques and a deep learning model. Input tweets were represented as vectors in the feature space, and the deep learning models was trained to learn features of document vectors to maximize its prediction accuracy.

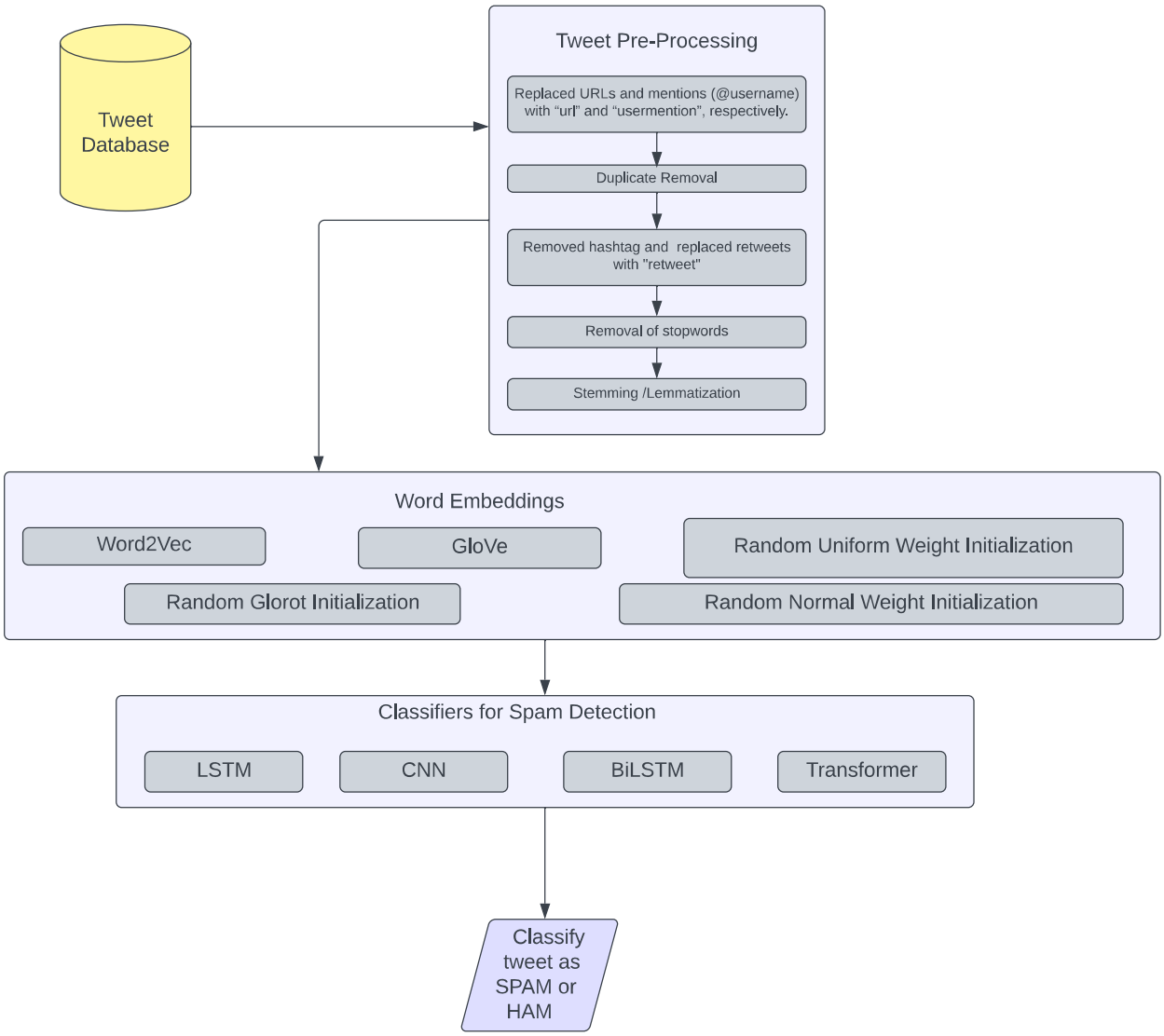

Figure 1: The proposed architecture for spam detection.

]Data pre-processing techniques that result in a dataset of high quality for machine learning include feature extraction and data cleansing. To ensure that the model performs optimally in terms of categorization, these procedures must be strategically planned and carried out. To extract features that are helpful for spam classification, feature extraction is an important process. Table 2 displays an example of the *"content_polluters_tweets.txt"* file's data. UserID represents a Twitter users identify. A Tweet's identification is its

TweetID. The time and date when a tweet was created are shown via the CreatedAt property. Due to the fact that these features typically contain unique data, they are not useful for machine learning. All of these features were not extracted as a result. The last tweet feature was instead retrieved for machine learning.

UserID	TweetID	Tweet	CreatedAt
6542	5702145484	I need a good person tomorrow at an event in Sydney. Please email apply@jacbos.com	2020-11-13 23:17:12

Table 2: A sample data containing four features.

More than 5 million tweets in the Social Honeypot dataset have the class labels ”Spam” and ”Non-Spam”. Because there were so many tweets, the spam and non-spam datasets had to be processed individually. The Pandas [35] software was used to execute Exploratory Data Analysis (EDA) on these datasets. Figure 2 depicts an example of a tweet for spam dataframe.

	Text
0	MELBOURNE ENQUIRY: Seeking a variety of acts for our end of year show. Payment is $120 per slot or 200 for 2.... http://bit.ly/4Ah3fF
1	THE BURLESQUE BOOTCAMP SYDNEY - Open Date tickets now available from http://bbootcampsyd.eventbrite.com/ for Jan /... http://bit.ly/rCenZ
2	THE BURLESQUE BOOTCAMP SYDNEY - Open Date tickets now available from http://bbootcampsyd.eventbrite.com/ for Jan /... http://bit.ly/1v5hvb
3	THE BURLESQUE BOOTCAMP SYDNEY - Open Date tickets now available from http://bbootcampsyd.eventbrite.com/ for Jan /... http://fb.me/3rzipF0
4	Come to "The Burlesque Bootcamp - Sydney" Saturday, 23 January 2010 at 10:00 until Sunday, 24 January 2010 at... http://bit.ly/38simD

Figure 2: Tweet sample in Spam Dataframe.

The dataset contains a number of tweets that are not in English, as was found after EDA. Due to the mix of languages, this will reduce the text classifier’s performance. As a result, the original dataset needed to have its data cleaned. To normalise the tweet format using for machine learning, text processing was done. Table 3 provides a summary of the data pre-processing for the dataset.

Data Cleaning	Actions Taken
Step 1	Removed URLs, mentions (@username), hashtags (#keyword), and retweets (RT).
Step 2	Removed stopwords (ex., I, the, you), and punctuations. Lemmatized texts to group similar words as a single item.
Step 3	Removed non-English words, and words with less than 3 characters.

Table 3: Overview of Data Pre-processing for Dataset.

First, using the regex approach, features including URLs, user mentions, hashtags, and retweets were eliminated. By using the regex method *regex.sub()*, which changes a string into any replacement, these traits were eliminated. By giving "https/S+" as the first argument and "empty string" as the second input to the *regex.sub()* method, all URLs that typically begin with "https://..." are deleted. On the other hand, passing "@/S+", "#/S+", and "/bRT/b" as the first input, respectively, deleted user mentions, hashtags, and retweets. Any non-space character is what the regex phrase "/S+" refers to. In order to detect users mentions and hashtags, this phrase was added to annotations like "@" and "#" because any non-space words after the annotations are supposed to be replaced with an empty string. Regex /b stands for the word border. To prevent the word "RT" from being removed from the word "CART," this term was appended to both ends of the word "RT".
Second, stopwords were eliminated since they were frequent and had less meaning than other vocabulary words. Every word in the *nltk.corpus.stopwords* package was changed to an empty string. Using the *string.punctuation* package, punctuation marks were deleted. Words that are similar but have distinct formats, like plural as well as singular terms, are combined into a single item using word lemmatization. This aids in lowering the computation's word count. To lemmatize words, the *WordNetLemmatizer()* function from the WordNet [16] package was utilised. Lemmatization is more accurate in terms of word reduction even though it takes longer to complete than stemming.
Thirdly, as pre-trained word embeddings contain vectors for these words, non-English words (such as misspelt and truncated terms) were eliminated. The nltk.corpus.words package's missing words were substituted with empty strings. Last but not least, terms with fewer than three characters were eliminated because they are rarely used in texts.

3.1 Word Embedding Models

3.1.1 Word2Vec Model

The training dataset was trained using the Word2Vec model, which generates word vectors. First, word tokens were created for each tweet in the collection. Second, a Word2Vec model from the Gensim [36] package was used to learn the vector representations of words and construct a vocabulary from the training dataset. Two distinct lists were used to store vocabulary words and their accompanying vectors. Calling the trained Word2Vec model's wv.index2word attributes will provide a list of vocabulary words. The trained Word2Vec model's wv.syn0 attribute, on the other hand, can be used to get learned vectors.
The Pickle package was used to preserve the vocabulary word vector representations as pre-trained embedding weights. Since these weights are reusable to different deep learning models, there is no need to train them again, this cuts down on the amount of time needed for experiments. In order to speed up computation, during model training these pre-trained weights won't be modified using a back-propagation approach. Table 4 provides a summary of the CBOW training parameters.
The training dataset should be re-tokenized excluding the $< OOV >$ token for unknown terms when the Word2Vec model is used. The Word2Vec model's inability to generate embedding weight for unidentified words outside of the training dataset is the cause. Finally, the Pickle package was used to save the trained tokenizer for the Word2Vec model.

Input	Training Dataset
Vector Dimension	50
Minimum Count	1
Window Size	5
Sg	0

Table 4: Training Parameters of Word2Vec

3.1.2 GloVe Embedding Model

Pre-trained 50D GloVe vectors were used to translate training dataset made up of sequence numbers to vector representation rather than learning word vectors from scratch. The GloVe vector for a vocabulary word will be retrieved and saved in an embedding weight matrix if it is discovered in the GloVe embedding dictionary by the trained tokenizer. If a vocabulary word is not included in the GloVe embedding dictionary, on the other hand, a zero-word vector is saved in the embedding weight matrix. In a neural network's embedding layer, embedding weight matrices were applied as pre-trained weights. The GloVe embedding outcome on training datasets is shown in Figure 3. According to the graph, around 25% of the vocabulary items in the trained tokenizer were not present in the GloVe embedding dictionary.

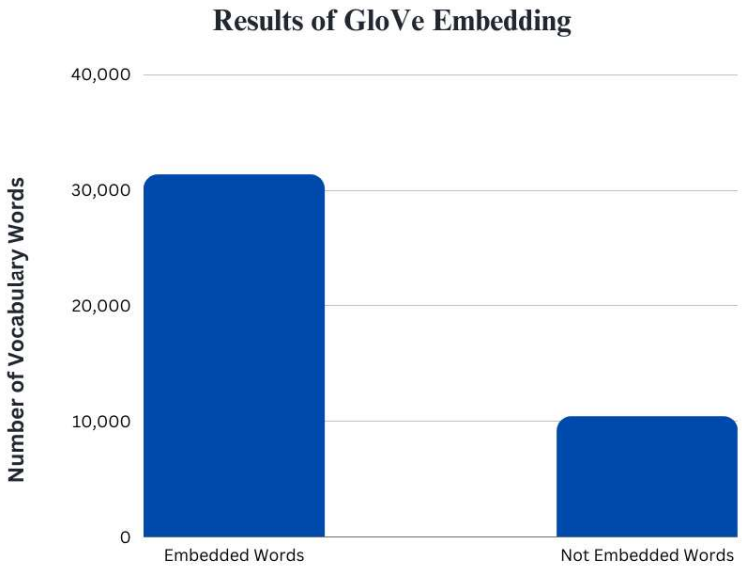

Figure 3: Results of GloVe Embedding Model.

3.1.3 Random Embedding Weights Initialization Model

The embedding weights can be initialised randomly using a uniform distribution in place of pre-trained embedding weights. When using embedding layers without pre-trained embedding weights, the weight and trainable parameters of an embedding layer are set to None and True, respectively. If pre-trained weights are used, however, the weight parameter must be set to weight = [pre_trained_weights] and the trainable parameter must be set to False in order to prevent the back-propagation method from updating the pre-trained weights after each training iteration.
The benefit of random embedding weight initialization is that it guarantees that every word in the vocabulary has non-zero embedding weights. Due to the training vocabulary size being substantially smaller than the one used to create pre-trained GloVe vectors; it may be difficult to achieve ideal weights.
Below are the different types random embedding weights initialization techniques which are used in the proposed work.

- **Random Uniform Weight Initialization:** In this technique, the embedding layer's weights are initialised with random values chosen at random from a uniform distribution. The weights are equally sampled from the distribution's range, which can be defined as [-0.05, 0.05].

- **Random Normal Weight Initialization:** In this case, random values taken from a normal distribution are used to initialise the weights. The weights are sampled in accordance with the distribution's defined mean and standard deviation. Gaussian initialization is another name for this approach.

- **Random Glorot Initialization:** The embedding layer's number of input and output units is taken into consideration while using the Glorot or Xavier initialization method. The weights are initialised with random values obtained from a distribution with zero mean and variance, where the number of input and output units is taken into account as well as a specific formula. The signal variance can be maintained using this technique both during forward and backward propagation.

3.2 Deep Learning Techniques

Deep learning models are powerful neural network models. They can be used to transform the features so as to form fairly complex non-linear decision boundaries. Various deep learning algorithms used for Twitter spam detection are discussed in this section.

3.2.1 Convolutional Neural Network

A popular deep learning model for computer vision applications like image classification, object identification, and picture segmentation is the convolutional neural network (CNN) [10]. CNN may also be used to learn three-dimensional video pictures and one-dimensional text data. This section will describe the architecture of CNN using two-dimensional picture data.

Multiple feature extraction layers are used in CNN designs to gradually extract multi-level hierarchical information from the input image. Convolution layer, pooling layer, fully-connected layer, and classification layer make up a standard CNN design. Figure 4 shows a CNN architecture as an illustration.

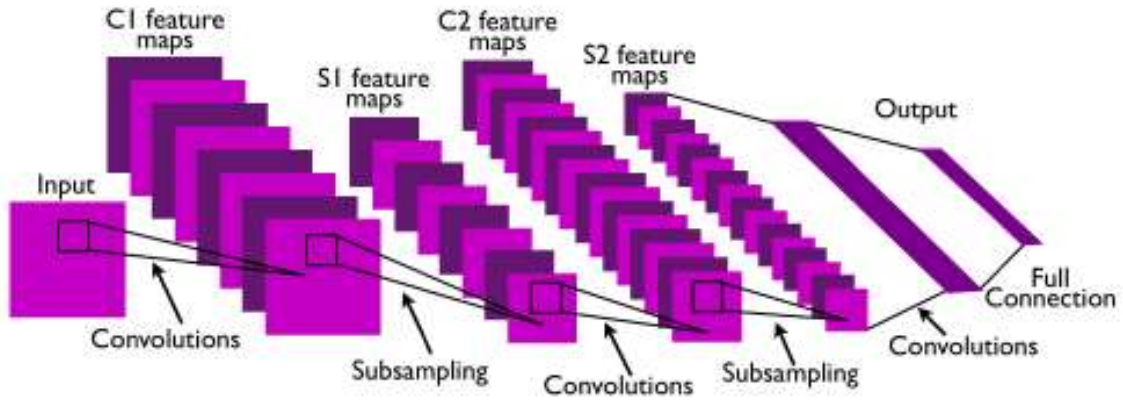

Figure 4: CNN Architecture [1]

3.2.2 Long Short-Term Memory

LSTM features a chain-like architecture with repeated modules. In contrast to RNN, which only has one activation function per module, LSTM [11] contains four activation functions. Figure 5 depicts an illustration of an LSTM design with three repeating modules.

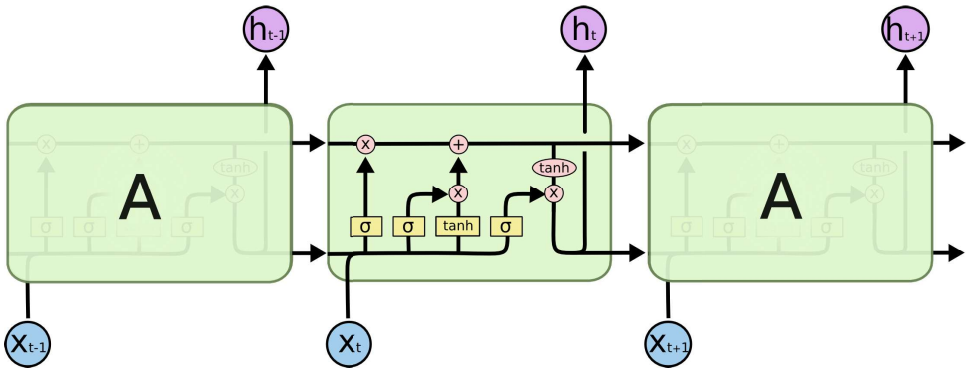

Figure 5: LSTM Architecture [2].

A LSTM module contains cell status, forget gate, input gate, and output gate. These gates are operated by the sigmoid activation functions which control information flow. Cell status denotes as *ct* has additional delay which allows the flow of information from previous module to be stored in the current cell. The cell status is modified by the forget gate and input gate denote as *ft* and *it* respectively. The forget gate determines which information from previous cell is important and not important. The decision is made by sigmoid activation function which outputs either a 0 or 1 based on *ht*-1 and *xt* If the output is 0, the information is eliminated. The input gate determines which information to be inserted into the cell state via sigmoid and hyperbolic tangent activation functions. The new cell state is updated by multiplying the old cell state with forget gate and adding the input gate. Finally, the output gate denotes as *ot* computes information from previous cell and current cell via sigmoid activation function. The output information is combined with the updated cell state and passes to next LSTM cell. The computation of information is based on element-wise multiplication and addition operations.

3.2.3 Bidirectional Long Short-Term Memory

A biLSTM, also referred to as a bidirectional LSTM, is a sequence processing model that consists of two LSTMs; the first model accepts the input exactly as it is, and the second model accepts a duplicate of the sequence in the reverse manner.
The network's data availability is effectively increased by biLSTM's unique architecture, providing the algorithm with better context. An illustration of the biLSTM architecture is found in Figure 6.

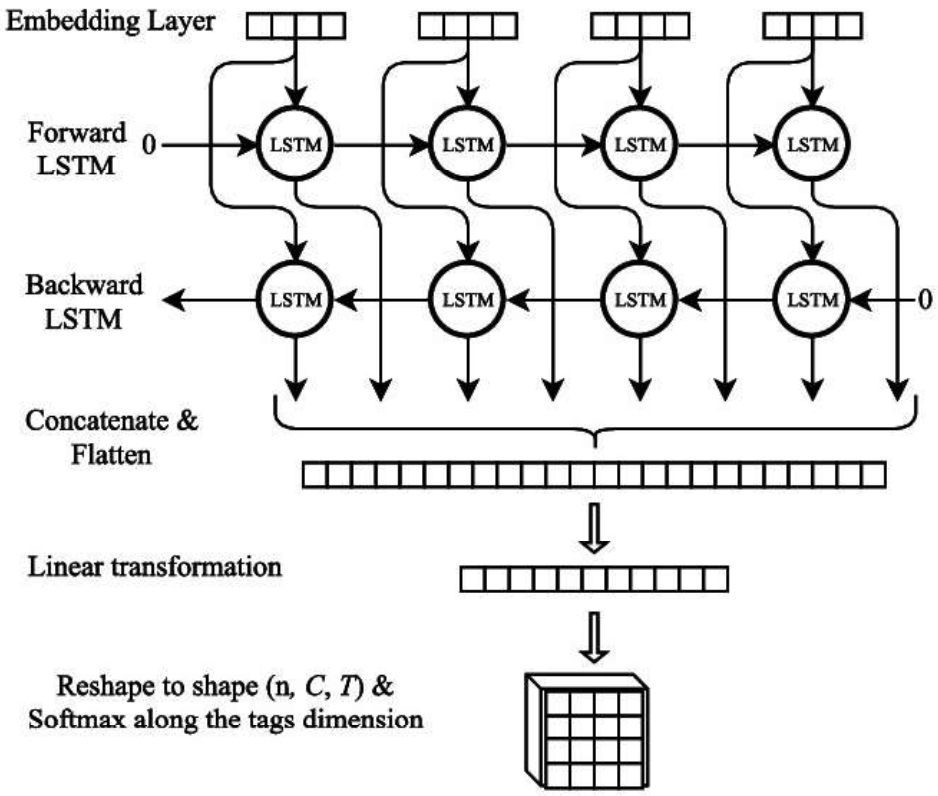

Figure 6: BiLSTM Architecture [3]

There are two LSTM models, one of which operates forward and the other of which operates backward. BiLSTMs are able to perform additional training because of this by sending the text sequence twice. In order to provide superior predictions, the biLSTM model completes more training on a given dataset than LSTM.

3.2.4 Self-Attention Transformer Model

A self-attention model called Transformer [12] can pay attention to various locations in an input sequence and construct a representation of that sequence. This deep learning method is frequently employed in NLP applications like text classification, language modelling, and translation. A typical Transformer architecture for language translation models is composed of tiers of encoder and decoder layers. Every decoder and encoder are identical to one another and each have a unique set of weights. Not all NLP applications, nevertheless, employ this style of architecture. Decoder layers are used in language model architecture to forecast the words that will come after the input phrase after decoding.

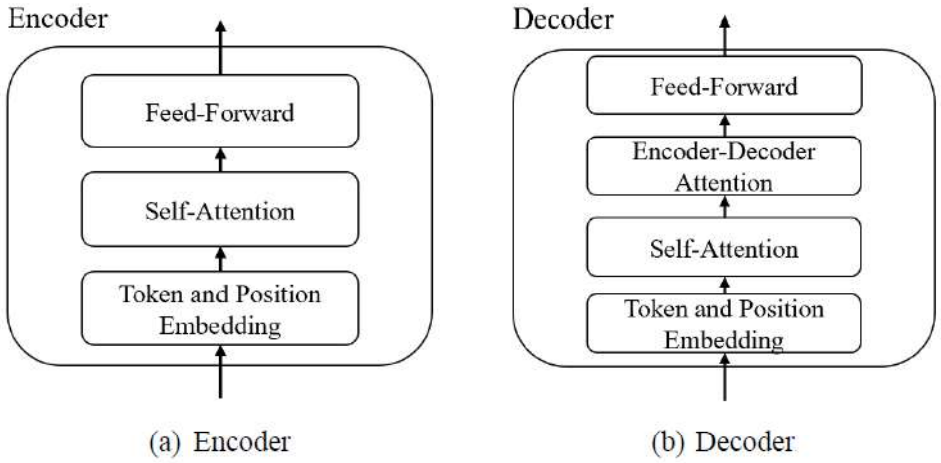

Figure 7: Illustration of an Encoder and Decoder Layer in Transformer.

Figure 7 illustrates a standard Transformer encoder, which includes embedding layers, a self-attention layer, and a feed-forward neural network. Tokenized words in the input sentence are transformed into embedding vectors by the token embedding layer. Transformer models do not have recurrent qualities to remember the exact positions of words because all the words in a sentence are input simultaneously, in contrast to RNN and LSTM models. In order to give the model information about the word positions in an input sequence, positional embedding layer is employed. To create word embeddings in n-dimensional space based on semantic similarity and word positions in a phrase, the position embedding is combined with the token embedding.

Chapter 4

Experimental Results

This chapter describes the training parameters of the neural network model. We evaluated the effects of using random embedding weight initialization, pre-trained GloVe vectors, and word embeddings from the Word2Vec model. Lastly, classification results of LSTM, CNN, biLSTM, and Self-Attention Transformer are evaluated and compared.

4.1 Simulation Setup

The hardware resources used in this thesis are personal computer with AMD Radeon graphics card. The software resources used in this thesis are Python language version 3.6.10 with Pandas, NLTK, Scikit-Learn, Tensorflow, Keras packages installed. The software applications used in this thesis are free and open-sourced.

4.1.1 Training Parameters

The summary of the deep learning network architectures and training parameters is provided in Table 5. Binary cross entropy loss function was used to calculate mistakes in deep learning models built using Adam optimizer [37] with a learning rate of 0.001. With a batch size of 512, the training data is divided into smaller chunks and utilised to fit the model during 8 training epochs.

	LSTM	CNN	Transformer	biLSTM
Network Architecture	- Embedding Layer - LSTM Layer - Dense Layer - Dropout Layer - Dense Layer (Classification)	- Embedding Layer - Convolution Layer - Global Max Pooling Layer - Batch Normalization Layer - Dense Layer - Dropout Layer - Dense Layer (Classification)	- Token Embedding Layer - Position Embedding Layer - Multi Head Attention Layer - Dense Layer - Layer Normalization Layer - Dropout Layer - Global Average Pooling Layer - Dropout Layer - Dense Layer - Dropout Layer - Dense Layer (Classification)	- Embedding Layer - biLSTM Layer - Dense Layer - Dropout Layer - Dense Layer (Classification)
Optimizer	Adam			
Learning Rate	0.001			
Loss Function	Binary Cross Entropy			
Number of Training Epochs	8			
Mini Batch Size	512			

Table 5: Network Architecture and Training Parameters of LSTM, CNN, Transformer, and biLSTM.

4.1.2 Dataset

The Twitter dataset that was selected to work on is called "Social Honeypot Dataset" created by Lee *et al.* [4]. This dataset was created by the deployment of 60 social honeypot accounts on Twitter which posted random messages to engage with Twitter users. Twitter accounts that interacted with these social honeypot accounts were recorded. The distributions of legitimate users and content pollutants collected in the experiment are tabulated in Table 6.

Sr. No.	Class	No. of User Profiles	No. of Tweets
1.	Legitimate Users	19,276	3,263,238
2.	Content Polluters	22,223	2,380,059

Table 6: Description of Social Honeypot Dataset [4]

The original dataset contains six text files as described in Table 7. In this, only "*content_polluters_tweets.txt*" and "*legitimate_users_tweets.txt*" were used since the proposed method is to detect spam tweets.

Sr. No.	Text File	Features
1.	“content_polluters.txt” “legitimate_users.txt”	UserID, CreatedAt, CollectedAt, NumberOfFollowings, NumberOfFollowers, NumberOfTweets, LengthOfScreenName, LengthOfDescriptionInUserProfile
2.	“content_polluters_followings.txt” “legitimate_users_followings.txt”	UserID, SeriesOfNumberOfFollowings
3.	“content_polluters_tweets.txt” “legitimate_users_tweets.txt”	UserID, TweetID, Tweet, CreatedAt

Table 7: List of Text Files and its Corresponding Features in Social Honeypot Dataset.

4.1.3 Evaluation Metrics

Accuracy, precision, recall, and F-score are the evaluation metrics that are computed using the confusion matrix in Figure 8.

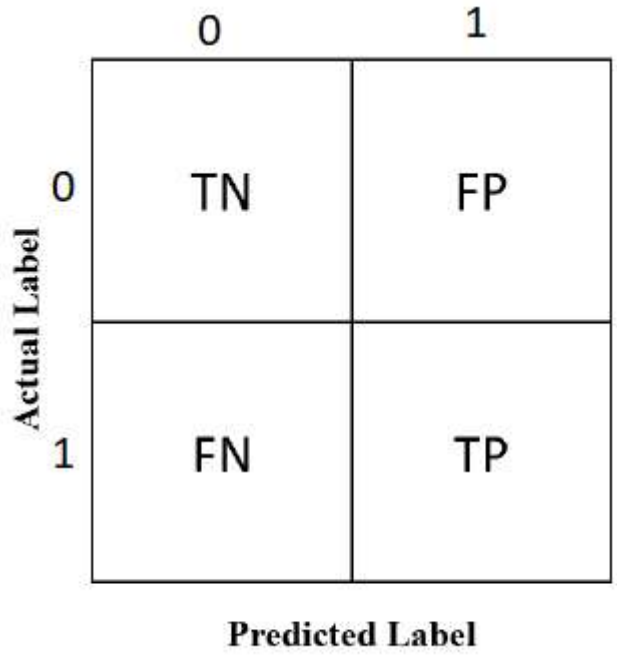

Figure 8: Confusion Matrix.

where, **True Negative (TN)** are the non-spam tweets that are correctly predicted. **True Positive (TP)** are spam tweets that are correctly predicted. **False Negative (FN)** are spam tweets that are falsely predicted. **False Positive (FP)** are non-spam tweets that are falsely predicted.

The formulae to compute the evaluation metrics are as follows:

Accuracy : The accuracy describes how many tweets are correctly predicted. It is computed as follows:

$$\text{Accuracy} = \frac{TN + TP}{TN + TP + FN + FP} \tag{1}$$

Precision : Precision computes the number of actual spam tweets among all the predicted spam tweet as follows:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2}$$

Recall : Recall computes how many tweets are predicted correctly among all the spam tweets. It is computed as follows

$$\text{Recall} = \frac{TP}{TP + FN} \tag{3}$$

F-score : F-score is the harmonic mean of Precision and Recall. F-score is computed as shown in the following

$$\text{F-score} = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{4}$$

4.2 Results and its Analysis

The deep-learning techniques such as Long short-term memory (LSTM), Convolutional Neural Network (CNN), Bidirectional LSTM (BiLSTM), and Self-Attention Transformer was utilised in the proposed work to determine whether the tweet was spam or not. Word2Vec embedding, pre-trained GloVe embedding, random normal embedding weight initialization, random Glorot initialization, and random uniform embedding weight initialization were all applied individually with these deep-learning technique in the proposed work.

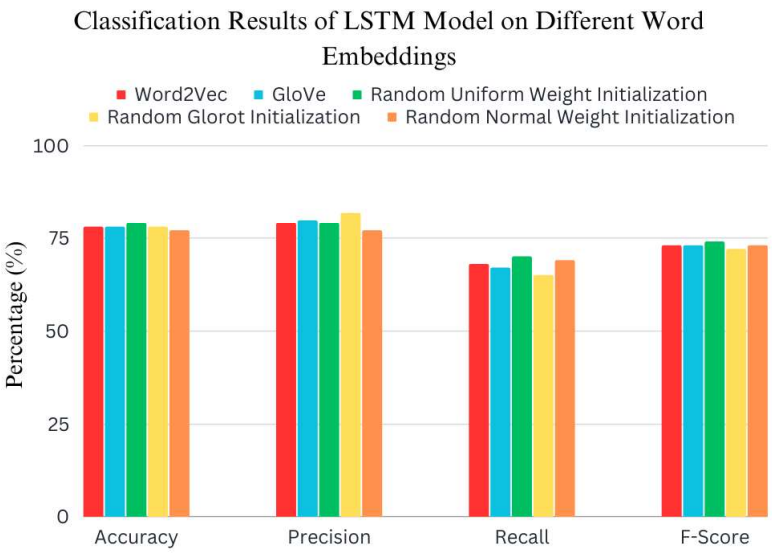

Figure 9: Classification Results of LSTM model on Different Word Embeddings.

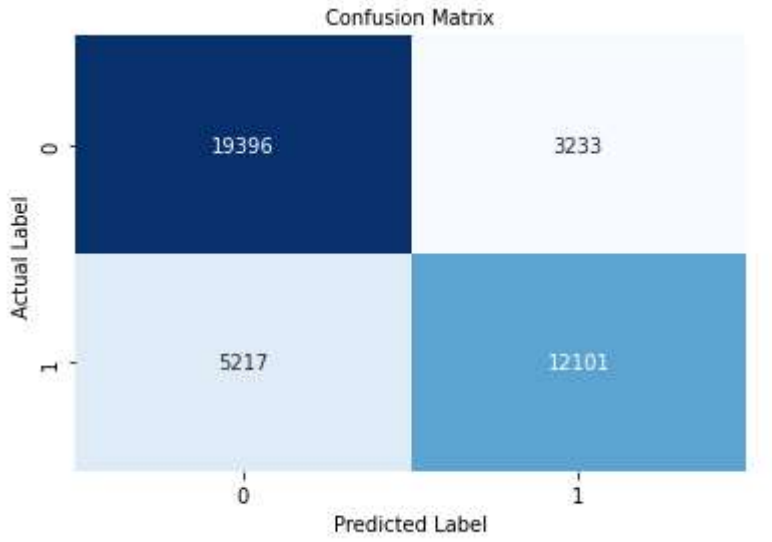

Figure 10: Confusion Matrix of LSTM model with Random Uniform Weight Initialization on Test Dataset.

Based on the classification results of LSTM model obtained as shown in Figure 9, random uniform weight initialization outperformed the other word embeddings. This result has highlighted the importance of assigning embedding weights to all vocabulary words, including unknown words. In pre-trained embedding weights, there is no embedding weights for unknown words which are not learnt by Word2Vec model and in GloVe vectors. As a result, this will cause poor generalization of model to new test data. A confusion matrix, like the one in Figure 10, can be used to visualise the number of incorrectly and properly identified test samples.

Spam Tweets	**Feature**
retweet usermention ready usermention usermention usermention usermention usermention usermention usermention usermention usermention	Retweet and multiple usermention
retweet usermention usermention next video interview series url	Mixture of usermention, retweet, and URL
legalize make sure everything business legit paper work	Plain normal text

Table 8: Examples of Common Spam Tweets Predicted as Non-spam Tweets.

Spam Tweets	**Feature**
retweet usermention sexy lady usermention usermention usermention usermention usermention usermention usermention usermention usermention	Retweet and multiple usermention
retweet usermention retweet usermention new york time cut newsroom job staff url	Mixture of usermention, retweet, and URL
must get back healthy eating plan far recently really feeling	Plain normal text

Table 9: Examples of True Negative Tweets.

Misclassified and correctly classified tweets were extracted and analyzed to investigate the model's prediction limitations. Based on the analysis, examples of the most common false negatives are extracted and shown in Table 8. False negative tweets contain features

such as 1) Retweet and multiple usermentions, 2) mixture of usermention, retweet, and URL, and 3) plain normal text without those common tweet features. These occurrences were expected because their features are matched to the features in true negative tweets as shown in Table 9. Despite false negatives, there are some common features which are correctly classified as spam tweets as shown in Table 10. Most common spam feature is having URL attached at the back of the sentence. Nevertheless, LSTM model can successfully classify spam tweets which do not contain common spam features such as URL and user mention.

Spam Tweets	**Feature**
movie maker peter king middle earth becomes knight url	URL at the back of sentence
retweet usermention heiress mistress sweetness lady two land url	Mixture of usermention, retweet, and URL
finally awesome making money program whole year money making business also change	Plain normal text

Table 10: Examples of True Positive Tweets.

The second experiment was performed using CNN to compare with LSTM's classification results. Based on Figure 11, CNN has poorer performance as compared to LSTM across all metrics. There was a significant drop of 15% in accuracy when random weights embedding is used.

In the third experiment, biLSTM outperformed CNN on the same datasets based on Figure 12. On the other hand, the classification results of biLSTM are comparable to LSTM in terms of precisions, specificity, and f-score. Although the accuracy is 1.14% below LSTM's, biLSTM has displayed potential to outperform LSTM before hyper-parameters tuning is performed.

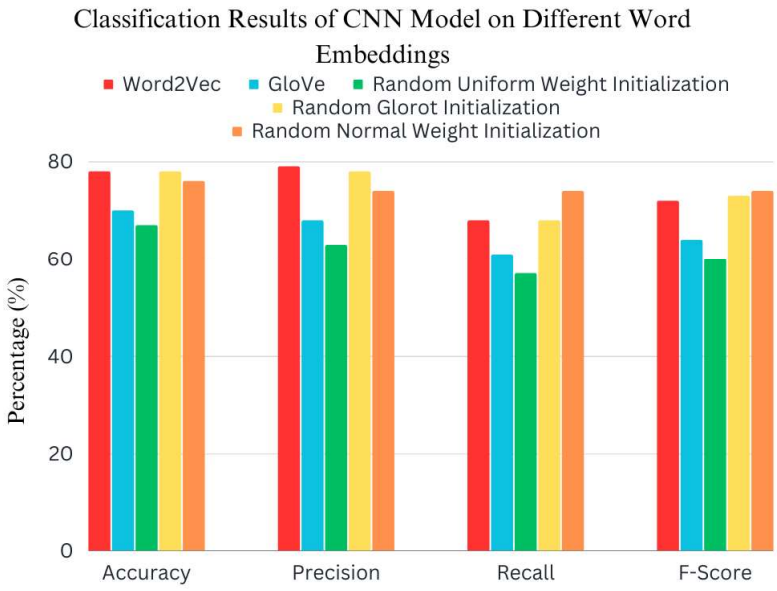

Figure 11: Classification Results of CNN model on Different Word Embeddings.

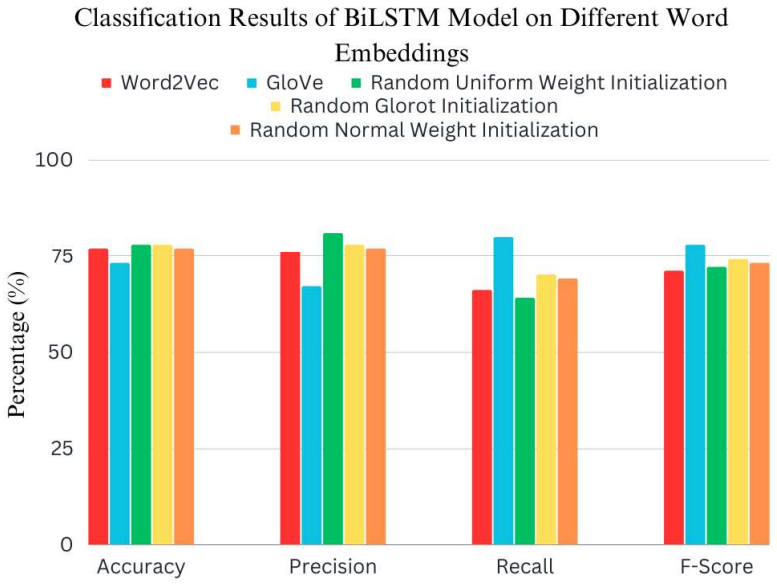

Figure 12: Classification Results of biLSTM model on Different Word Embeddings.

Figure 13: Classification Results of Self-Attention Transformer model on Different Word Embeddings.

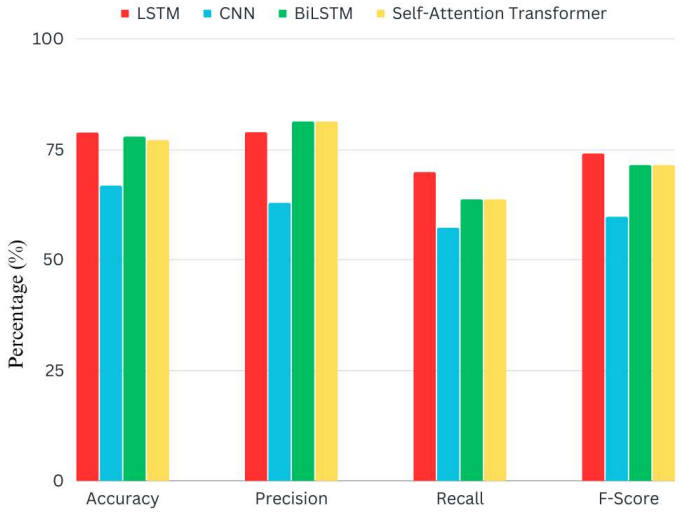

Figure 14: Classification Results of Random Uniform Weight Initialization model on Different Deep Learning models.

The fourth experiment was performed using Transformer to compare with LSTM. Based on Figure 13, Transformer has comparable performance as compared to LSTM across accuracy. There was a little drop of 2.16% in accuracy when random weights embedding

is used.

Based on the classification results of Random Uniform Weight Initialization model obtained as shown in Figure 14, LSTM model outperformed the other deep learning models.

Chapter 5

Conclusion and Future Work

The evaluation of spam detection on the Twitter dataset was done using deep learning models like LSTM, CNN, biLSTM, and Transformer. In terms of spam accuracy, recall, and f-score, experimental results demonstrated that LSTM and Transformer outperformed. In comparison to previous efforts, the size of the training data employed in the suggested approach is around 100 times larger. As a result, the proposed technique has an 79% spam precision and is more universal to new tweets. In conclusion, the best spam detection performance in terms of spam accuracy and recall is achieved by LSTM with uniform random embedding weight initialization.

By experimenting with different Transformer model variations, the spam detection effectiveness can be further enhanced. Transformer's attention function has demonstrated its ability to draw attention to word locations and word relevance in a text sequence. The spam precision of 78% attained by the traditional Transformer network design is comparable to that of proposed model, LSTM. Thus, the hyper-parameters of the Transformer model can be adjusted and further investigated to enhance Twitter spam detection.

Additionally, the effectiveness of the pre-trained LSTM model's spam detection may be assessed using the latest tweets. From December 30, 2009, to August 2, 2010, the dataset utilised for this study was created. The most recent tweets must be collected via a web crawler in order to maintain the effectiveness of spam detection because spammers' methods are continually changing.

References

[1] Yann LeCun, Koray Kavukcuoglu, and Clement Farabet. Convolutional networks and applications in vision. In *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, pages 253–256, 2010.

[2] C. Olah. *Understanding LSTM Networks*. https://colah.github.io/posts/2015-08-Understanding-LSTMs/ (accessed 31 March, 2021).

[3] Savelie Cornegruta, Robert Bakewell, Samuel Withey, and Giovanni Montana. Modelling radiological language with bidirectional long short-term memory networks. In *Proceedings of the Seventh International Workshop on Health Text Mining and Information Analysis*, pages 17–27, Auxtin, TX, November 2016. Association for Computational Linguistics.

[4] Kyumin Lee, Brian Eoff, and James Caverlee. Seven months with the devils: A long-term study of content polluters on twitter. *Proceedings of the International AAAI Conference on Web and Social Media*, 5(1):185–192, Aug. 2021.

[5] Twitter Twitter Inc. 25 Jan 2021. [Online] Available: https://twitter.com.

[6] Bo Wang, Arkaitz Zubiaga, Maria Liakata, and Rob Procter. Making the most of tweet-inherent features for social spam detection on twitter, 2015.

[7] Yoel Roth and Del Harvey. How twitter is fighting spam and malicious automation, 2018.

[8] Onur Varol, Emilio Ferrara, Clayton A. Davis, Filippo Menczer, and Alessandro Flammini. Online human-bot interactions: Detection, estimation, and characterization, 2017.

[9] Xinbo Ban, Chao Chen, Shigang Liu, Yu Wang, and Jun Zhang. Deep-learnt features for twitter spam detection. In *2018 International Symposium on Security and Privacy in Social Networks and Big Data (SocialSec)*, pages 208–212, 2018.

[10] Yann LeCun and Yoshua Bengio. *Convolutional Networks for Images, Speech, and Time Series*, page 255–258. MIT Press, Cambridge, MA, USA, 1998.

[11] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 11 1997.

[12] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[13] Nello Cristianini and Elisa Ricci. *Support Vector Machines*, pages 928–932. Springer US, Boston, MA, 2008.

[14] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, oct 2001.

[15] Steven Bird, Ewan Klein, and Edward Loper. *Natural Language Processing with Python*. 01 2009.

[16] George A. Miller. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41, nov 1995.

[17] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics.

[18] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, Oct 1986.

[19] Andrew Mccallum and Kamal Nigam. A comparison of event models for naive bayes text classification. *Work Learn Text Categ*, 752, 05 2001.

[20] N. S. Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185, 1992.

[21] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, Mar 1986.

[22] J.S. Cramer. The Origins of Logistic Regression. Tinbergen Institute Discussion Papers 02-119/4, Tinbergen Institute, December 2002.

[23] F. Benevenuto, G. Magno, T. Rodrigues, and V. Almeida. Detecting spammers on twitter. In *Collaboration, Electronic messaging, Anti-Abuse and Spam Conference (CEAS)*, July 2010.

[24] Mokhtar Ashour, Cherif Salama, and M. Watheq El-Kharashi. Detecting spam tweets using character n-gram features. In *2018 13th International Conference on Computer Engineering and Systems (ICCES)*, pages 190–195, 2018.

[25] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013.

[26] Fabio Martinelli, Francesco Mercaldo, and Antonella Santone. Social network polluting contents detection through deep learning techniques. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–10, 2019.

[27] M. Minsky and S. Papert. *Perceptrons; an Introduction to Computational Geometry*. MIT Press, 1969.

[28] Feng Wei and Uyen Trang Nguyen. Twitter bot detection using bidirectional long short-term memory neural networks and word embeddings. In *2019 First IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA)*, pages 101–109, 2019.

[29] Antigoni-Maria Founta, Despoina Chatzakou, Nicolas Kourtellis, Jeremy Blackburn, Athena Vakali, and Ilias Leontiadis. A unified deep learning architecture for abuse detection, 2018.

[30] Zulfikar Alom, Barbara Carminati, and Elena Ferrari. A deep learning model for twitter spam detection. *Online Social Networks and Media*, 18:100079, 2020.

[31] K. Archchitha and E.Y.A. Charles. Opinion spam detection in online reviews using neural networks. In *2019 19th International Conference on Advances in ICT for Emerging Regions (ICTer)*, volume 250, pages 1–6, 2019.

[32] Jenq-Haur Wang, Ting-Wei Liu, Xiong Luo, and Long Wang. An LSTM approach to short text sentiment classification with word embeddings. In *Proceedings of the 30th Conference on Computational Linguistics and Speech Processing (ROCLING 2018)*, pages 214–223, Hsinchu, Taiwan, October 2018. The Association for Computational Linguistics and Chinese Language Processing (ACLCLP).

[33] Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew. Extreme learning machine: Theory and applications. *Neurocomputing*, 70(1):489–501, 2006. Neural Networks.

[34] D Ganesh Kumar, M Kameswara Rao, and K Premnath. A recurrent neural network model for spam message detection. In *2020 5th International Conference on Communication and Electronics Systems (ICCES)*, pages 1042–1045, 2020.

[35] Wes McKinney. Data Structures for Statistical Computing in Python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 56 – 61, 2010.

[36] Radim Řehůřek and Petr Sojka. Software framework for topic modelling with large corpora. pages 45–50, 05 2010.

[37] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.