COMPROMISED TWEET DETECTION USING WEIGHTED SUB-WORD EMBEDDINGS

by

Mihir Joshi

Submitted in partial fulfillment of the requirements
for the degree of Master of Computer Science

at

Dalhousie University
Halifax, Nova Scotia
July 2019

To my mother and my sister

Table of Contents

List of Tables

List of Figures

Abstract

Extracting features and writing styles from short text messages for compromised tweet detection is always a challenge. Short messages, such as tweets, do not have enough data to perform statistical authorship attribution. Besides, the vocabulary used in these texts is sometimes improvised or misspelled. Therefore, in this thesis, I propose combining four feature extraction techniques namely character n-grams, word n-grams, Flexible Patterns and a new sub-word embedding using the skip-gram model. The proposed system uses a Multi-Layer Perceptron to utilize these features from tweets to analyze short text messages. This proposed system achieves 85% accuracy, which is a considerable improvement over previous systems. Furthermore, Siamese networks are employed to model the representation of user tweets in order to identify them based on a limited amount of ground truth data. The results show that the proposed system achieves a promising accuracy as the number of authors increase.

List of Abbreviations Used

RAT Remote Administration Tool.

CNN Convolutional Neural Network.

MLP Multilayer Perceptron.

IP Internet Protocol.

CIDR Classless Inter-Domain Routing.

SMTP Simple Mail Transfer Protocol.

VPN Virtual Private Network.

AA Authorship Attribution.

SPI Simplified Profile Intersection.

SVM Support Vector Machines.

RCVI Reuters Corpora.

LSTM Long Short Term Memory.

EMD Earth Mover's Distance.

DAN Deep Averaging Network.

PTB Penn Treebank.

GRU Gated recurrent units.

KNN K-nearest Neighbors.

TF-IDF Term frequency-inverse document frequency.

BOW Bag of Words.

CBOW Continous Bag of Words.

ReLU Rectified Linear Unit.

TanH Hyperbolic Tangent.

ADAM Adaptive Moment Estimation.

RMSProp Root Mean Square Propagation.

AdaGrad Adaptive Gradient.

MSE Mean Square Error.

CW Content Word.

HFW High-Frequency Word.

t-SNE t-distributed Stochastic Neighbor Embedding.

Acknowledgements

Firstly, I want to express my heartfelt gratitude to my supervisor, Dr. Nur Zincir-Heywood for accepting me as her student, providing me with a graduate funding which makes it easier for me to put the focus on my research, and helping me throughout the thesis with her ideas and feedback. She constantly assisted me with suggestions and technical details and patiently worked with me whenever I was slow in understanding somethings.

I would also want thank my friend Parmeet Singh who used to come up with some outrageous ideas and was always ready to discuss my problems which ultimately helped me out whenever I was stuck with something.

Lastly, I want to thank my mother for giving me the support I needed to come to Dalhousie University and study and my sister for saying words of encouragement throughout my thesis.

Chapter 1

Introduction

With the advancement in network technologies and the vast amount of freely available resources over the internet, it is easier even for an unskilled person to hack someone's online account. Where an advanced attacker could use sophisticated resources like keyloggers, Remote Administration Tool (RAT) [6] and Trojan horses [7], an inexperienced person could use phishing pages or various GUI based brute force tools readily available online. Figure 1.1 depicts the cost associated with the network attacks.

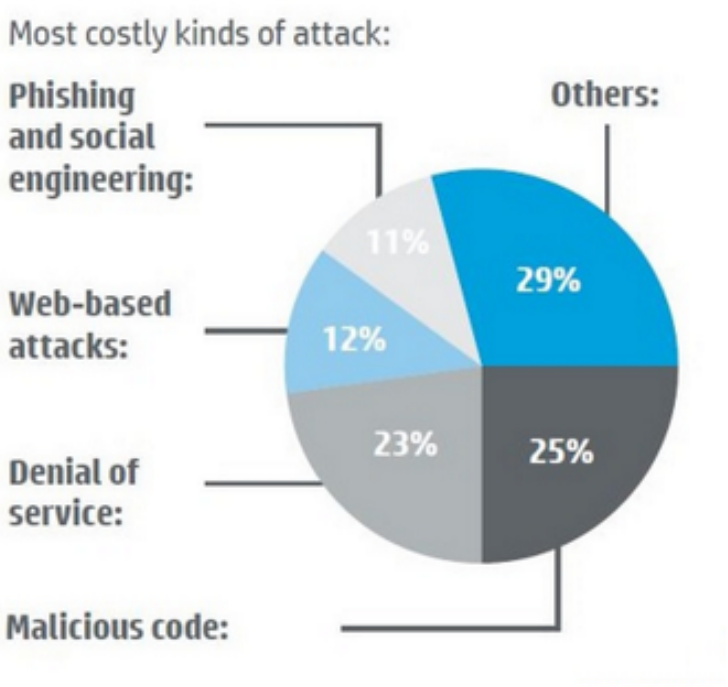

Figure 1.1: Cost associated with different types of network attacks [2]

In addition to all of that, nowadays there are various instances of data breaches which is increasing quite linearly every year exposing millions of user accounts as shown in Figure 1.2. These compromised user accounts are then being sold on the

black market. Although websites employ security measures to prevent unauthorized access, once the account credentials are acquired, the attacker can masquerade as the user and then there are no provisions to separate him from the real owner of the account. The attacker would generally use the compromised account to send phishing links, spam, or fraudulent messages to the victimized account's friend list for the purpose of asking money or make them click on a link of an unwanted website or a piece of unwanted or malicious application.

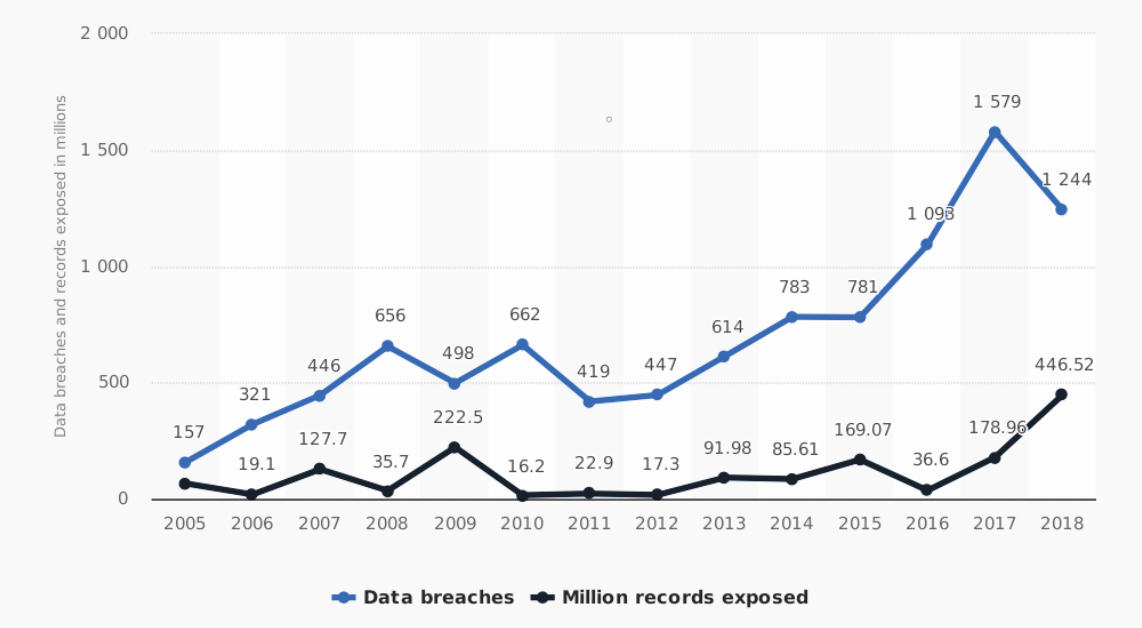

Figure 1.2: Data breaches and account exposed in United States [3]

Earlier, the stolen accounts are generally e-mail accounts but now they can belong to any social media websites like Twitter, Facebook or YouTube as well. Due to the restriction on the length of a text on these websites, one can write something that is only a couple of lines long. Also, the number of words in these social media messages ranges from a mere five to a maximum of twenty. Some of these websites like Twitter even restricts the user to post texts that are less than 140 characters. These type of shot texts are also called micro-messages. As the length of the text decreases, it's harder to find a pattern in the text and learn the writing style of the person to correctly identify whether the message belongs to them or someone else.

Most of the current systems [8, 9] aims to classify longer text corpora such as PAN-2018[10].The text analysis and natural language processing approaches employed by

these systems use statistical feature extraction techniques such as term frequency, inverse document frequency and a bag of words approach. Due to the feature extraction process being statistical in nature, all these techniques require a certain amount of data to use them effectively for determining patterns or perform authorship attribution and alone would not perform effectively in case of micro-messages. Thus, having short text messages such as tweets which are around 140 characters or less makes it difficult to identify the patterns on a given text and make predictions about the author.

Even the neural network approaches like the one implemented by Shrestha et al. [11] used a convolutional neural network (CNN) architecture using character embeddings instead of word embeddings for short texts showed less than five percent improvement on previous researches [5, 12] in this area.

In this thesis, I explore four feature extraction techniques - word n-grams, character n-grams, flexible patterns and TF-IDF weighted sub-word embeddings for effective classification of micro-messages. My goal is to identify users based on their writing styles. Such a system can be leveraged for compromised user detection. To this end, I start by implementing a simpler approach; training a multilayer perceptron (MLP) [13] after combining the two consecutive records from the same author just using the word and character n-grams to train my model. Then I apply the feature known as Flexible patterns [5] after modifying the existing method and finally introducing the new feature based on the word vector approach [14]. To the best of my knowledge, this is the first time the combination of these four feature extraction techniques is used for the classification of micro messages for modeling the writing styles of users. Another contribution of my system is demonstrating how the word embedding and modification of existing flexible pattern approach not only works well with the existing dataset by Schwartz et al. [5] but and can be easily transposed to other datasets as well. My system [15] is a significant improvement over previous works in this area.

Afterwards, I also work on a model that learn latent representations from the text. The latent similarity between two documents is the semantic closeness between them and is based on the context they are used. In this paper, I employ Siamese Networks[16] to analyze and identify two texts either from the same author or from different authors and showing how accuracy changes when I increase the number

of tweets and the number of authors. To evaluate my system, I also conduct the student's T-test on the data to see if the difference in the results are statistically significant or not. The T-test results show that the decrease in the performance of the user detection system is not significant as the number of authors increases.

The thesis is organized into the following sections. Chapter 2 introduces the related works. In Chapter 3, I present my model and the architecture followed by the datasets and the feature extraction techniques. Chapter 4 shows the results of the proposed system compared with the previous works. Lastly, in Chapter 5, I discuss conclusions and future work directions.

Chapter 2

Related Works

One of the most remarkable inventions of the twentieth century is Virtual Identities [17]. The virtual identity or as some websites (like Twitter) call it a 'handle' is an online personification of someone using that website. In this age, where there are a plethora of social media websites, most people have multiple virtual identities or online accounts which they use to communicate with each other over the internet. Although, there are various provisions to safeguard these accounts, like everything else on the internet these accounts are can be hacked and the user is compromised. In this chapter, I looked at the previous work done on identifying if the user is compromised or not. There are mainly three approaches to do that, first is the disclosure by the parent company that they have been compromised, second is to recognize whether the account is hacked based on the IP data of the account and lastly is to classify and separate the compromised account from the original user of that account based on textual data shared between the users, i.e. attributing the text to the original user of that text. This is called authorship attribution [18]. The following is a brief summary of these systems to the best of my knowledge.

2.1 Company's Disclosure

There are two types of incidences, one when only a single account is compromised and the other where there is a website-wide attack. The attacks on an individual account like the one on Home Box Office's (HBO) twitter account [19] are often discreet and reported by the person whose account is compromised. On the other hand, the data breaches on a company's website cause thousands of compromised accounts if not millions. One of the biggest data breaches is in 2013 when data related to three billion Yahoo accounts [20] is leaked. Even recently, when there are more sophisticated security algorithms, there are still frequent cases of a data breach. Some of the famous ones are -

- Marriott International [21]
- Truecaller [22]
- Facebook [23]

Although, it is always better to learn from the official source about the account breach before the information is released by the parent company the attacker might already miss using the information obtained from these accounts.

2.2 IP or Location Data Analysis

The idea behind this approach is that all the email data and in some cases, social media data (Twitter) is tagged with the geolocation which can be used to identify the compromised user messages. Lin et al. [24] proposed a system to detect a forgery in e-mails using this method. The assumption of their approach is that a user is most likely to receive emails from a limited number of locations. Any e-mails received from an account that usually sends messages from one location claiming to be from a different place could likely be a compromised account. They captured IP address from the headers of an e-mail sample and map it to the physical location of the server where the message is generated. Finally, they created an email classifier for known senders based on these properties of a sender -

- Limited IP addresses
- Limited CIDR[1] addresses
- Consistent city geolocation
- consistent country geolocation

Using the combinations of these four properties, they built a client-side script that would mark emails of known senders.

Similarly, Schäfer [25] proposed another approach for identifying compromised email accounts using the information available in the meta-data of an email without using the actual body of the email messages. They extracted the time and source of

[1]Classless Inter-Domain Routing

the IP of an account that is used to send an email. Using this data they implemented a system that detects anomalies based two features -

- Country Counting - Using the geolocation data they counted the number of locations from where the same account is connected to an SMTP server in a given time period. The shortcoming of this approach is when the person is traveling the system might identify the user's account as a bot.
- Theoretical Geographical Travelling Speed - To overcome the above problem they proposed this feature. They argued that there would be an ample amount of time gap when a person travelling between countries would access the account whereas in case of the bot it would be rather frequent.

The flaw in their approach is that there are cases when an actual bot is accessing the account from different locations within a single country. Moreover, sometimes the user has multiple devices that connect using different IPs or a user themselves use a proxy or a VPN to hide their original location.

2.3 Authorship Attribution

The process of associating the original author of a text with the given text, based on the other messages of that author, is called Authorship Attribution (AA) . Apart from other applications of authorship attribution, such as detecting plagiarism in text, and identifying the author of the anonymously written old unknown books, it can be used to identify a compromised user account based on the messages sent by that account and comparing them with previous messages received from the same user. Authorship attribution is an unstructured data classification problem but it is different from other such classifications as it not only dependents on the nuances of the writing style of an author, but the process is very different for short texts compared to longer texts [26]. There are two ways to identify the author of an unknown text -

- Text Classification - It is a process of assigning a label (author name) to an unknown text document from the existing set of labels. In machine learning, a text classifier would take text input and determine the approximate function based on that data that could suitably put an unlabelled sample into an existing

category. Naive Bayes [27], Support Vector machines [28], Decision trees [29] and various deep learning approaches can be used as a classifier.

- Text Similarity - It is a process to identify the sameness between two text document. It is a process to separate text which are different not only lexically but also semantically. For example, the sentences "Is leopard a fast animal?" and "How fast is a leopard?" are lexically close but semantically are very different from each other.

2.3.1 Text Classification

Although there is a vast amount of research on authorship attribution, my only focus is on the research on short texts analysis. Research on classifying authors of short text messages based on the writing styles primarily started after the advent of social media websites like Twitter and Facebook. Layton et al. [30] carried out one of the earlier researches on the short text using Twitter data. They proposed the lazy learning approach using character n-grams. Their approach is based on the previous work by Kešelj et al. [31], which uses character n-grams to make profiles of different authors. To determine an unknown document, the profile of that document is compared with all the authors using a distance metric that uses top M n-grams and how frequently they appear in the document. The premise behind this approach is that an author has the same frequency of n-grams in all its documents. Layton et al. [30] worked on the variation of this approach called the Simplified Profile Intersection (SPI) [32]. The SPI is a less complex approach approach that calculates the intersection size between the author and the document profile instead of the relative distance. It equals and sometimes outperforms the approach by Kešelj et al. [31] for smaller n (character n-grams) and higher values of m (profile size). The approach follwed by Layton et al. [30] summarized into following steps -

- combined all author documents
- calculate top m n-grams for the author and the unknown document
- Calculate the SPI similarity to identify the unknown document.

Their system achieved an accuracy of 70% for 50 authors keeping @*replies* in tweets and dropped upto 27% when they removed it.

One of the most significant works in this area is done by Schwartz et al. [5]. They extended the character n-gram approach when they proposed the supervised learning model using Support Vector Machines (SVM) . They presented the concept of k-signatures, which is the unique signature that appears in k% of an author's document. They argued that there is at least one signature, in each author, and it successfully captures the writing style of an author. They extracted three features from each tweet -

- Character n-grams
- Word n-grams
- Flexible pattern

The last feature extraction technique (Flexible pattern) is based on functional words Koppel et al. [33]. Flexible patterns are obtained from the plain text in an unsupervised way. Finally, using all these features, they trained their SVM classifier with 10-fold cross-validation. Their system achieved an accuracy of 71% for 50 authors.

Combining deep networks with supervised learning, Rhodes [34] implemented a model based on convolutional neural networks (CNN) using word embeddings [35]. They created word vectors that are trained using the skip-gram approach and negative sampling from the Google news dataset while using random initialization for unseen words. They feed the resulting vector to a CNN [36]. The CNN model was based on a similar model [37] for language processing, where the activation function is changed to rectified linear units and introducing dropouts. Their approach demonstrated that the system performs well on longer text sequences. Although their research was based on longer texts [38], it formed the foundation for the system implemented by Shrestha et al. [11]. They make use of a similar architecture; a convolutional neural network (CNN) using character embeddings instead of word embeddings for short texts. The CNN model takes character unigram or bigram as an input that passes through the character embedding layer before feeding it to the convolutional layer. The architecture for their system -

- Character unigrams or bigrams as input
- Embedding layer that is initialized with random weights
- Three convolutional layer and max pooling after every layer
- A fully connected layer for classification

Similar datasets and evaluation criteria as Schwartz et al. [5] are used to determine the performance of the system. This approach increases the overall accuracy to approximately 76%.

A more recent study on this field [1] used word embeddings and the neural network to identify the authors of short text messages. They used the three different datasets namely Reuters Corpora (RCVI) [39], Enron dataset [40], and Twitter dataset [41]. The used the most frequent hundred thousands words from the RCVI to train the embeddings. These embeddings in turn generated the high-level features from the other two corpora by concatenating the vector of means and standard deviations. They then used the feed forward neural network to identify the authors. They used five authors (*Ashley_Nunn75, bradshaw1984, shawnevans81 , terrymarvin63, and WhieRose65*) from the twitter dataset and calculated the results averaged over twenty runs. Their system achieved 84% validation accuracy.

Finally, Jain et al. [42] proposed a system for Spam detection social media texts such as Facebook by using a combination of Convolutional Neural Network (CNN) and Long Short Term Memory (LSTM) network architectures. They argue that this system better captures diverse text by using pretrained embeddings. The CNN extracts the n-gram features from the text whereas LSTM detects the long-term dependencies. Their model performs better than each neural network architecture used on its own. They also compared the model against others from the literature with promising results.

2.3.2 Text Similarity

Most of the previous works are carried out on longer texts. In this section, I look at various techniques that measure text similarity based on the context, i.e., how two documents relate to each other semantically.

2.3.3 Word Movers Distance

Kusner et al. [43] proposed this system that calculates dissimilarity between two text documents based on Earth Mover's Distance (EMD) [44]. They showed that the semantic similarity can be measured as the sum distance between word embeddings in one document to the word embeddings of the other document. To calculate word embeddings, they vectorized the words using word2vec [35] after removing the stop words.

2.3.4 Variational Autoencoder

Chaidaroon and Fang [45] implement a system for text similarity by text hashing using variational autoencoders. The autoencoder can learn a low dimensional latent representation of the high dimensional data. They explained this by giving word examples - "government", "mafia" and "playboy" would be closer to the word Berlusconi in the non-linear distribution of the data. They used the bag of words approach before feeding the document into the encoder. The encoder itself has two hidden layers, followed by a stochastic layer and the decoder module. They used various datasets including Reuters Corpus Volume I (RCV1) [39] to train their system. For similarity evaluation, they query the relevant document to the hashed document in the test set using hamming distance [46].

2.3.5 Universal Sentence Encoder

Using transfer learning, Cer et al. [47] presented a system that produces sentence embeddings. They argued that although word2vec has a limited scope and performance can be further improved using sentence level embeddings. They proposed two encoder models -

- Transformer - This encoding architecture is based on the attention mechanism introduced by Vaswani et al. [48]. It works by considering both context of the word, i.e., the order of the words in a sentence and the identity of all the other remaining words. Therefore, it is more accurate but complex compared to the other model.

- Deep Averaging Network (DAN) - In this architecture, the word and bi-gram embeddings are averaged before using them in the feedforward deep neural network. It is less accurate than the Transformer model but has a linear runtime.

Both of these variations take Penn Treebank styled (PTB) tokenized the input and produces a 512-dimensional output. Finally, they concluded that though sentence level embeddings performed better compared to word embeddings alone, the combination of both would result in a better model.

2.3.6 Siamese Networks

Recently, there are several systems that implemented another form of artificial neural networks called Siamese networks [16]. Qian et al. [49] used them to verify the result of authorship attribution on C50 [50] and Gutenberg [38] datasets. Their model consists of a GRU followed by an average pooling layer. The input layer takes word vectors as inputs that are formed using 50-dimensional Glove [51] embedding. For the similarity measure, they used cosine distance.

Subsequently, Parikh et al. [52] implemented an architecture where they used skip-gram embeddings of words, character trigrams and combinations of these two approaches to train their encoder and learn the embeddings from the data. For training, they created 200 pairs, where 100 pairs belong to the same author and the other 100 are of different authors. They experimented with both BiLSTM and CNN as the encoders. Later the embeddings that are extracted from the network are used for classification using SVM, KNN , their cohort algorithm which is the binary classification of one author against each of the other authors, and a dense neural network. They experimented on CCAT-50 [53] dataset and their word based ensemble model produced the best results.

More recently, Boenninghoff et al. [54] constructed a hierarchical LSTM based siamese networks to compare the text from two authors. They used 300-dimensional GloVe [51] embeddings as the input vector. They evaluated their results on the dataset used by Halvani et al. [55]. Their hierarchical LSTM based Siamese network was proved to be effective even when the data is cross-topic and outperformed the previous work, that uses the same data, by 15%.

2.4 Summary

I presented here three approaches to detect the compromised account with increasing levels of difficulty. Each of the approaches has its own advantages and disadvantages -

- The first method doesn't require any analysis as the information about the compromised account is coming directly from the parent company. Although, this information is highly accurate, until the organization discloses these attacks, the attacker would have already caused damage by sending spam messages from that account. Thus, a machine learning base system could help automate the identification process.
- IP and geolocation based methods are only good in case of emails when there is IP data presents in the document header. Even then, there is no guarantee that the compromised account is not mimicking the original users IP.
- Author attribution approach seems the most plausible method to identify the compromised account. Although this method is not a hundred percent accurate, it works without the involvement of the human. The process is based on the identification of the writing style of the user which is not easily distinguished. Therefore, sophisticated machine learning algorithms that are efficient in finding hidden patterns from the data are used. Also, this method generalizes well on any type of data irrespective of the account type.

Chapter 3

Methodology

In this section, I describe my proposed approach and the features I use to train my system. I employ four feature extraction techniques and the combinations thereof to train my model. As my model design, I build two artificial neural network architectures - a Multilayer Perceptron (MLP) as my classification model and an MLP based siamese network for detecting text similarity. As my research is primarily focused on short text analysis, I only worked on the data from Twitter (Tweets).

3.1 Approach

As mentioned earlier, static data analysis is not enough to detect a compromised account. I need more advanced methods that could automate this process and works without human intervention. In this age, where I have a vast amount of data, machine learning algorithms seems like a most fitting approach. Machine learning based systems can identify patterns in unstructured data which otherwise be impossible for a human to find. Based on the learned patterns, they can calculate an approximate function that can accurately generalize most, if not all, the data. Besides that, the accuracy of this method is directly proportional to the amount of data and increases further as the data grows in size. To this end, I present an authorship attribution a.k.a text classification system using a neural network called an MLP. The intuition behind my approach is that if there is a substantial amount of short text data, I can extract sufficient features from that data. This can uniquely identify the writing style of an author. So, whenever there is an account breach, and it sends a new message, my model that is trained over the old messages of that account could predict that it is a compromised account.

Apart from classification, deep neural networks are also proficient in capturing latent representation from the text. I can use these representations to detect similarity. In the real-world scenario, one would have two messages from the same user which

could look very similar. This happens because the compromised account would try to mimic the writing style of the original user. After learning these hidden representations, the system can compare these messages and could tell if these are actually similar or not, i.e., they belong to the same or different authors. To this end, I implement siamese networks, using MLPs which I am also using for classification. The architecture of the MLPs are a little different from what I used for classification but I used the same feature extraction techniques to train my model.

Any machine learning requires numeric data as an input. Therefore, textual data is first converted in to fixed length integers or real numbers before feeding it to the model. This is known as feature extraction. Therefore, to generate numeric vectors from the data I employ two approaches - Term frequency-inverse document frequency (TF-IDF) and word embeddings [35]. Whereas TF-IDF assigns values (weights) to a word based on its relevancy in the document, word embeddings is the vector representation that captures context of the word. In word embeddings, words that are semantically similar are close to each other in the vector space. This is achieved by unsupervised training on the large corpus. The advantage with word embeddings is that it make use of transfer learning [56] and one could even use them on a smaller dataset by training the embeddings on the large text data like Wikipedia. After learning the embeddings, an n dimensional vector representation for each word is obtained from the corpus. It is 300 in my case. In the following sections I will explain what modified approach of the vanilla TF-IDF and word embedding I used.

Figure 3.1 shows an overview of my classification system.

- I extract the text from the dataset. The raw data from the Twitter also contains other information like id, date of creation and geolocation which is of no use for us. After extracting text from the data I learned embeddings from it.

- Next I prepossess the data for both cleaning it and reducing the vocabulary size.

- After that I extract four types of features from the text. For the first three features namely word n-grams, character n-grams and flexible patterns I used the TF-IDF approach as mentioned earlier.

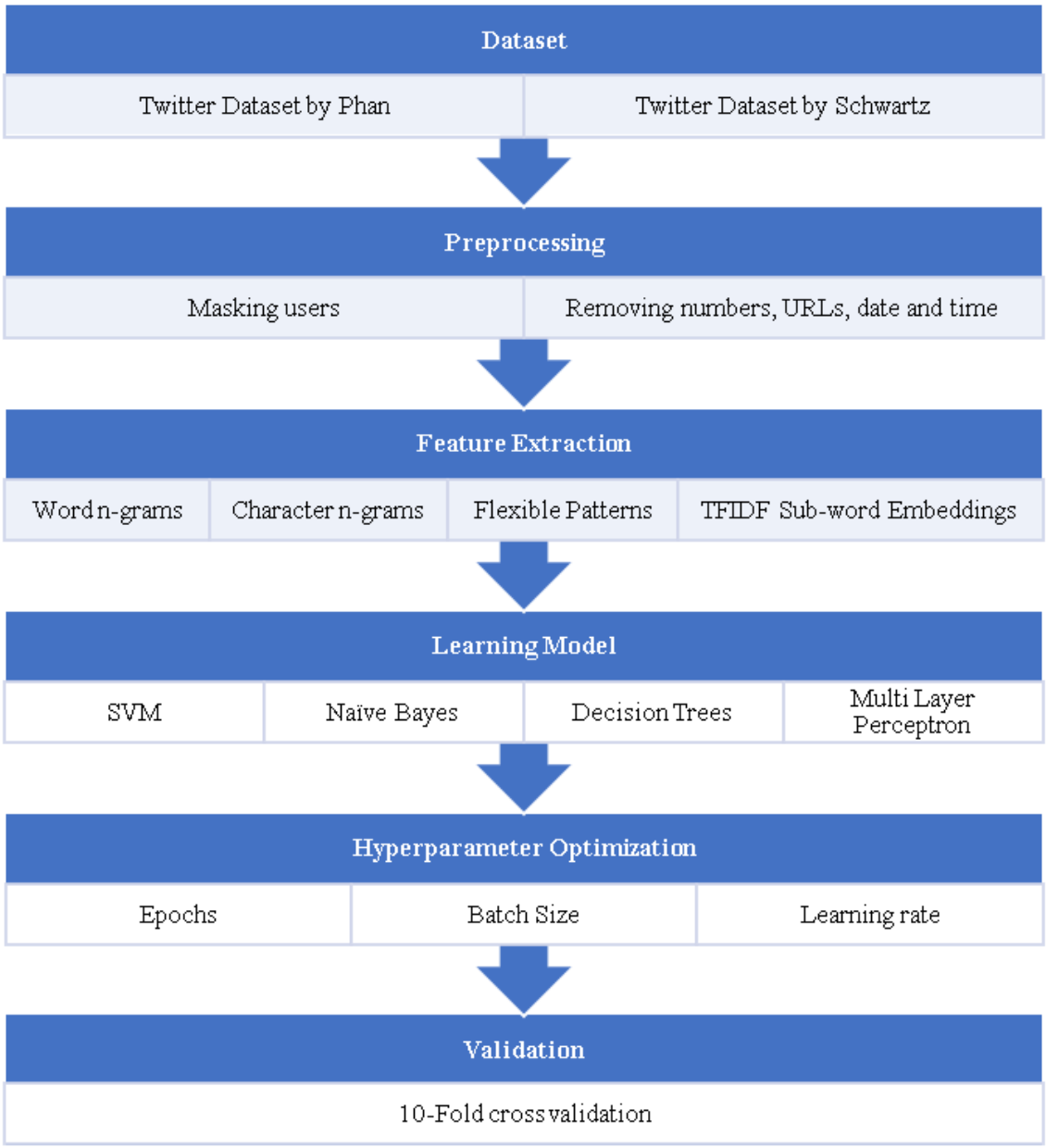

Figure 3.1: Overview of the classification model

- The output of the feature extraction step is used as an input for the learning models. I use four learning algorithms for the sake of comparison.
- I try optimize my neural network (MLP) by tuning the parameters and chose those which gave the best results.
- Lastly, I evaluate my model based using 10-fold cross validation.

In my similarity detection system everything remains as is except the fourth phase where I used siamese network as the learning model. Important thing to note here is that in both the cases - TF-IDF and word embeddings, the size of the feature vectors remains same for irrespective of the size of the document. This happens because the TF-IDF vectorizations gives a sparse representation of the data and the size of the input depends of the vocabulary size. In case of word embeddings, I get a 300-dimensional vector for each word in a tweet but I am taking weighted TF-IDF mean resulting in the final feature vector as the same size. I also showed how the learned embeddings could take advantage of the transfer learning on a different database

3.2 Algorithms

In this section, I define all the feature extraction techniques and the classification models I used for the classification. As shown in 3.1, I used three different classifiers apart from MLP. I also explain the statistical test I performed to test my results.

- TF-IDF
- Skip-Gram embeddings
- Multilayer Perceptron
- Support Vector Machines
- Naive Bayes
- Decision Trees
- Siamese Networks
- Student's t-test

3.2.1 TF-IDF

TF-IDF, which stands for term frequency-inverse document frequency, is the successor for the bag of words (BOW) approach. BOW assigns a count to a word based on the number of times it occurs in the document. This approach has a flaw as sometimes words like $'the', 'so', 'there'$ which are also called stop words appears the most. Even after removing these stop words, there are other words that have a higher frequency. BOW will assign these words, numbers, according to their frequency instead of how relevant they are in capturing the uniqueness of the text. TF-IDF handles this problem by also taking into account the number of times that word has appeared in the entire corpus. Hence, if a word appears in a single document and rarely appears in the entire corpus, it would have a higher count as it gives more information about the context of that document.

- Term Frequency (TF) - As the name suggests, it is the frequency of each word in a document. The word frequency is divided by the total number of words in document. Thus, TF value of a word is proportional to its occurrence in the document and inversely proportional to the number of words in the document.
- Inverse Document Frequency (IDF) - It is to counteract the importance of high occurring word in the corpus. Rare words have a higher IDF score compared to frequent words.

In a text document d, the frequency of term t with total number of words n, where $\widehat{\mathrm{N}}$ is the total number of documents D in the corpus, the TF-IDF is given by -

$$TF - IDF = \frac{t}{n} \times log \frac{N}{t \in D} \tag{3.1}$$

Let's take an example of these two sentence -

- *Document 1 - The boy calls the father*
- *Document 2 - The girl calls the mother*

Vocab	TF		IDF	TF - IDF	
	Document 1	Document 2		Document 1	Document 2
The	2/4	2/4	0	0	0
boy	1/4	0	log(2/1)	0.075	0
girl	0	1/4	log(2/1)	0	0.075
calls	1/4	1/4	0	0	0
the	2/4	2/4	0	0	0
father	1/4	0	log(2/1)	0.075	0
mother	0	1/4	log(2/1)	0	0.075

Table 3.1: TF-IDF of two documents

Table 3.1 shows how to calculate TF-IDF of the above sentences. The advantage with this feature extraction approach is that not only one don't have to worry about the stop words but the unique and context relevant word in the document will get higher weighting than others.

3.2.2 Skip-Gram Embeddings

Word embedding [35] is a semantic parsing technique used to create the vector representation of a text in a smaller dimensional space compared to the classic Bag of words approach [57]. The idea behind word embedding is that semantically similar words should be close to each other or I can say that words that appears in the same context would have the same vector value. That is, in an n-dimensional space, the angle between the similar words should be close to zero. For example, vector of word *budapest* would be closer to vector of word like $Hungary, city, country$ and *London* because they are used in the similar context. The similarity between these words can be measured by cosine of the angle between their vectors. There are two types of methods to achieve this; namely, Continuous Bag of Words (CBOW) and Skip-Gram methods [35].

- Continuous Bag of Words (CBOW) - It predicts the target words by taking the context word as input. The input can be a single word or a window of size n before and after the word.

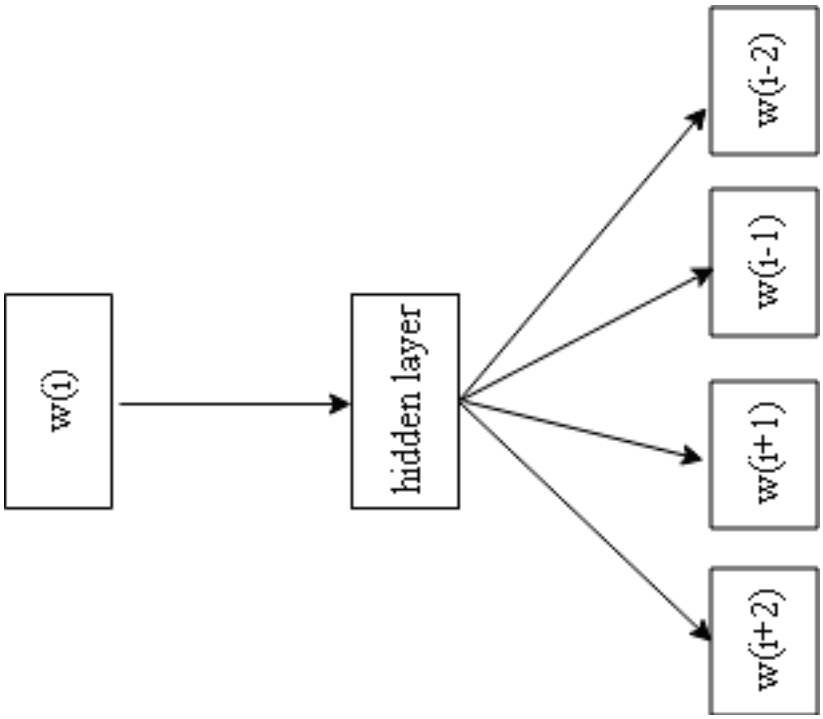

Figure 3.2: Skip-gram model

- Skip-gram - It is the opposite of CBOW model. Skip-gram predicts the probability of the context words using the target words. So, the outcome is the context of the target word by predicting n neighbouring words. Here n is a positive integer. Figure 3.2 shows the skip-gram model. Therefore, given a the words $w_1, w_2....w_k$, where n is the window size and V denotes the number of training words, the skip-gram model maximizes the average log probability which is given by equation 3.2 -

$$\frac{1}{V}\sum_{v=1}^{V}\sum_{j=-n,j\neq 0}^{n} logP(w_{v+j}|w_v) \tag{3.2}$$

Although, CBOW is a faster method, the reason for choosing Skip-gram is that it works better with the small amount of data. Also, I am dealing with tweets and the vocabulary used in them is sometimes improvised or misspelled; the skip-gram model works well to represent these rare words.

3.2.3 Multilayer Perceptron

Multi-Layer Perceptron (MLP) [58] is a supervised learning algorithm, which is a class of feed-forward artificial neural networks. An MLP consists of at least three layers of nodes: an input layer, a hidden layer and an output layer. The input is a single vector which is modified in series by a set of hidden layers. The hidden layers consist of neurons that provide an output value from applying a function to the input values from the previous layers. This function is essentially a matrix of weights and a bias. The neural network learns through small changes to these weights and bias using back propagation. Except for the input nodes, each node is a neuron that uses a nonlinear activation function. Fig. 3.3 shows the basic MLP with only one hidden layer. MLP is applied to data, usually unstructured, that is non-linear an cannot be classified using the linear perceptron. Training in MLP consists of adjusting the weights and the bias to minimize the error. Two of the most used methods to calculate the loss are -

- Mean Square Error - This is the performance measure of the regression model. It is the average of square difference between the actual value over the predicted value for all labels. In eq. 3.3, N represents the total samples having value y_i and the predicted value $y_{p,i}$ for the i^{th} label.

$$s(y, y_p) = \frac{1}{N} \sum_{i}^{i \in N} (y_i - y_{p,i})^2 \tag{3.3}$$

- Cross Entropy - It is the performance measure for the classification problem. In classification, neural network predicts the probability of each output label. Cross entropy loss value increases non-linearly with the increase in difference of the predicted score from the actual label value. For a predicted score y_p for samples N having the actual score of y is given by Eq. 3.4. That is why it is also called log loss.

$$s(y, y_p) = \sum_{i}^{i \in N} y_i log \frac{1}{y_{p,i}} \tag{3.4}$$

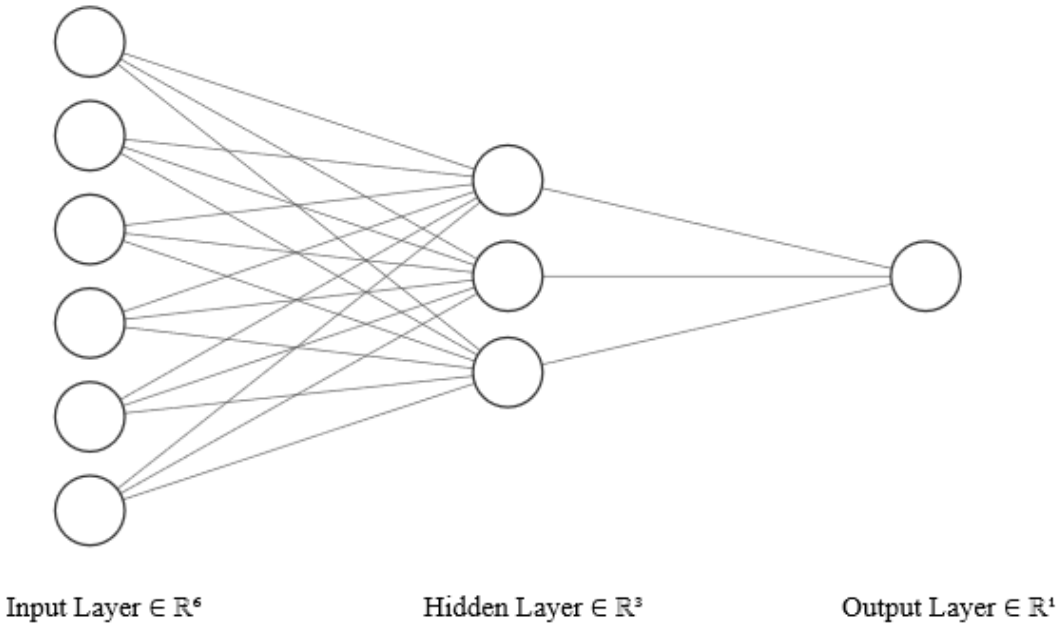

Figure 3.3: A simple MLP with input layer, one hidden layer and out layer

Author1	Author2	Author3	Author4
1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

Table 3.2: One hot encoding for four authors

For a multi-class problem like my work, categorical cross entropy loss is used which is a variation of cross entropy. In such problems, loss is calculated over the one hot encoded value of the positive classes as all the negative ones are zero. To better explain this imagine four authors $author1, author2, author3$ and $author4$. These authors are represented in the bits equals to the number of labels. As shown in table 3.2 the loss term for negative classes is zero.

Activation functions

Next, I talk about activation functions [59]. As I already discussed, the neural network is just a function approximater based on the weight matrix and the bias. This could result in the value of Y to exist between negative infinity to infinity. An activation function helps the neuron to decide for what value of Y it should get "fired". In my approach I used ReLU and TanH for my hidden layers and SoftMax for the output layer.

- Rectified Linear Unit (ReLU) - It is a linear function when the input is positive and zero otherwise. It is denoted by $f(x) = max(0, x)$. ReLU works well for most neural networks and it is faster to train as most of the negative inputs produces zero output.

- Hyperbolic Tangent Function (TanH) - It is another non-linear activation function whose value lies between -1 and 1. It is given by $f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$. The advantage of using TanH function is that it helps the back propagation process by producing output that centers around zero [59].

- SoftMax - This is used in the output layer as it predicts the probabilities of each class label. It is represented by $f(x) = \frac{e^{x_i}}{\sum_i e^{x_j}}$. The sum of the probabilities of the class labels are one with predicted class having the highest probability and is retrieved by using argmax.

Optimizer

I used Adaptive Moment Estimation (ADAM) [60] as my optimization algorithm to update network weights. ADAM differs from the conventional stochastic gradient descent as the method computes different learning rate for different parameters instead of maintaining the same one. According to Kingma and Ba [60], ADAM is the combination of two existing optimization algorithms -

- Root Mean Square Propagation (RMSProp) - This algorithm maintains separate learning rate for each parameter based on how quickly the average weight gradient is changing.

- Adaptive Gradient Algorithm (AdaGrad) - AdaGrad also have a different learning rate based on different parameters which makes it suitable for NLP problems having sparse gradients.

ADAM differs from RMSProp that it not change the learning rate based on average weight gradient of the mean but also the average weight gradient of the variance. It computes the exponential moving average of the gradients. Below is the list of four parameters while using ADAM optimizer. Important thing to note here is that I used the default value for these.

- learning rate - It is the magnitude by which weights are updated. The default value is 0.001.
- beta1 - The exponential decay rate for the first moment estimates having default as 0.9
- beta2 - The exponential decay rate for the second moment estimates with the default value 0.999
- epsilon - A constant having the default value as $1e-07$ to prevent divide by zero errors.

3.2.4 Other Machine Learning Algorithms Used

As shown in Figure 3.1, apart from MLP, three different machine learning algorithms are employed at the learning phase of the proposed system in this work. The following gives an overview of these algorithms, namely SVM, Naive Bayes, Decision Tree.

SVM

The Support Vector Machine (SVM) [28] classifier is a supervised learning algorithm for classification and regression analysis. In the literature, SVM has been employed in document classification. It is a discriminative classifier, which employs hyperplanes to separate data samples. The hyperplane is the decision boundary between two classes, which is obtained in such a way that it maximizes the distance between the elements of those classes. The optimal hyperplane is determined in the training

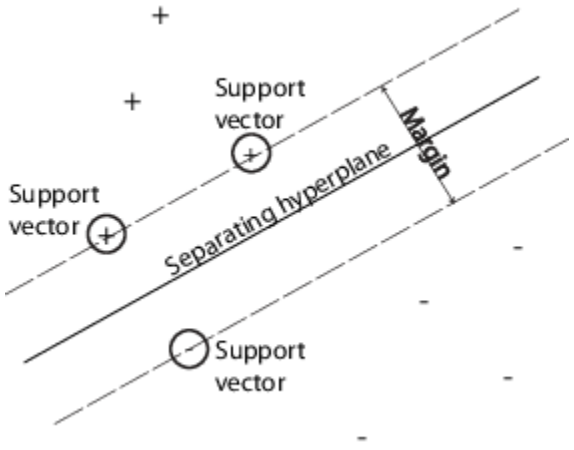

Figure 3.4: SVM with decision boundry [4]

process (supervised learning), and used to classify new data samples (testing). As shown in fig. 3.4, the data points that are at the separation boundary are called support vectors.

Naive Bayes

The Naive Bayes Classifier [27] is a supervised learning algorithm based on the Bayesian theorem with the assumption that the features are conditionally independent of each other. It is able to work with high dimensional input data. Naive Bayes is one of the most well known machine learning algorithms, which has been employed in numerous studies since the mid - 1990s. Despite its simplicity, the Naive Bayes algorithm is able to solve complicated tasks, such as text classification. Equation 3.5 shows that the features $x_i, ..., x_n$ are conditional independent.

$$P(y|x_i, .., x_n) = \frac{P(y)\prod_i^n P(x_i|y)}{P(x_i, ..., n)} \tag{3.5}$$

Decision Tree

Tree based classification algorithms are supervised learning algorithms, which employ a divide and conquer strategy to model the given data. A decision tree [29] is a tree structure where each decision node represents a feature and the leaf node represents the classification label. A decision tree splits the data samples based on the features

that provide the maximum information gained. In this research, Random Forest (ensemble classifier) of decision trees is used to implement this model.

I implemented SVM, Naive Bayes and Random Forest (Decision Trees) classifiers using the Scikit-Learn Python machine learning library [61]. As discussed earlier, the goal is to classify micro-text. In doing so, I aim to choose the most suitable classifier using the extracted features proposed in the previous section.

3.2.5 Siamese Networks

Siamese networks were used for one-shot image classification by Koch et al. [16]. They used Siamese networks which consist of two sub-CNNs with shared weights. Pairs of images from the same class and also from different classes were created in equal proportion for a single batch of training. They presented the image pairs to the CNNs in the Siamese network. The structure of the CNN was a series of convolution and pooling layers. The feature representations from the layers are flattened to a one-dimensional vector. The flattened vector represents the projection of the image onto a continuous vector space. They calculated the distance between the two vectors using the Euclidean distance [62]. This distance is fed into a fully connected layer and then finally, optimized using the cross-entropy loss function. The results showed that the network was correctly differentiating pairs that are from the same class and those that are from different classes.

I used contrastive [63] loss as my loss function. This loss function is based on the distance compared to previously mentioned losses such as MSE and is suitable for pair-wise learning. It learns parameters in such a way that semantically closer samples stays together. As shown in equation Eq. 3.6, f_i and f_j are representations of tweets i and j. y_{ij} is a boolean variable which is 1 when both tweets are from the same user, and 0 when the tweets are from different users. The contrastive loss requires the distance between embedded representations of tweets to be larger than a margin, m. The margin is decided in a way that limits the penalty to dissimilar samples.

$$Loss = \begin{cases} \frac{1}{2}{||f_i - f_j||_2}^2 & \text{if } y_{ij} = 1 \\ \frac{1}{2} * max(0, m - {||f_i - f_j||_2}^2) & if y_{ij} = 0 \end{cases} \tag{3.6}$$

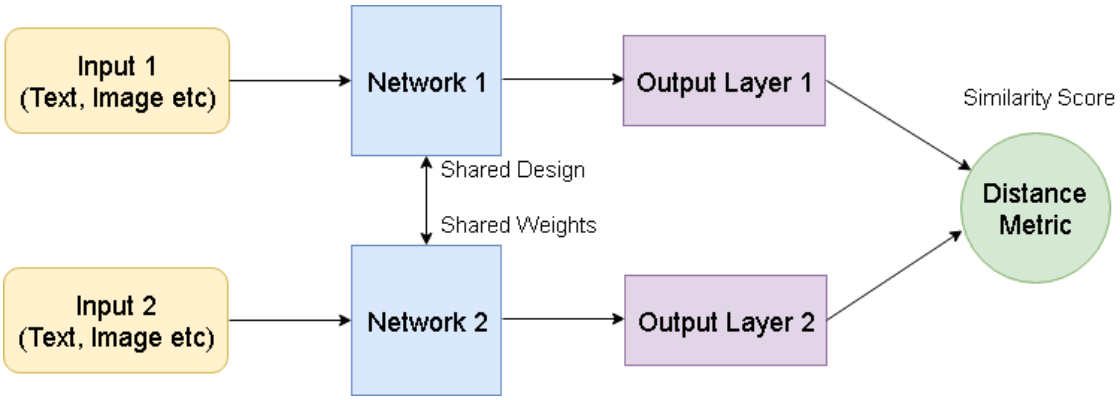

Figure 3.5: Typical Siamese network

3.2.6 Student's T-test

Student's T-test [64] is a statistical method that is used to check relationship between the mean of two normally distributed samples. In my similarity detection approach using Siamese networks, I used this to measure that there is a difference between the mean of two different calculated results and it doesn't happen due to experimental error which is also called the null hypothesis [65]. There are two values associated with the T-tests -

- T score - It is the inter group vs intra-group difference ratio. A large T score translates to a larger difference between the groups.
- P value - P value is the probability value which denotes the chance of occurring the different results. A P value of 0.05 denotes that there is only 5% chance that the results happened because of experimental errors or sampling problems.

In my experiments, I took accuracy groups based on different authors and tweets. Based on the T test, if I am getting the P value of less than 0.05, I concluded that the mean accuracy of any two of these independent groups rejects the null hypothesis.

3.3 Datasets

In this thesis, I work on two datasets one from Schwartz et al. [5] and the other one from [1]. As discussed earlier, I use the first dataset to train my skip gram embeddings. The purpose here is to train the word embedding model and used it on

```
@rationalbehavio yes, and there are so many plots - oy! I just want to see super-powered fights!
@DaisiesAndShit That is what I originally envisioned, since then I saw this from @SurlyAmy http://tweetphoto.com/56240011 now I'm spoiled
@nihiliad got it, thanks
Watching @stephenfry on Top Gear - wOOt!
@petrnotail mmm, thanks. I really needed a vice virtual vocktail.
@brucefp They shouldn't be selling them, they should be applying for the million dollar challenge
@chelseaepp @tricycles Clarkson appears to be driving around on a firing range being shot by tanks. This time real bullets. Obviously fake.
@TerilynnS Which DVD?
http://twitpic.com/tokd2 - Main St. USA xmas party
@40yroldatheist There's a soap opera or something on now, I vaguely understand whats happening.
I'm going to have to self-censor my response to that one.
@nikolasco Possibly, but the NYC thing is ubiquitous.
#VirtualDS is ON! http://bit.ly/QtO83
@NoisyAstronomer Yeah, apparently they arrest people taking pictures on the chesapeake bay bridge too.
@krelnik Aren't you going to be in DC next week? Planning to attend any of our D/S meets?
@xTexasBelle But they like to blow shit up - that's WAY more interesting than "going out"
@TheDarlaShow You can tell by the way they respond to you, sometimes if you know someone well you can see it in their eyes.
@misterperturbed Perhaps you'll luck out and only have to sit in a room for a day. I've been called 4 times and never had a trial.
Thanks for the #FollowFriday @mattincinci
@Petursey I liked it much! Although that episode seems rushed - we'll see if it smooths out next week.
```

Figure 3.6: Tweets Dataset from Schwartz et al. [5]

test data as well as the other dataset. Moreover, I use the same embeddings in my text classification and similarity model. In this section, I would discuss the datasets used by my system.

3.3.1 Twitter Corpus From Schwartz

This dataset is collected by Schwartz et al. [5]. It has over 7000 authors with every author having a thousand tweets each. The name of the authors are anonymized using a number. This is probably done for privacy reasons. There are ten files under every author having hundred tweets in each one. Fig. 3.6 shows an excerpt from the dataset. Apart from the condition that each tweet should have atleast three words, there is no other restriction on the tweet size or content. A Tweet can have following attributes -

- @user - These are called mentions, and it alerts the user whose name is included in the tweet.
- *#hashtags* - This is a part of the tweet that identify a topic and make it searchable on twitter. For e.g., during elections, tweets could include *#election*2019 hashtag.
- URL - This could be a website link.
- [*pic*] - This is an image reference followed by a link to that image.

3.3.2 Twitter Corpus From Phan

This corpus is originally collected by Yilu et al. [41]. It consists total of 970 authors with a varied number of tweets. Some users have over 3000 tweets while others have less than 1000. In this dataset the name of the original authors are captured as the purpose of this dataset was linking tweets to real world entities. Moreover, there is no restriction on the size of tweets. For the purpose of uniformity I followed the same tweet size as mentioned earlier. Fig. 3.7 shows the sample from of the authors. Each line has following components -

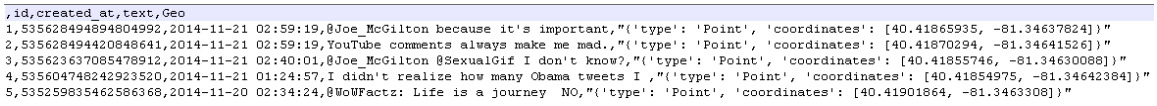

Figure 3.7: Tweets Dataset from Phan and Zincir-Heywood [1]

- id - A unique tweet id assigned by Twitter.
- created_at - The date at which this tweet is posted at the website.
- text - The actual body of the tweet.
- Geo - The co-ordinates of the location from which this tweet is posted.

As my focus is to identify user from their writing style, I only extract text data from the samples. It is important to not that location could better identify authors but as mentioned earlier a user could tweet from more than one location. Subsequently, not every tweet contains the location data. I chose same five user accounts as Phan and Zincir-Heywood [1] with each having 2000 tweets. Table 3.3 shows the name of selected five authors.

I can see in fig. 3.8 and 3.9 one of the most used phrases by *bradshaw19840* and *Ashley Nunn75* respectively. Even looking at the phrase construction I could say that *bradshaw19840* tweets are more opinionated and about his views on various sports whereas *terrymarvin63* are more about greetings. My system will find other such patterns in the data with the help of sophisticated feature extraction techniques.

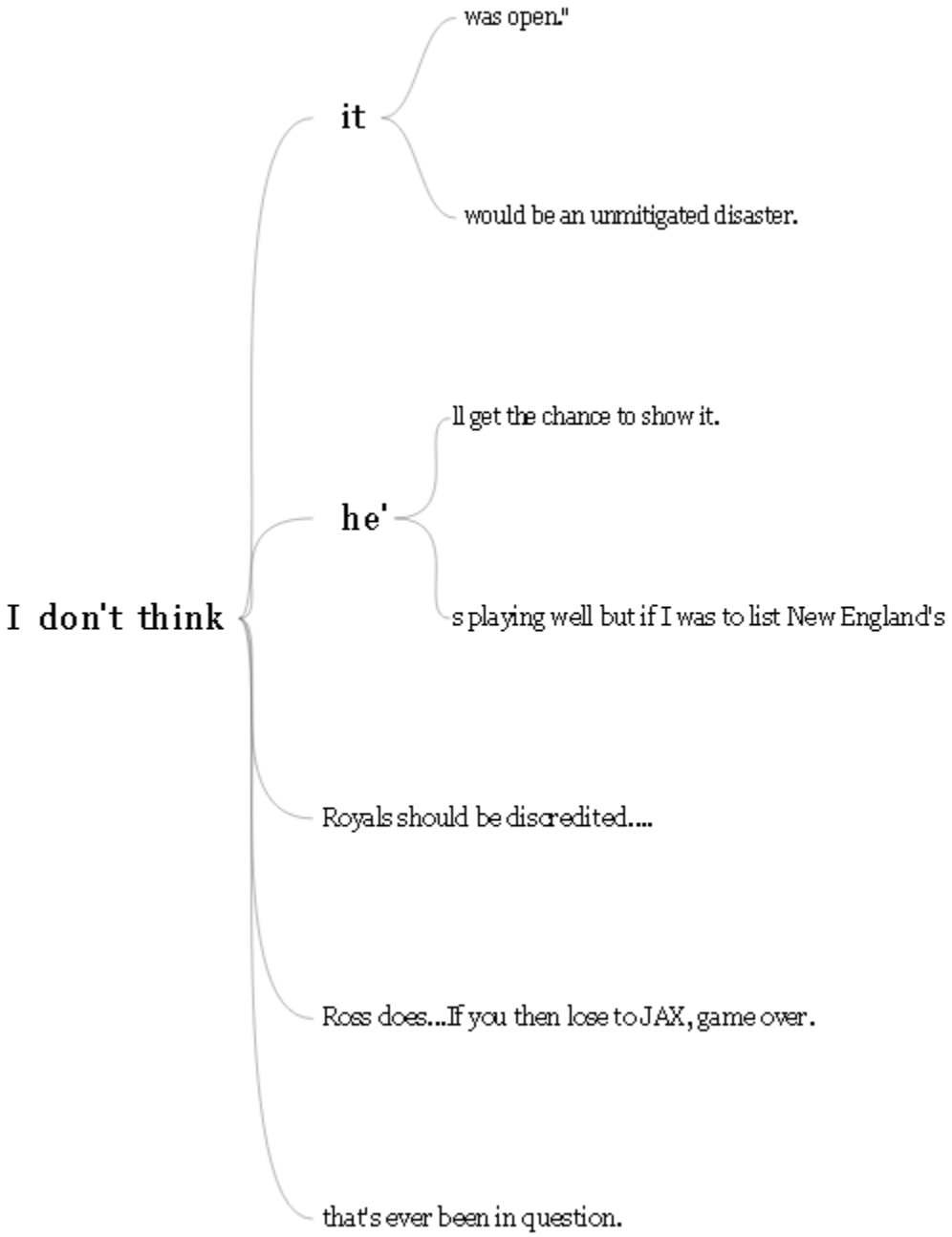

Figure 3.8: One of the Frequent phrases in *bradshaw1984* Tweets

Figure 3.9: One of the Frequent phrases in *terrymarvin63* Tweets

Authors
Ashley_Nunn75
bradshaw1984
shawnevans81
terrymarvin63
WhieRose65

Table 3.3: Authors selected from [1]

3.3.3 Dataset Preprocessing

Before applying any feature extraction technique or creating word embeddings I refined my data and made it suitable to better capture the writing style. Although, five users are selected from [1], I consider all 7026 authors from [5]. These are the changes I made to the corpus -

- I replace all mentions (@*user*) with the text *user*. This is important to prevent overfitting and make the system generalize the prediction on basis of writing style rather than on the names used by the author.
- Stemming to reduce the vocabulary size.
- All instances of date and time are replaced by the word *date* and *time*, respectively.
- I also replace all number and the URLs with the word *num* and *URL*, respectively.
- Punctuation are considered as valid tokens but a space is put between them and any other words.
- Apart from that I remove any special characters.
- Sometimes there are multiple spaces, tabs or newline character. I replace all of them with a single space character.

- Lastly, I mark the start of the message with the word *"BEGIN"* and the end of the message with the word *"END"*

3.4 Feature Extraction

There are a total of four features that are obtained from the text namely word n-grams, character n-grams, flexible patterns and word embeddings.

3.5 Word and Character N-grams

N-grams are consecutive sequences of words and characters in text which can potentially capture repeating phrases shown to be useful for authorship attribution. For example, in the phrase - *"This is a sentence"* -, a word unigram would be - *"This", "is", "a"* - and a word bigram would be *"this is", "is a", "a sentence"* - and so on. On the other hand, character n-grams are similar to word n-grams except, they are a sequence of characters. For instance, if I use the previous example, "This is a sentence", a character unigram would be - *"T", "h", "i", "s"* - and a character bigram would be - *"th", "hi", "is"*. I made following considerations while extracting n-grams -

- I choose word n-grams of length $2 \leqslant n \leqslant 5$
- For character n-grams - $3 \leqslant n \leqslant 4$
- I also take into consideration the maximum occurrence of the n-grams. For example, a pattern including prepositions (as shown in 1 below) or a masked username (as shown in 2 below) are very common in tweets.

  1. *for a*

  2. *<User> I*

  This is even more common in character n-grams. For this reason, I keep the upper limit of the n-grams to 0.9, which means I do not consider any word or character n-grams that appear in more than 90 percent of the documents. This further helps us identify the unique writing style of an author.

- I restrict each feature to a maximum of 50,000 in my experiments. To assign weights to the n-grams, I used TF-IDF. Moreover, I employed sub-linear scaling to the TF-IDF by taking the log of the term frequency, Eq.1.

$$wf_{t,d} = \begin{cases} 1 + logtf_{td}, & \text{if } tf_{td} = 1 \\ 0, & \text{otherwise} \end{cases} \tag{3.7}$$

3.6 Flexible Patterns

Next I extract flexible patterns from the text. Flexible patterns were introduced by Schwartz et al. [5]. The idea behind flexible patterns is that some users tend to use the same sequence of words in their writing style and only change a few keywords called content words (CW). For example, the flexible pattern of the following phrases -

1. *"I read the paper today"*
2. *"I drove the car yesterday"*

Flexible pattern - *"I CW the CW CW"*.

The words *read, paper, today, drove, car* and *yesterday* are replaced by the word *CW* based on the pre-defined condition. Therefore, masking some words, would create a pattern and separate the texts of some users from other users. I modify the existing approach [5] to make it suitable for smaller datasets and also to make it easier to implement.

According to Schwartz et al. [5] flexible patterns are a branch of word n-grams, where each word is either a high-frequency word or a content word and some words can be both. For a corpus size s if a word appears more than $10^{-4} \times s$ times it is a High-Frequency Word (HFW) and if it appears less than $10^{-3} \times s$ times it is a Content Word (CW). Also, the previous method takes into consideration that flexible patterns start/end with an HFW and there can be no consecutive HFWs. The problem with this approach is that it does not work with a smaller corpus. For a corpus where the value of s is 1000, no word is a CW. Thus, to overcome this limitation, Eq.2 is used to calculate the CW for a bigger corpus. Therefore in a corpus with a vocabulary size

n, the CW is calculated as the common log of the threshold for CW is selected as twice of log of s which are total number of words in the training dataset vocabulary. This method is based on the various experiments I carried out to choose the optimum number.

$$CW \leqslant 2log_{10}n \tag{3.8}$$

Algo 1 describes the algorithm for generating the flexible pattern of a tweet. First, the total number of tokens for a user (author) is calculated along with frequency of each token. Next, I replace words whose count is less than the threshold with a keyword "CW". These words are rare in occurrences and are called Content Words (CW). For example, the sentence "I have a bat" will converted to "I have a CW" if bat is the content word. After replacing all the CWs in the corpus, I then used the same approach for vectorization as I did for word n-grams by applying the TF-IDF scheme to those flexible n-grams before inputting them into the model.

3.7 TF-IDF Weighted Word Embeddings

Now, I talk about the word embeddings feature set I used the word embedding approach from Bojanowski et al. [14], which is called fastText. It is an extension of the original word embedding approach by Mikolov et al. [35]. FastText models can be trained to create a word embedding by making use of character level representations. These models learn a vector representation for a word by learning a representation for each of its character n-grams. Therefore, the overall word embedding is a weighted sum of the embeddings of these character n-grams. For example, for n=3, the vector for the word *hello* would be represented by a sum of trigrams: $< he, hel, ell, llo, lo >$ where $<$ and $>$ denote the beginning and end of a word.

Tweets contain several words, including, but not limited to, hashtags that are rare and sparsely occur in a corpus. To address this sparsity issue, each word is represented by the sum of the vector representations of its n-grams. Having a dictionary as size D and a word w belongs to this dictionary having $D_w \subset \{1, ..., D\}$, the set of n-grams appearing in w. Associating a vector representation of z_d to each n-gram d, the scoring function for w can be represented by Eq. 3.9:

Data: T = given tweet

C = corpus

V = vocabulary of corpus

Result: Find the flexible pattern for the tweet

$freq \leftarrow \{\}$

for *word in* V **do**
| $freq[word] \leftarrow$ *count of word in* C

end

$thresh \leftarrow 2 \log_{10} V$

$flex\_pattern \leftarrow []$

for *word in tweet* **do**

| **if** $freq[word] \leqslant thresh$ **then**
| | $flex\_pattern \leftarrow flex\_pattern + "CW"$
| **else**
| | $flex\_pattern \leftarrow flex\_pattern + word$

end

$flex\_pattern \leftarrow flexible\ pattern\ for\ tweet\ T$

Algorithm 1: Pseudo-code for the flexible pattern

$$s(w, c) = \sum_{d \in D_w} z_d^T v_c \tag{3.9}$$

For the value of n, I chose $2 \leqslant n \leqslant 6$. Empirically, instead of using pre-trained embedding, I train it on my corpus with a 300-dimension vector space. I construct the embedding using the Skip Gram approach which works better for a smaller amount of data and is preferable when there are a greater number of rare words in the corpus [5]. The weights of this network are learned giving as input a word in the middle of a sentence and use the surrounding words in a specific window size as supervisory signal. Figure 3.10 shows the created vector representation from the dataset.

Before using this as an input feature for my model, I weighted the embedding of each word in a text with the IDF value of that word. The resulting dimension is the same as the dimension of each word, which in my case is 300. Ultimately, I calculate the mean of the words to keep the dimensionality of the entire text the same as my vector space. Having a text of size T and words w where $T_w \subset \{1, ..., T\}$, the TF-IDF weighted embedding feature $f(w)$ of a word w_T is given by using Eq.4:

$$f(w) = \frac{\sum_{w \in Tw} idf(w_T) \times emb(w_T)}{T} \tag{3.10}$$

3.7.1 Embedding Visualization

I use t-distributed Stochastic Neighbor Embedding (t-SNE) [66] technique, which is a way to visualize high-dimensional data, into a low dimensional space, for visualization. Figure 3.10 shows the representation of 300-dimensional data in a two-dimensional space. Because of space constraints, I only visualized random 150 word vectors on the graph.

The graph shows the words which are used in the same context are grouped closer to each other. The idea behind this approach is that a particular author would use

the same combination of words in his/her tweets and therefore, they should be close to each other.

3.8 Summary

In this chapter, I discussed the methodology behind my proposed system and means to implement it. I gave an overview of my model which includes -

- List of algorithms I used in my system.
- An outline of the dataset and how to process that data before extracting features from it.
- Four feature extraction techniques, and the importance of each one of them.
- Various visual elements that provide a better interpretation of the proposed approach.

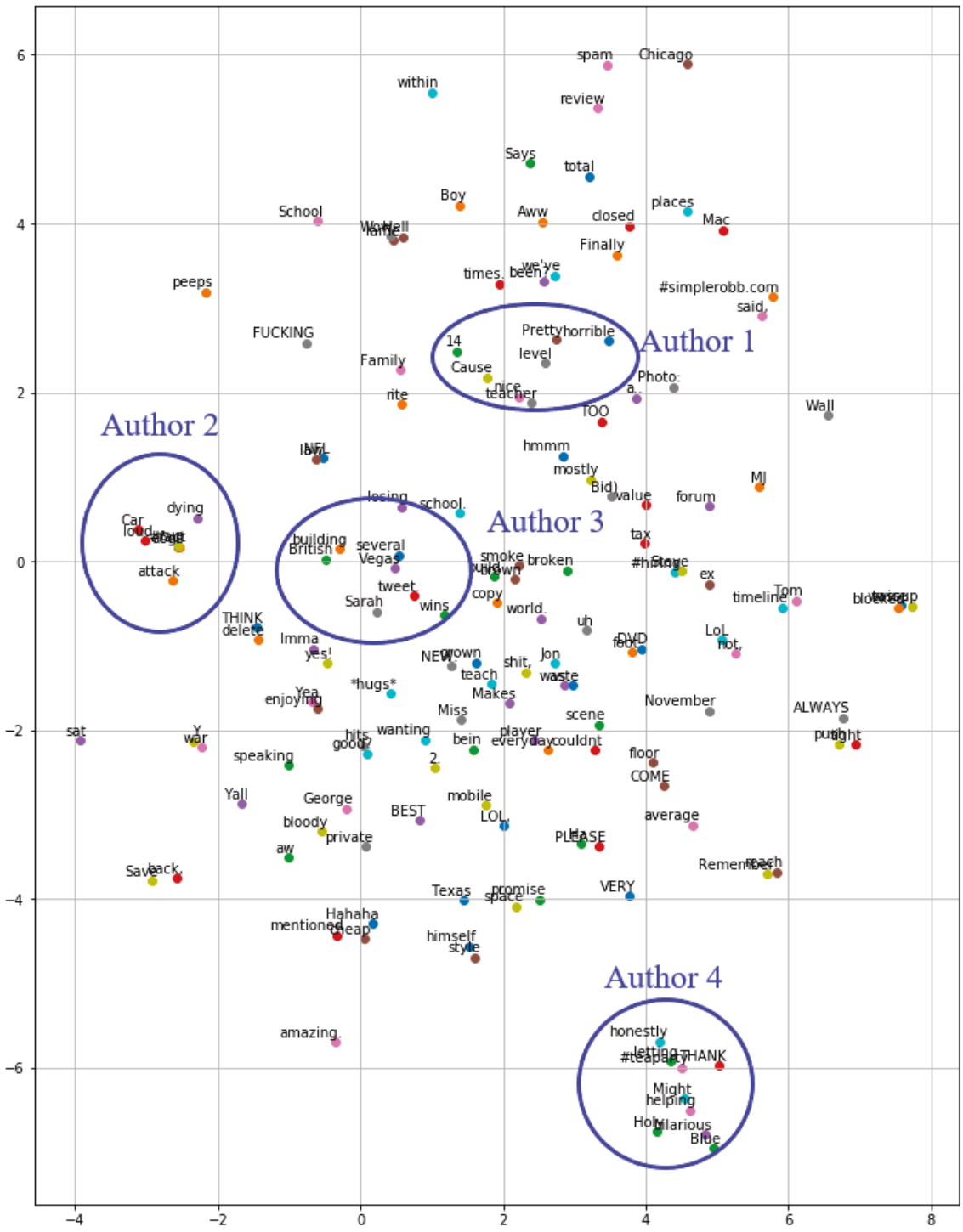

Figure 3.10: Skip-Gram word embeddings

Chapter 4

Experiments And Results

In this section, I will explain about the experiments in detail. This will include all the model specific techniques and parameters I chose for my system. Apart from that I also explain the results I get for both tweet classification and similarity approaches.

4.1 Tweet Classification

As discussed earlier, the goal is to classify micro-text. In doing so, I aim to choose the most suitable classifier using the extracted features proposed in the previous section. To this end, I employ the same small dataset used in Phan and Zincir-Heywood [1].

It is important to note here that before extracting features and feeding data into the MLP or any other classifier, I concatenated sets of two adjacent records. For example, the first text is combined with the second, the third with the fourth and so forth. Though the initial number of records remains the same, combining the original texts enables us to have a larger sequence, which achieves better accuracy even with the earlier approaches of feature extraction [5, 11]. The larger sequence also helps us to achieve more meaningful patterns which is not otherwise possible.

Table 4.1 shows the 10-fold cross validation accuracy of the four classifiers on that dataset for the 5 authors previously mentioned, where 2000 tweets are used for each author. In this case, TF-IDF word Embedding system (Table 4.1) achieves more than 99% 10-fold cross-validation accuracy which is 15% more than the best result reported in Phan and Zincir-Heywood [1].

MLP	SVM	Naive Bayes	Random Forest
.99	.979	.96	.82

Table 4.1: Accuracy for 5 users with 2000 tweets using different classifiers

Although, word embeddings are extracted both in this thesis and the work in Phan

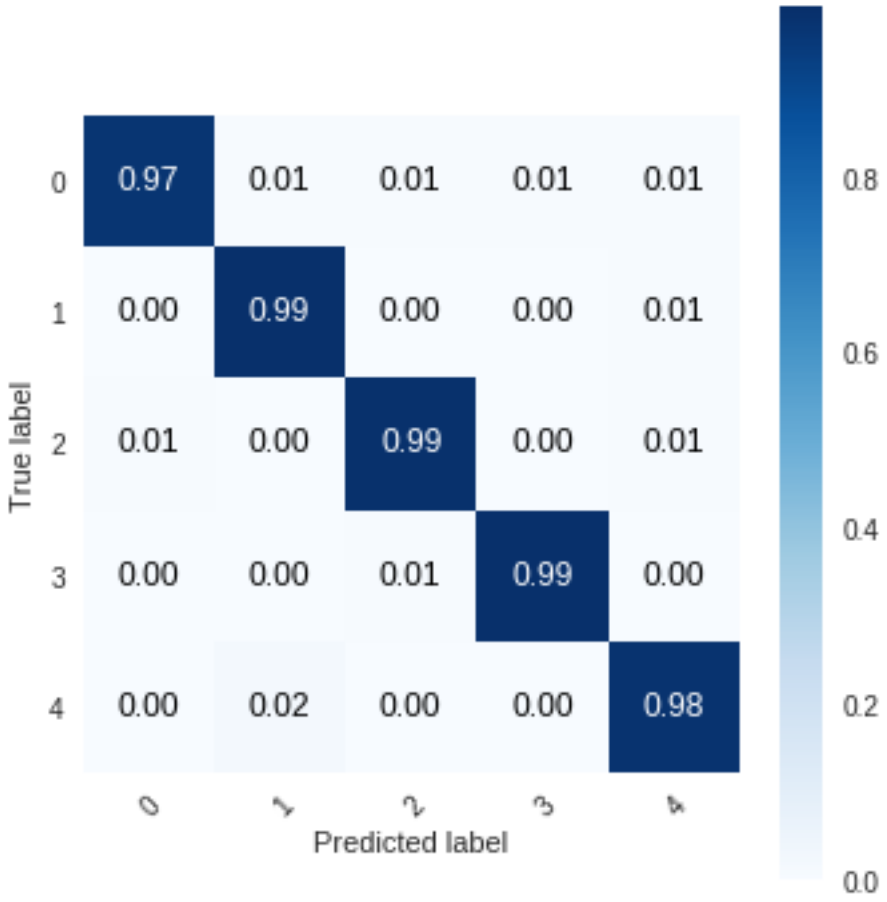

Figure 4.1: Confusion matrix for 5 authors

and Zincir-Heywood [1], my embeddings are formed using sub-words from a Twitter corpus. In Phan and Zincir-Heywood [1], the embeddings are formed using words from the Reuters Corpus Volume I (RCV1) [50]. Moreover, their word embeddings do not contain the TF-IDF weights. In this thesis, besides word embeddings, I also added three other feature extraction techniques - word n-grams, character n-grams and flexible patterns.

Moreover, Figure 4.1 (confusion matrix) demonstrates that these results are not biased to any specific author in the dataset used. Additionally, Table 4.2 shows how the 10-fold cross-validation accuracy improves the proposed MLP classifier as the different feature extraction techniques are used, where the first three columns show the accuracy of the combination of extraction techniques and the last column is the best test results given in Phan and Zincir-Heywood [1].

TFIDF Embedding	Flexible Patterns	Joining records	Phan2018
.99	.983	.972	.841

Table 4.2: Accuracy for 5 users with 2000 tweets each using proposed system with different feature sets

Given the above observations, MLP is chosen as the most suitable classifier for my research purpose: classifying the tweets according to their authors. Figure 4.2 presents the overview of the proposed MLP model. Here are the salient features of my model -

- The number of nodes in the input layer depend on the size of the input feature. *None* represents that the dimension is variable, and in this case, it is *86121*. The size would increase as the number of tweets increases to train the model.
- Then, there is a dense layer, which is a fully connected layer, where each input node is connected to each output node.
- One of the dense layer is a hidden layer of 1000 nodes.
- The Tanh activation function [67] is used for the hidden layer.
- The other dense layer is the output layer of 50 nodes, which corresponds to 50 authors. This can be changed depending on the number of authors (output classes).
- The SoftMax activation function [59] is used for the output layer.
- In between the two dense layers, there is a dropout layer for regularization.
- Furthermore, I have a 30% dropout.
- ADAM [60] is adapted as the optimization algorithm with a learning rate of 0.001.
- I divided the data into batches of 64.
- The model is trained for the total of 40 epochs.

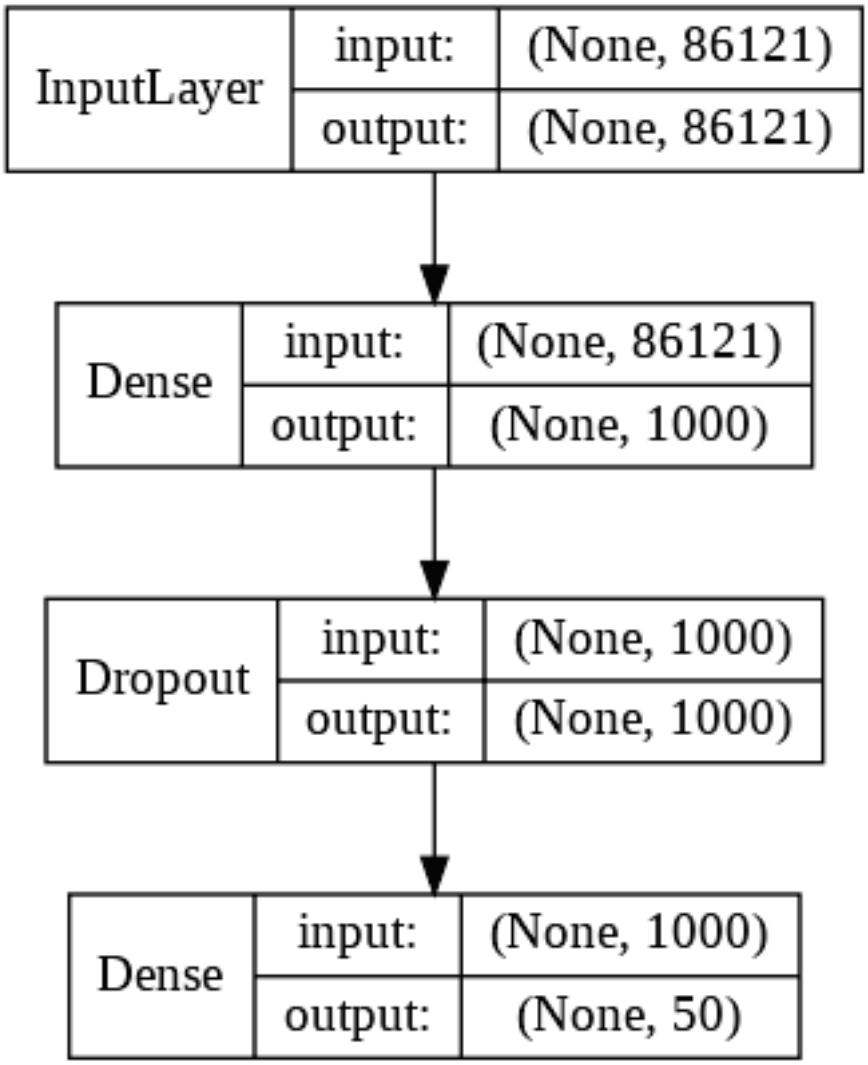

Figure 4.2: Overview of the proposed model using a Multi-Layer Perceptron

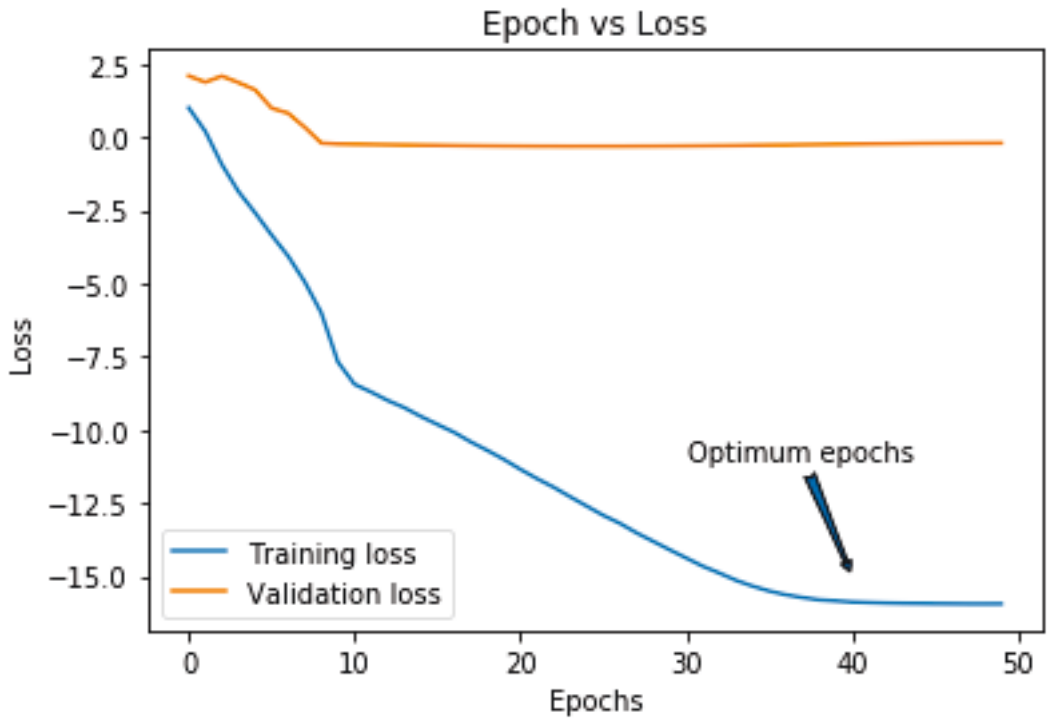

Figure 4.3: Epochs vs Loss for 50 epochs

Figure 4.3 and Figure 4.4 show the number of epochs vs loss, and the number of epochs vs accuracy graphs for training and validation data, respectively. These results show that both the loss and accuracy are optimal at around 40 epochs.

In the following experiments, I incrementally apply all three feature extraction techniques using the MLP classifier and observe the improvements compared to the previous research results (see Section 2). To this end, I first combine the subsequent tweets and apply the approach presented by Schwartz et al. [5], then I improve that method to calculate flexible patterns and their effect on the accuracy of the system. Last but not the least, I build weighted TF-IDF embeddings and combine them with the improved flexible patterns to form the combined set of features as input into the proposed MLP model. Ten-fold cross-validation approach is used to evaluate the performance of the proposed system. I evaluate the proposed system on the same dataset as used by Schwartz et al. [5], Shrestha et al. [11], where the dataset consisted of 50 authors, each having 1000 tweets.

Columns 1-3, in Table 4.3, shows the results of all three feature extraction techniques used with the MLP, and columns 4-7 represents the results from the previous

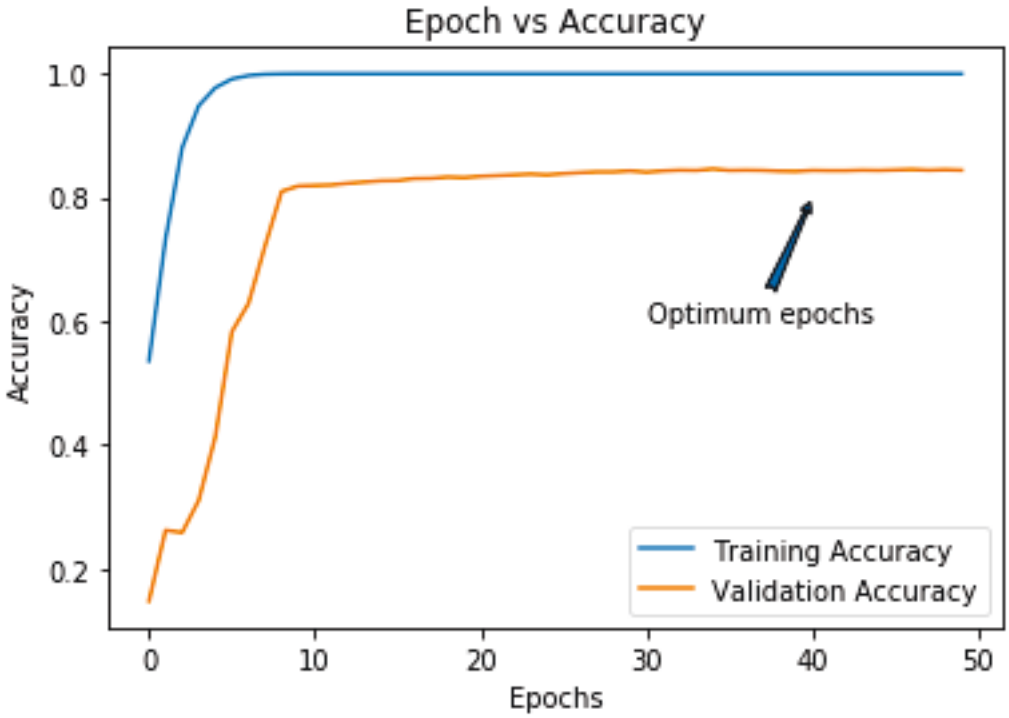

Figure 4.4: Epochs vs Accuracy for 50 epochs

TFIDF Emb.	Flex	Joining rec.	CNN-C	SCH	Char	LSTM-2
.852	.829	.81	.761	.712	.703	.645

Table 4.3: Accuracy for 50 users with 1000 tweets each

#tweets	Accuracy	Std dev
50	.589	.03
100	.648	.02
200	.730	.02
500	.791	.01
1000	.852	.01

Table 4.4: Accuracy and Standard deviation for 50 users from 1000 to 50 tweets

#	TFIDF Emb.	Mod. flex	Joining rec.	CNN-C	SCH	Char	LSTM-2
500	**.791**	.76	.748	.724	.672	.655	.597
200	**.730**	.694	.679	.665	.614	.585	.528
100	**.648**	.619	.608	.617	.565	.517	.438
50	**.589**	.563	.53	.562	.505	.466	.364

Table 4.5: Accuracy for 50 users from 500 to 50 tweets for each

works on the same dataset. My results are mutually inclusive, and the results are build upon combining all the feature extraction techniques. For example, I use the weighted TF-IDF embeddings in combination with the flexible patterns. Table 4.4 represents the mean accuracy and standard deviation for different number of tweets keeping the number of authors as 50.

- **Shrestha (CNN-C):** CNN-C, shown in Table 4.3, represents the best result obtained by Shrestha et al. [11] where a convolutional neural network architecture is proposed using character n-grams, specifically unigrams and bigrams as input. The convolutional model use is a three-level architecture with the input layer as the character embedding layer, a convolutional module, and finally, a dense layer with a Softmax activation function for classification. The unigram model performed well on the smaller dataset. Alternatively, the bigram model had better accuracy on the bigger dataset.

- **Schwartz (SCH):** SCH, shown in Table 4.3, represents the best result obtained by Schwartz et al. [5]. They used a linear SVM for classification and their model

was a combination of word and character n-grams along with a new feature set called flexible patterns (see section 3.3), which is modified and used in my system as well.

- **Char:** Char, shown in Table 4.3, represents one of the systems used by Shrestha et al. [11] in which they compared the performance of their system based on the earlier character n-gram approaches [5, 30] and proposed a logistic regression model that employed character n-grams of sizes two to four.

- **LSTM-2:** LSTM-2, shown in Table 4.3, represents the state-of-the-art LSTM model based on the success of previous implementations [68, 69]. This model was also used with bigrams as input to evaluate the performance of those systems, with respect to other models on the same dataset.

Introducing the concatenation of the consecutive records enables the accuracy of the proposed system to be improved by 5% compared to the previous approaches. Then applying the flexible patterns, further improves the accuracy by approximately 2%. Finally, implementing the weighted TF-IDF word embedding, and combining it with all the other features increases the accuracy of the proposed system to approximately 85%, Table 4.3.

In Table 4.5, I also compare the proposed system's accuracy to other approaches as I reduced the number of tweets from 500 to 50 for each author (50). After reducing the number of tweets, the proposed system still outperformed all the other previous approaches and performs well even when the dataset became as small as 50 tweets per author.

MLP vs SVM

Although, I already compared the SVM and MLP on the dataset used in Phan and Zincir-Heywood [1], the accuracy does not represent a clear difference between the two approaches as there are only 5 authors. So, in table 4.6, I compare the accuracy and time (computational cost) taken by SVM and MLP for different number of tweets where the number of authors is 50. The accuracy is better in all cases but the time is better for SVM only when there are less number of tweets. As the data increases,

there is a steep rise in the time taken by SVM compared to MLP. Figure 4.5 and 4.6 show the accuracy and time taken by the two models, respectively.

#tweets	Accuracy		Total time (sec)	
	MLP	SVM	MLP	SVM
100	.648	.40	28.65	24.65
200	.730	.49	76.38	75.41
500	.791	.57	321.03	351.24
1000	.852	.68	600.68	1112.22

Table 4.6: Accuracy and Time taken for 50 User for MLP and SVM

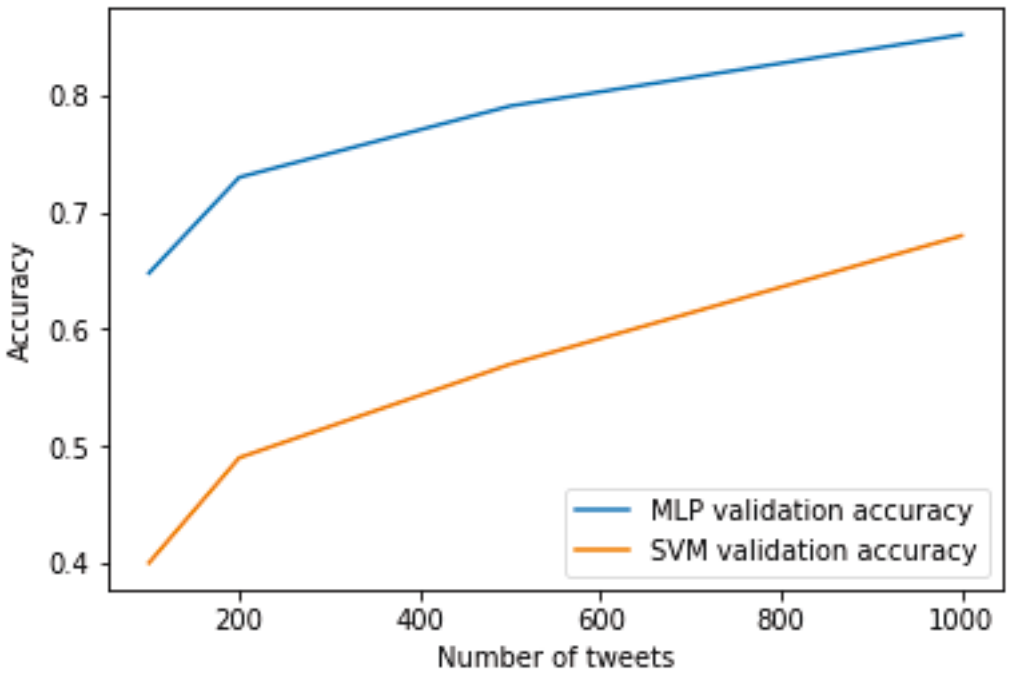

Figure 4.5: Number of tweets vs Accuracy for MLP and SVM

4.2 Tweet Similarity

The proposed system is trained for 200 epochs with a patience number of 50. For feature extraction, I combine the word, character and flexible n-grams in a single stack and consider this as one input. In comparison to this, I also consider just the

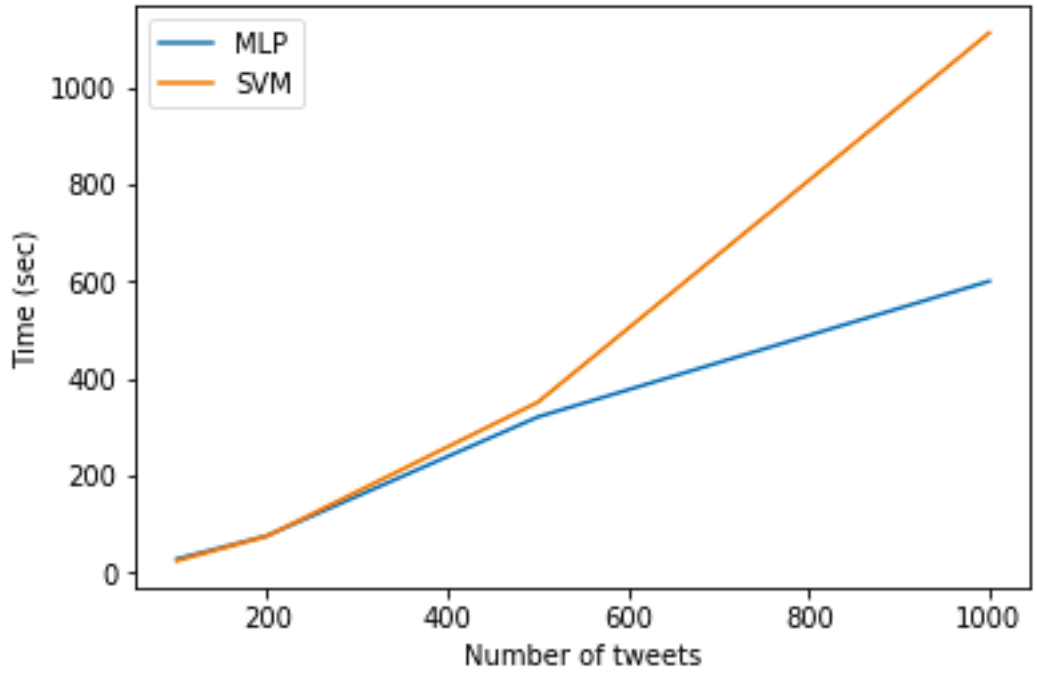

Figure 4.6: Number of tweets vs Time Taken for MLP and SVM

tf-idf weighted word embedding feature set as the input. The word embeddings have a maximum of 100 dimensions as I am taking the mean of every word in the dataset.

The training process is divided into two phases. In phase 1, the total number of users is 50 and the number of tweets ranges from 10 to 100. This is done to compare the aforementioned extraction techniques. In phase 2, the total number of tweets is 100 and the number of users ranges from 10 to 100 to test whether the performance of the model is impervious to the number of users.

It should be noted here that 90% of the dataset is used for training and 10% for testing. While training, I combine all the tweets from a particular user and when I pair them with the same user, the label is 1. On the other hand, if the tweets from the same user are combined with a randomly selected tweet from any other user, then the label is 0. I keep an equal number of positive and negative samples to keep the training set balanced. Although, after every epoch, I include more random samples to make my network more robust [16].

Fig 4.7 shows the architecture of the Siamese network containing two branches of a MLP. The MLP has four fully connected layers, where three of them have 128 units

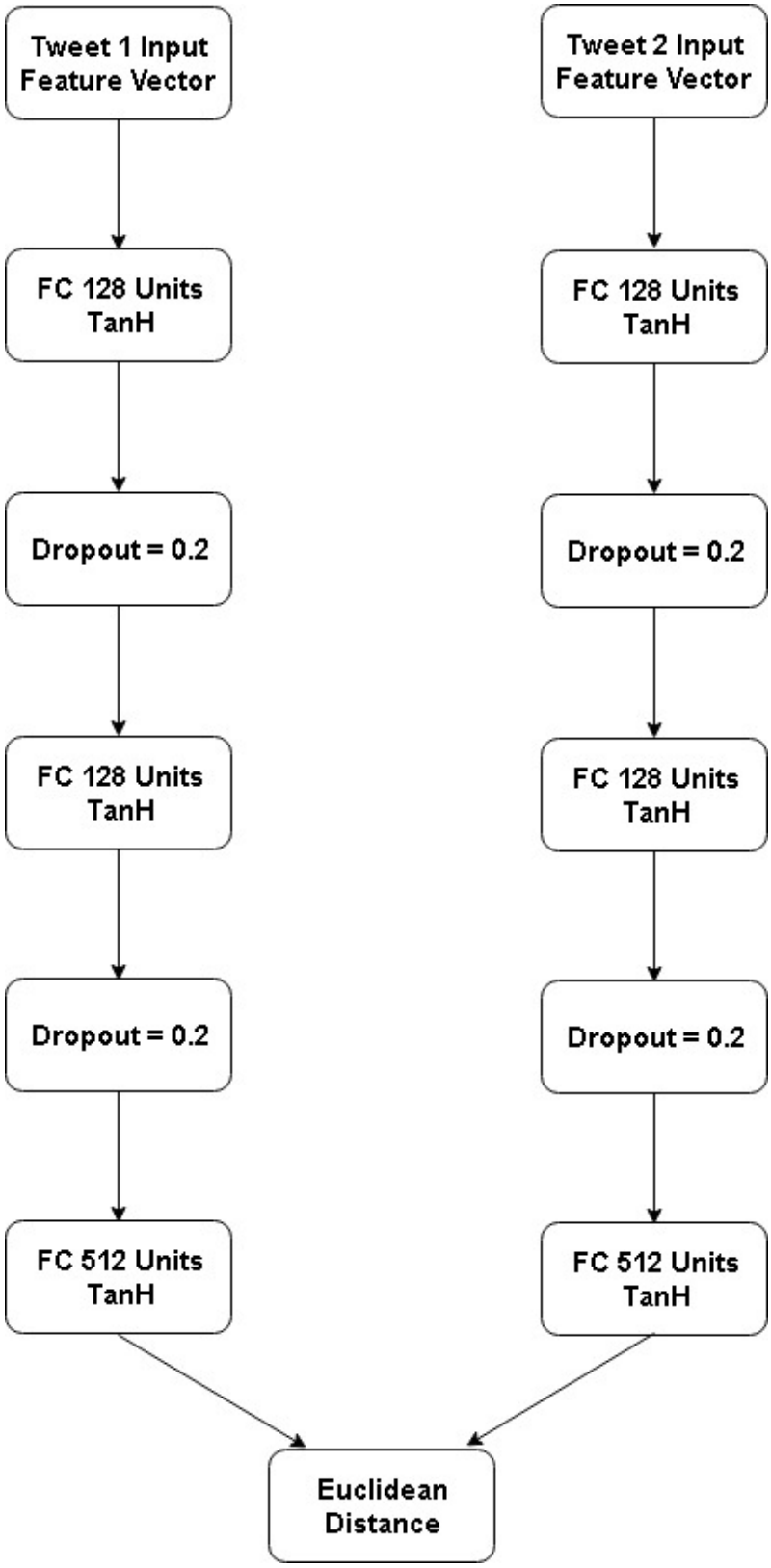

Figure 4.7: Siamese network model for compromised tweet learning.

#tweets	Word, Char and Flexible patterns		FastText Embeddings		p-value
	mean	std dev	mean	Std dev	
10	.57	.03	.60	.04	0.14
20	.65	.02	.70	.05	0.02
30	.66	.02	.69	.05	0.06
50	.66	.01	.71	.03	0.002
100	.71	.01	.73	.04	0.12

Table 4.7: Accuracy for 50 users as the number of tweets changes.

with Tanh activation function, while the last layer having 512 units. I use dropout layers with rate 0.1 for regularization. The last layer is a lambda layer which uses output from both branches as input and calculates the euclidean distance between the two representations. I pass the combined representation of users into the left network and the user tweet that is tested for compromised detection into the right. I use contrastive loss with a margin of 1. I use ADAM optimizer [60] with a learning rate of 0.001.

Fig 4.8 shows the test accuracy as the number of tweets changes for 50 users using Siamese networks. I employ two types of features as input to the Siamese network. First, I vectorize the tweets using word n-grams, character n-grams and flexible patterns. Secondly, I use the fastText word embedding tweets with a dimension of 100. I train both models 10 times and calculate the mean test accuracy as shown in Fig 4.8. The mean test accuracy increases as the number of tweets increases.

A student t-test between test accuracies for word, char and flexible pattern models with 10 tweets and 20 tweets gives a p-value of $2.5 \times e^{-5}$. This p-value is less than 0.05, showing that the I can reject the null hypothesis and data have been drawn from different distributions. Therefore, the test accuracy for 20 tweets is greater than 10 tweets. Although, there is no major difference between the test accuracies of 20, 30 and 50 tweets. However, when I increase the number of tweets to 100, I observe an increase in the mean test accuracy to 71% (1% std). A student t-test between test accuracies of 50 tweets and 100 tweets gives p-value of $2.8 \times e^{-10}$, rejecting the null-hypothesis. This shows that there is a difference between the test accuracies for

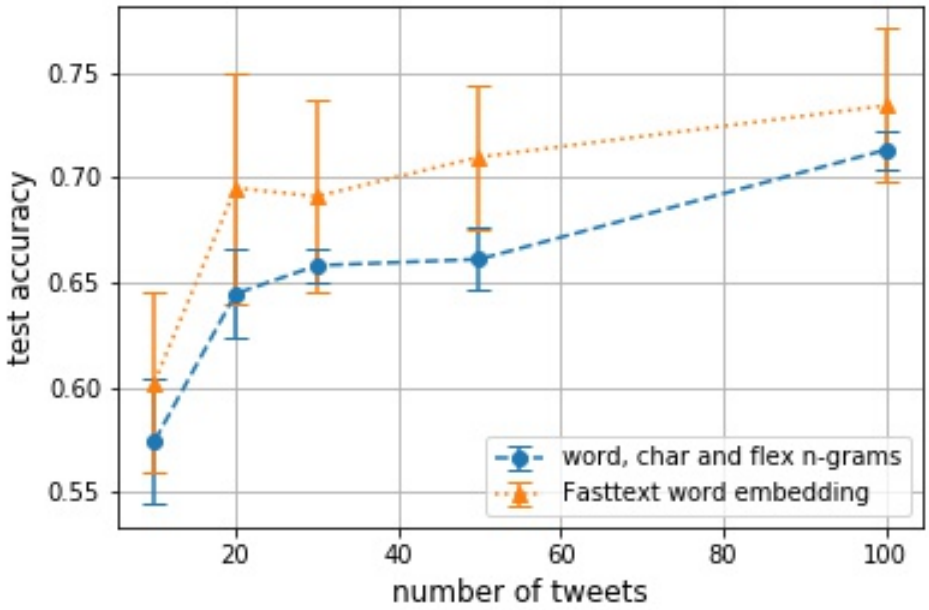

Figure 4.8: Test accuracy vs the number of tweets.

50 and 100 tweets.

Table 4.7 shows a comparison between test accuracies for word, char and flexible pattern model and fastText word embedding model. The test accuracies are similar for 10,30 and 100 tweets with p-value greater than 0.05. However, fastText word embedding model performs better with 20 and 50 tweets. The fastText model input has a lower dimension than the word, char and flexible pattern models.

Fig 4.9 shows the test accuracy as the number of users (each trained with 100 tweets) changes using Siamese networks. I use the fastText word embedding of tweets with a dimension of 100 as input to the Siamese networks. I ran the training process 10 times and calculated the mean test accuracy as shown in Table 4.8. The increase in the number of users does not impact the performance of the Siamese networks. The mean test accuracy is similar for 10, 20 and 30 user. The test accuracies of 50 and 100 authors exceed the test accuracy of 30 users. A student t-test between test accuracies with 50 tweets and 30 tweets gives a p-value of 0.001. This p-value is less than 0.05, showing that I can reject the null-hypothesis and data have been drawn from different distributions. Therefore, the test accuracy for 50 tweets is greater than the accuracy for 30 tweets.

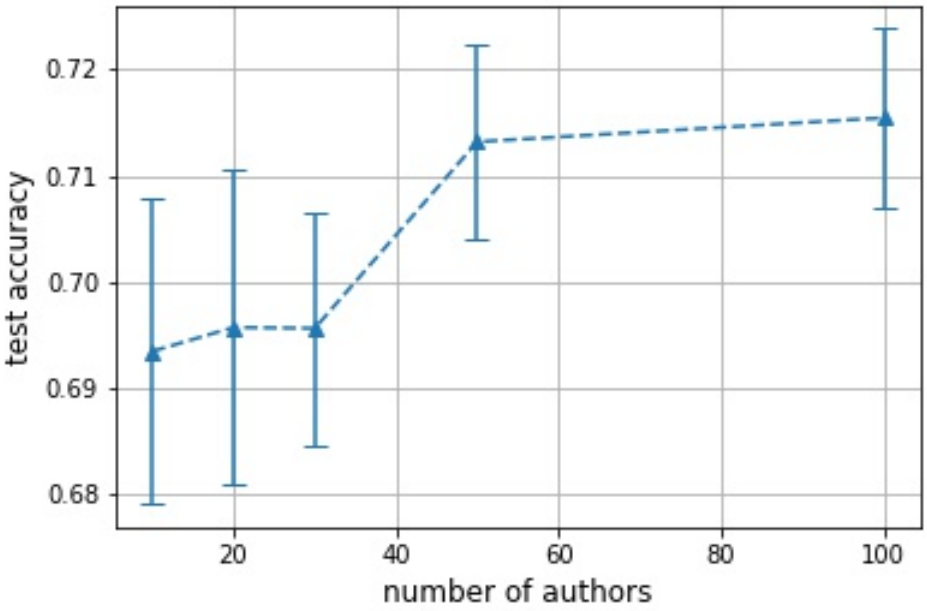

Figure 4.9: Test accuracy vs the number of users (authors)

#authors	FastText Embeddings	
	mean	Std dev
10	.69	.01
20	.69	.01
30	.69	.01
50	.71	.01
100	.72	.01

Table 4.8: Accuracy for different users, each trained with 100 tweets.

Chapter 5

Conclusion

In this research, I implemented a detection system for identifying users based on their writing styles. To this end, I employed two models, an MLP for classification of messages and a Siamese networks that identify text similarity. I also explored four feature extraction techniques. Apart from word n-grams and character n-grams, I introduced a feature extraction technique that was based on a word embedding model, namely sub-word embeddings, weighted by TF-IDF for short text messages. Additionally, I worked on modifying and improving the existing implementation of flexible patterns and proposed a neural network architecture that makes use of a combination of these features to perform authorship attribution based on writing styles of authors (users). To evaluate my model, I used two Twitter datasets: one from Schwartz et al. [5] and the other one from [1]. The salient features of my system are -

- It is trained using one dataset and the transfer learning can be applied to other corpa.
- Since I trained my embeddings from scratch my system performs equally well, irrespective of the language.
- It works exceptionally well for short texts and my model outperformed all the existing systems based on similar testing criteria and using the same datasets.
- It can also learn latent representation from the data and gives good results even when there is limited data.

For siamese network approach, I observed that it performs well as the number of tweets increases. The mean test accuracy improves as the number of tweets increases, Fig. 4.8. Moreover, the test accuracy does not degrade as the number of users increase as shown in Fig. 4.9. I also perform a comparison between the fastText

word embedding model and the model that used word, character and flex n-grams as the feature representations, Table 4.7. The test accuracies are similar across both models except the fastText model has a lower dimensional input resulting in a lower computational cost.

With the success of sub-word embeddings, potential future work directions are: working with new word embedding techniques such as Elmo [70] and Bert [71], which are based on the context of a word in a corpus. Alongside that, I am also interested in improving my neural network architecture using transfer learning models such as ULMFit. I also plan to use auto-encoder models to create another latent representation of the tweets before passing them as input to a Siamese neural network.

Bibliography

[1] Tien D Phan and Nur Zincir-Heywood. User identification via neural network based language models. *International Journal of Network Management*, page https://doi.org/10.1002/nem.2049, 2018.

[2] HP Enterprise Security. Micro focus security on twitter: "which types of cyber security attacks are most common to your business? http://t.co/4hhrt5nr1d #hp #infosec http://t.co/psmlji4co7".

[3] Statista. U.s. data breaches and exposed records 2018 — statistic. `https://www.statista.com/markets/424/topic/1065/cyber-crime/`. (Accessed on 06/08/2019).

[4] Support vector machines for binary classification - matlab. `https://www.mathworks.com/help/stats/support-vector-machines-for-binary-classification.html`. (Accessed on 06/21/2019).

[5] Roy Schwartz, Oren Tsur, Ari Rappoport, and Moshe Koppel. Authorship attribution of micro-messages. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1880–1891, 2013.

[6] Shawn Denbow and Jesse Hertz. pest control: taming the rats. *Matasano Security, www. steptoecyberblog. com/files/2012/11/PEST-CONTROL1. pdf*, 2012.

[7] Carl E Landwehr, Alan R Bull, John P McDermott, and William S Choi. A taxonomy of computer program security flaws, with examples. Technical report, NAVAL RESEARCH LAB WASHINGTON DC, 1993.

[8] Mike Kestemont, Michael Tschuggnall, Efstathios Stamatatos, Walter Daelemans, Günther Specht, Benno Stein, and Martin Potthast. Overview of the author identification task at pan-2018: cross-domain authorship attribution and style change detection. In *Working Notes Papers of the CLEF 2018 Evaluation Labs. Avignon, France, September 10-14, 2018/Cappellato, Linda [edit.]; et al.*, pages 1–25, 2018.

[9] Efstathios Stamatatos. Authorship attribution using text distortion. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1138–1149, 2017.

[10] Pan @ clef 2018 - author identification. `https://pan.webis.de/clef18/pan18-web/author-identification.html`. (Accessed on 05/27/2019).

[11] Prasha Shrestha, Sebastian Sierra, Fabio Gonzalez, Manuel Montes, Paolo Rosso, and Thamar Solorio. Convolutional neural networks for authorship attribution of short texts. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, volume 2, pages 669–674, 2017.

[12] Tie-Yun Qian, Bing Liu, Qing Li, and Jianfeng Si. Review authorship attribution in a similarity space. *Journal of Computer Science and Technology*, 30(1):200–213, 2015.

[13] Nick Gillian. Mlp nickgillianwiki. `http://www.nickgillian.com/wiki/pmwiki.php/GRT/MLP`, 2014. (Accessed on 04/25/2019).

[14] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

[15] Mihir Joshi and Nur Zincir-Heywood. Classification of micro-texts using subword embeddings. Accepted for publication in RANLP 2019.

[16] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML Deep Learning Workshop*, volume 2, 2015.

[17] Booz Allen Hamilton. Virtual identity. `http://federalnewsnetwork.com/wp-content/uploads/pdfs/VirtualIdentity.pdf`. (Accessed on 06/13/2019).

[18] Ilker Nadi Bozkurt, Ozgur Baglioglu, and Erkan Uyar. Authorship attribution. In *2007 22nd international symposium on computer and information sciences*, pages 1–5. IEEE, 2007.

[19] Alex Johnson. Hbo investigating hack of its twitter accounts. `https://www.nbcnews.com/tech/security/hbo-investigating-hack-its-twitteraccounts-n793391`, 2017. (Accessed on 06/13/2019).

[20] Nicole Perlroth. All 3 billion yahoo accounts were affected by 2013 attack - the new york times. `https://www.nytimes.com/2017/10/03/technology/yahoo-hack-3-billion-users.html`, 2017. (Accessed on 06/13/2019).

[21] David Sanger. Marriott data breach is traced to chinese hackers as u.s. readies crackdown on beijing - the new york times. `https://www.nytimes.com/2018/12/11/us/politics/trump-china-trade.html`, 2018. (Accessed on 06/13/2019).

[22] Sohini Mitter. Truecaller user data for sale? company says investigating "illegal activity". `https://yourstory.com/2019/05/truecaller-data-sale-company-launches-investigation`, 2019. (Accessed on 06/13/2019).

[23] Jason Silverstein. Facebook data breach: Hundreds of millions of records exposed on amazon server, according to upguard cybersecurity research firm - cbs news. https://www.cbsnews.com/news/millions-facebook-user-records-exposed-amazon-cloud-server/, 2019. (Accessed on 06/13/2019).

[24] Eric Lin, John Aycock, and Mohammad Mannan. Lightweight client-side methods for detecting email forgery. In *International Workshop on Information Security Applications*, pages 254–269. Springer, 2012.

[25] Carlo Schäfer. Detection of compromised email accounts used by a spam botnet with country counting and theoretical geographical travelling speed extracted from metadata. In *2014 IEEE International Symposium on Software Reliability Engineering Workshops*, pages 329–334. IEEE, 2014.

[26] Kim Luyckx and Walter Daelemans. The effect of author set size and data size in authorship attribution. *Literary and linguistic Computing*, 26(1):35–55, 2011.

[27] TIBCO. Naive bayes classifier. http://www.statsoft.com/textbook/naive-bayes-classifier, 2019. (Accessed on 04/26/2019).

[28] Marti A. Hearst, Susan T Dumais, Edgar Osuna, John Platt, and Bernhard Scholkopf. Support vector machines. *IEEE Intelligent Systems and their applications*, 13(4):18–28, 1998.

[29] Lior Rokach and Oded Maimon. *Decision Trees*, volume 6, pages 165–192. 01 2005. doi: 10.1007/0-387-25465-X_9.

[30] Robert Layton, Paul Watters, and Richard Dazeley. Authorship attribution for twitter in 140 characters or less. In *2010 Second Cybercrime and Trustworthy Computing Workshop*, pages 1–8. IEEE, 2010.

[31] Vlado Kešelj, Fuchun Peng, Nick Cercone, and Calvin Thomas. N-gram-based author profiles for authorship attribution. In *Proceedings of the conference pacific association for computational linguistics, PACLING*, volume 3, pages 255–264. sn, 2003.

[32] Georgia Frantzeskou, Efstathios Stamatatos, Stefanos Gritzalis, Carole E Chaski, and Blake Stephen Howald. Identifying authorship by byte-level n-grams: The source code author profile (scap) method. *International Journal of Digital Evidence*, 6(1):1–18, 2007.

[33] Moshe Koppel, Jonathan Schler, and Shlomo Argamon. Computational methods in authorship attribution. *Journal of the American Society for information Science and Technology*, 60(1):9–26, 2009.

[34] Dylan Rhodes. Author attribution with cnns. *Avaiable online: https://www. semanticscholar. org/paper/Author-Attribution-with-Cnn-s-Rhodes/0a904f9d6b47dfc574f681f4d3b41bd840871b6f/pdf (accessed on 22 August 2016)*, 2015.

[35] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[36] Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10): 1995, 1995.

[37] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of machine learning research*, 12(Aug):2493–2537, 2011.

[38] Michael Hart. Project gutenberg. `https://www.gutenberg.org/wiki/Main_Page`, 1971. (Accessed on 06/02/2019).

[39] David D Lewis, Yiming Yang, Tony G Rose, and Fan Li. Rcv1: A new benchmark collection for text categorization research. *Journal of machine learning research*, 5(Apr):361–397, 2004.

[40] Bryan Klimt and Yiming Yang. The enron corpus: A new dataset for email classification research. In *European Conference on Machine Learning*, pages 217–226. Springer, 2004.

[41] Zhou Yilu, Alsarkal Yaqoub, and Zhang Nan. Linking virtual and real-world identities twitter dataset, 2016.

[42] Gauri Jain, Manisha Sharma, and Basant Agarwal. Spam detection in social media using convolutional and long short term memory neural network. *Annals of Mathematics and Artificial Intelligence*, 85(1):21–44, 2019.

[43] Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. From word embeddings to document distances. In *International Conference on Machine Learning*, pages 957–966, 2015.

[44] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. A metric for distributions with applications to image databases. In *Sixth International Conference on Computer Vision (IEEE Cat. No. 98CH36271)*, pages 59–66. IEEE, 1998.

[45] Suthee Chaidaroon and Yi Fang. Variational deep semantic hashing for text documents. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 75–84. ACM, 2017.

[46] Peter Danziger. Linear codes. `https://math.ryerson.ca/~danziger/professor/MTH108/Handouts/codes.pdf`. (Accessed on 06/17/2019).

[47] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*, 2018.

[48] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[49] Chen Qian, Tianchang He, and Rao Zhang. Deep learning based authorship identification. 2017.

[50] T.G. Rose, M. Stevenson, and M. Whitehead. The reuters corpus volume 1-from yesterdays news to tomorrows language resources. *Proceedings of the Third International Conference on Language Resources and Evaluation*, pages 29–31, 2002. URL `http://about.reuters.com/researchandstandards/corpus/LREC_camera_ready.pdf`.

[51] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014. URL `http://www.aclweb.org/anthology/D14-1162`.

[52] Kieran Sagar Parikh, Vinodini Venkataram, and Jugal Kalita. Towards a universal document encoder for authorship attribution. 2018.

[53] John Houvardas and Efstathios Stamatatos. N-gram feature selection for authorship identification. In *International conference on artificial intelligence: Methodology, systems, and applications*, pages 77–86. Springer, 2006.

[54] Benedikt Boenninghoff, Robert M Nickel, Steffen Zeiler, and Dorothea Kolossa. Similarity learning for authorship verification in social media. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2457–2461. IEEE, 2019.

[55] Oren Halvani, Christian Winter, and Anika Pflug. Authorship verification for different languages, genres and topics. *Digital Investigation*, 16:S33–S43, 2016.

[56] Lisa Torrey and Jude Shavlik. Transfer learning. In *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*, pages 242–264. IGI Global, 2010.

[57] Yin Zhang, Rong Jin, and Zhi-Hua Zhou. Understanding bag-of-words model: a statistical framework. *International Journal of Machine Learning and Cybernetics*, 1(1-4):43–52, 2010.

[58] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[59] Chigozie Nwankpa, Winifred Ijomah, Anthony Gachagan, and Stephen Marshall. Activation functions: Comparison of trends in practice and research for deep learning. *arXiv preprint arXiv:1811.03378*, 2018.

[60] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[61] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python . *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[62] Per-Erik Danielsson. Euclidean distance mapping. *Computer Graphics and image processing*, 14(3):227–248, 1980.

[63] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *null*, pages 1735–1742. IEEE, 2006.

[64] Joost CF De Winter. Using the student's t-test with extremely small sample sizes. *Practical Assessment, Research & Evaluation*, 18(10), 2013.

[65] B Everitt. The cambridge dictionary of statistics cambridge university press. *Cambridge, UK*, 1998.

[66] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[67] Eric W Weisstein. Inverse hyperbolic tangent. 2002.

[68] Kai Sheng Tai, Richard Socher, and Christopher D Manning. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*, 2015.

[69] Duyu Tang, Bing Qin, and Ting Liu. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 1422–1432, 2015.

[70] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018.

[71] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.