2020

An Evaluation of Text Representation Techniques for Fake News Detection Using: TF-IDF, Word Embeddings, Sentence Embeddings with Linear Support Vector Machine.

Sangita Sriram

Technological University Dublin

Recommended Citation

Sriram, S. (2020). *An evaluation of text representation techniques for fake news detection using: TF_IDF, word embeddings, sentence embeddings with linear support vector machine.* Masters Dissertation. Technological University Dublin. DOI:10.21427/5519-h979

An Evaluation of Text Representation Techniques for Fake News Detection using: TF-IDF, Word Embeddings, Sentence Embeddings with Linear Support Vector Machine

OLLSCOIL TEICNEOLAÍOCHTA BHAILE ÁTHA CLIATH
DUBLIN
TECHNOLOGICAL UNIVERSITY DUBLIN

Sangita Sriram

D17129392

A dissertation submitted in partial fulfilment of the requirements of Technological University Dublin for the degree of M.Sc. in Computer Science (Data Analytics)

2020

I certify that this dissertation which I now submit for examination for the award of MSc in Computing (Data Analytics), is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the test of my work.

This dissertation was prepared according to the regulations for postgraduate study of the Technological University Dublin and has not been submitted in whole or part for an award in any other Institute or University.

The work reported on in this dissertation conforms to the principles and requirements of the Institute's guidelines for ethics in research.

Signed: **SANGITA SRIRAM**

Date: **05th January, 2020**

ABSTRACT

In a world where anybody can share their views, opinions and make it sound like these are facts about the current situation of the world, Fake News poses a huge threat especially to the reputation of people with high stature and to organizations. In the political world, this could lead to opposition parties making use of this opportunity to gain popularity in their elections. In the medical world, a fake scandalous message about a medicine giving side effects, hospital treatment gone wrong or even a false message against a practicing doctor could become a big menace to everyone involved in that news. In the world of business, one false news becoming a trending topic could definitely disrupt their future business earnings. The detection of such false news becomes very important in today's world, where almost everyone has an access to use a mobile phone and can cause enough disruption by creating one false statement and making it a viral hit. Generation of fake news articles gathered more attention during the US Presidential Elections in 2016, leading to a high number of scientists and researchers to explore this NLP problem with deep interest and a sense of urgency too.

This research intends to develop and compare a Fake News classifier using Linear Support Vector Machine Classifier built on traditional text feature representation technique Term Frequency Inverse Document Frequency (Ahmed, Traore & Saad, 2017), against a classifier built on the latest developments for text feature representations such as: word embeddings using 'word2vec' and sentence embeddings using 'Universal Sentence Encoder'.

Key words: Fake News Detection, Linear Support Vector Machine, Word Embedding, Sentence Embedding, TF-IDF

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my Supervisor **Jack O'Neill**, whose complete guidance, advice, insights, right from the beginning of my thesis, helped me very greatly into shaping what it is today. His constant support whenever I was stuck and constant encouragement whenever I made huge progress, kept me going steadily.

I would also like to thank Dr. Luca Longo, Dr. Robert John Ross, Dr. Susan Mckeever, Dr. Sarah Jane Delany and all the TUD Computing Department staff for contributing significantly to my Master's journey.

A day would never become complete if I don't thank my parents enough for their utmost support, encouragement and love for me throughout this entire journey in Ireland. Forever indebted to them.

A huge shoutout and gratitude to all my friends and classmates, who have always been there for me, with their kind words of encouragement, which kept me going.

INDEX OF CONTENTS

1. INTRODUCTION..1

1.1 Background..1

1.2 Research Problem...1

1.3 Research Objectives...3

1.4 Research Methodologies..3

1.5 Scope and Limitations...4

1.6 Documentation Outline..5

2. LITERATURE REVIEW AND RELATED WORK……………………...7

2.1 Related Work………………..……………………….........................7

2.1.1 Fake news detection……….…………………………….……7

2.1.2 Linear Support Vector Machine Classifier for Text Classification………………..……………………….........9

2.1.3 TF-IDF text representation technique for text analysis.……10

2.1.4 Word embedding text representation technique for text classification…………..……………….………………........11

2.1.5 Sentence embedding text representation technique for text classification..……….…………………………….….........12

2.2 State of the art approaches in Fake News Text Detection……..…….13

2.3 Research Gaps………………..………………………......................15

3. DESIGN AND METHODOLOGY...17

3.1 Data Understanding..………..……………………………….............17

3.1.1 First dataset description..………..…………………………….17

3.1.2 Second dataset description..………..………………………….20

3.1.3 Third dataset description..………..……………………………22

3.2 Data Preparation.………..…………………………….....................22

3.2.1 Handling missing data and Lowercasing…….……………..23

3.2.2 Dealing with Duplicates.……….…………………………...23

3.2.3 Regular Expressions – Punctuation, Numbers and

Alphanumeric texts removal……….…………………….24
3.2.4 Named Entity Recognition - Spacy……….……………….24
3.2.5 Removal of Stop words and Non-English words…………...24
3.2.6 Lemmatization.………..……..…………………………...25
3.2.7 Text Feature Extraction Techniques..………..……………25
3.2.7.1 Term Frequency – Inverse Document Frequency…25
3.2.7.2 Word Embedding - Word2vec.………..…………..26
3.2.7.3 Sentence Embedding-Universal Sentence Encoder 27
3.3 Modeling.………………..……..…………..……………………..27
3.4 Evaluation.………………..……..…………..…………………....28
3.5 Software Used.………..……………..…………..…………………….30
3.6 Strengths and Limitations.………………..……..……………………30
4. IMPLEMENTATION AND RESULTS.………..……..…………………….32
4.1 Data Preparation…………………….…………….......................32
4.1.1 Removing Null values and Lowercasing……………………32
4.1.2 Removing Duplicates…………………..…………………32
4.1.3 Regular Expressions - Removing Punctuation, Numbers,
Alpha-numeric texts……………………..…………………33
4.1.4 Named Entity Recognition – Spacy………………………….33
4.1.5 Removal of Non-English words and Stop words…………….34
4.1.6 Lemmatization…………………..…………………………34
4.1.7 An experimental example………….………..……………..34
4.2 Modeling…………………..………………...............................36
4.2.1 Baseline Model…………………..….……………………...36
4.2.2 Linear SVM with Word Embeddings….……………………...37
4.2.3 Linear SVM with Sentence Embeddings..……………….........39
4.3 Evaluation……..……………………………….............................41
4.3.1 10-fold Cross-Validation Mean Scores for all datasets…..........41
4.3.2.1 Individual scores of 10-fold Cross-Validation for
Kaggle Competition Dataset………………………..41
4.3.2.2 Individual scores of 10-fold Cross-Validation for
Kaggle Combined Dataset…………………………..42
4.3.2.3 Individual scores of 10-fold Cross-Validation for

Politifact Buzzfeed Dataset…………………………..43

4.3.3.1 Confusion matrix scores for the Kaggle Competition Dataset……..……………………………………...43

4.3.3.2 Confusion matrix scores for the Kaggle Combined Dataset……..……………………………………...43

4.3.3.3 Confusion matrix scores for the Politifact Buzzfeed Dataset……..……………………………………...44

5. EVALUATION, ANALYSIS AND DISCUSSION…………………….................47

5.1 Evaluation……..…………………………………...............................47

5.1.1 Normality Tests…………………….......................................47

5.1.2 Statistical Difference Tests……………………......................48

5.2 Acceptance/Rejection of Null and Alternate Hypothesis………............49

5.3 Summary of Results and Discussion………………….........................50

5.4 An additional experiment of Sentiment Analysis for Future Work..……..51

6. CONCLUSION……………………………………….................................53

6.1 Research Overview……………………………….................................53

6.2 Problem Definition……………………………….................................53

6.3 Experiments, Evaluation and Results……………………………………..53

6.4 Contribution and Impact………………………………..........................54

6.5 Future Work and Recommendation…………………………..................55

BIBLIOGRAPHY……………………………………….................................57

INDEX OF FIGURES

3.1 Thesis implementation diagram
3.2 Top 20 Frequent word count plot for the Kaggle Competition dataset
3.3 Top 20 Frequent Bi-gram word count plot for the Kaggle Competition dataset
3.4 Word cloud for reliable article ‘text’ column in Kaggle Competition dataset
3.5 Word cloud for fake article ‘text’ column Kaggle Competition dataset
3.7 Confusion matrix
3.8 Recall
3.9 Precision
4.1 ELI5 (Explain Like I am 5) feature image
4.2 Overfitting, Underfitting diagram
4.3 Classification Report for Kaggle Competition dataset’s TF-IDF based LSVM model
4.4 Classification Report for Kaggle Combined dataset’s TF-IDF based LSVM model
4.5 Classification Report for Politifact Buzzfeed dataset’s TF-IDF based LSVM model
4.6 Classification Report for Kaggle Competition dataset’s word embeddings based LSVM model
4.7 Classification Report for Kaggle Combined dataset’s word embeddings based LSVM model
4.8 Classification Report for Politifact dataset’s word embeddings based LSVM model
4.9 Classification Report for Kaggle Competition dataset’s Sentence embeddings based LSVM model
4.10 Classification Report for Kaggle Combined dataset’s Sentence embeddings based LSVM model
4.11 Classification Report for Politifact Buzzfeed dataset’s Sentence embeddings based LSVM model
4.12 Individual scores of 10-fold Cross-Validation for Kaggle Competition Dataset
4.13 Individual scores of 10-fold Cross-Validation for Kaggle Combined Dataset
4.14 Individual scores of 10-fold Cross-Validation for Politifact Buzzfeed Dataset

4.15 Confusion matrix scores for the Kaggle Competition Dataset
4.16 Confusion matrix scores for the Kaggle Combined Dataset
4.17 Confusion matrix scores for the Politifact Buzzfeed Dataset

INDEX OF TABLES

3.1 Distribution of fake and reliable articles in the Kaggle Competition dataset
3.2 Distribution among the fake and real articles for the Kaggle Combined dataset
3.3 Distribution among the fake and real articles for the Politifact Buzzfeed dataset
4.1 Accuracy result achieved using TF-IDF before and after data-preprocessing using Linear SVM model on the Kaggle Competition dataset.
4.2 10-fold Cross-Validation Mean Scores for all three datasets
4.3 Confusion matrix scores for the Kaggle Competition Dataset
4.4 Confusion matrix scores for the Kaggle Combined Dataset
4.5 Confusion matrix scores for the Politifact Buzzfeed Dataset
5.1 Normality Test Score Results
5.2 Statistical Difference Test Results
5.3 Sentiment scores on Fake articles
5.4 Sentiment scores on Reliable Articles

LIST OF ACRONYMS

BoW	Bag of Words
TF-IDF	Term Frequency Inverse Document Frequency
LSVM	Linear Support Vector Machine
SVM	Support Vector Machine
LSTM	Long Short-Term Memory
ML	Machine Learning
NLP	Natural Language Processing
Regex	Regular Expressions
NN	Neural Network
DL	Deep Learning
SGD	Stochastic Gradient Descent
DT	Decision Tree
KNN	K-Nearest Neighbors
DAN	Deep Averaging Network
CRISP-DM	Cross InduStry Process for Data Mining

CHAPTER 1
INTRODUCTION

1.1 Background

Owing to the given rise in the usage of Social Media and other platforms, anybody can share their opinions, perspectives, sharing seemingly truthful yet actually deceptive facts about the current stories and make them a trending topic without really having the need to go to the source and verify the credibility of what they are saying or sharing. “Fake news detection” can be defined as the task of categorizing news as truthful, yet compromised with the occurrence of intended deceptions, with an associated measure of certainty (Conroy, Rubin & Chen, 2015). It is rather a challenge to prove the intentionality behind the deception. This could have been purposely carried out by the writer for various reasons: spreading false propaganda, creating fabricated news, partial lies, unsupported accusations in order to drive web traffic and to get quick eyes on their articles & gather attention.

Detecting such fake news content and removing it immediately from any medium of print is a crucial step. One such example is when the German Government Officials were put into turmoil in January 2017, to undo the effects caused by the unprecedented spread of fake news of its political leaders.[1]

Sometimes the published news is attached with a controversial image of the people involved in the article, to convince readers into believing what they read is actually true. Thus, the real impact of the need to detect fake news and stop it from spreading, has risen from the realization that the public is not really equipped, to separate quality, truthful information from false information and rumours.

1.2 Research Project/Problem

Most text classification algorithms today use the traditional N-gram Term Frequency Inverse Document Frequency (TF-IDF) as the text feature representation technique for converting text into vectors, as text cannot be directly fed into a machine learning classifier. TF-IDF is a measure of relative importance of the word in a corpus of documents, based on its frequency of occurrence in the document. This potentially means that a word that appears many times (after excluding stop-words) across the corpus could be of relatively high importance to this body of text. In contrast, TF-IDF doesn't really grasp or capture the position of a word in text, the context to which it belongs to, semantics, occurrences of the same word in different documents, as TF-IDF is based on the bag-of-words (BoW) model.

Whereas, in word embeddings the position of a word within the vector space is learned from text and is based on the words that surround the word when it is used. For fake news detection, using word embeddings makes more sense, as understanding the contextual meaning of a word becomes highly important. A word separately could be of a different meaning and a word surrounded by a group of words in a paragraph, could have a contextual meaning.

Sentence embedding is the extended version of word embeddings, where an entire sentence is mapped to a vector of real numbers, which will capture the meaning of text at a greater length.

Word and sentence embeddings were never used as the text feature extraction technique for classifying fake and real news. Thus, the main objective of this research is to use word embeddings and sentence embeddings as the text feature representation technique with the Linear Support Vector Machine model to classify data as fake news or not.

An additional research area for future work was identified and experimented to explore if negative sentiment gave rise to more fake news articles being written and to explore if reliable articles were written with a positive or neutral sentiment.

Research Question

The research question that this research intends to answer can be written as below:

Can Linear Support Vector Machine model built using 'word2vec' word embeddings and 'Universal Sentence Encoder' sentence embeddings as the text

[1] www.theguardian.com/world/2017/jan/09/germany-investigating-spread-fake-news-online-russia-election

feature representation technique, be able to achieve a statistically significant higher accuracy than the Linear Support Vector Machine model built using traditional Term Frequency-Inverse Document Frequency as the text feature representation technique, for classifying fake news text?

1.3 Research Objectives

To answer the research question that this research intends to evaluate, the question should be first written in the form of hypotheses, and based on the results of the experiment, the hypothesis will either be accepted or rejected. To ultimately accept or reject a hypothesis, it has to be done through implementation of experiments and to calculate and compare the metrics. After performing statistical difference tests between the two models, if the difference is statistically significant ($p<0.05$), we reject the Null Hypothesis and accept the Alternate Hypothesis.

H0 (Null Hypothesis):

LSVM classifier model built on ‘word2vec’ word embedding and ‘Universal Sentence Encoder’ sentence embedding text feature representation technique achieves a statistically significant higher accuracy for predicting fake news, than a LSVM classifier model built on traditional TF-IDF text feature representation technique.

H1 (Alternate Hypothesis):

LSVM classifier model built on ‘word2vec’ word embedding and ‘Universal Sentence Encoder’ sentence embedding text feature representation technique does not achieve a statistically significant higher accuracy for predicting fake news, than a LSVM classifier model built on traditional TF-IDF text feature representation technique.

1.4 Research Methodologies

This research carries out a secondary desk research methodology and employs quantitative research strategies throughout the life cycle.

Secondary desk research was used by collecting three different fake news datasets available in Kaggle website. [2,3,4,5] A more detailed explanation about the nature of the data will be explained in the later section.

A deductive reasoning methodology is followed as the research question or theory is stated, clear hypotheses are written, experiments are carried out to accept or reject the research hypothesis and finally results are obtained which are evaluated using statistical methods.

An exploratory research is also carried out by framing a possible area for future work for performing sentiment analysis in fake and reliable news articles, and also provides certain solutions with the tools available at hand.

1.5 Scope and Limitations

All the datasets used in this research are almost collected from around the same period of 2016 US Presidential elections, as fake news became prevalent from that time. Having datasets belonging to recent times, or even news of older times, would help to generalize the model more, so that the model learns and deals with the data points that were collected more recently.

The dataset was also surprisingly well balanced, having equal number of fake and reliable news articles, which may not be the actual scenario in the real world.

Word embeddings and Sentence Embeddings are mostly used for training deep learning algorithms, but this research has used these techniques for a machine learning algorithm.

The hyperparameters used on the LSVM model were completely default features, as the previous research work (Ahmed, Traore & Saad, 2017) had no mention about the parameters they had used to recreate what they did, yet the same accuracy as

---

2 https://www.kaggle.com/mrisdal/fake-news/kernels

3 https://www.kaggle.com/c/fake-news/overview

4 https://www.kaggle.com/snapcrack/all-the-news

5 https://www.kaggle.com/mdepak/fakenewsnet

theirs was achieved by using 1-gram TF-IDF text representation technique. Tweaking the hyperparameters only seemed to lower the accuracy achieved by every single model used in this method, hence the hyperparameters were default.

1.6 Documentation Outline

The thesis is structured as follows:

Chapter 2 Literature Review:

This chapter explores a detailed review written on all the previous research works implemented in the area of fake news detection, word embeddings, sentence embeddings, text classification, TF-IDF. The research gaps are found through this review and a research question is framed from the gaps identified.

Chapter 3 Design / Methodology:

This chapter describes how the thesis is implemented by following a CRISP-DM methodology for carrying out experiments laid out by the research question. Datasets are initially described and then data understanding forms a major part of this section. Data understanding helps to explain why these particular data preparation/data cleansing tasks were selected to use on these datasets. Recreating the previous research using Linear SVC with TF-IDF is discussed and then creating Linear SVC with word and sentence embeddings is explored.

Chapter 4 Implementation and Results:

This chapter describes all the experiments and the results obtained in detail. The data preparation steps that were conducted are described in detail initially. Then, the text feature representation techniques are described. Finally, the results obtained by employing TF-IDF, word2vec, Universal Sentence Encoder with Linear SVC are presented and written on in great detail.

Chapter 5 Evaluation and Analysis:

The cross-validation results, statistical tests that were carried out to accept or reject the hypothesis are defined and described in this chapter. An analysis on the results obtained and what they could mean are discussed later. An experiment was explored to perform sentiment analysis on fake and reliable news articles and discussed.

Chapter 6 Conclusions and Future Work:

This chapter finally summarizes the research work carried out in this project. The potential future work that can be explored in the field of fake news detection is also suggested.

CHAPTER 2
LITERATURE REVIEW AND RELATED WORK

This section is dedicated to record the previous works of established researches performed in Fake news detection, papers that had used Linear Support Vector Machine model. TF-IDF, word embedders and sentence embedders that were used for classifying text. This is followed by specifying the state of the art and the research gaps that were identified.

2.1 Related Work:

2.1.1 Fake news detection:

Ruchansky, Seo and Liu aim to create a model that captures, scores and integrates: text, responses, and source, for accurate and automatic prediction of fake news. This paper tries to improve the accuracy and effectiveness of the method in identifying if a news article is fake or not. There are three separate manual modules of detecting fake news and its limitations that have been spoken widely in this paper. 'Text' of the article is used in the first module to check whether the headings are matching with its content. Simple machine learning techniques and Natural language processing are performed over text to extract the textual features and classify them as fake or true. However, the first module can lead to false-positive scenarios where linguistic features are not taken into consideration. The second module was about analyzing the response for the fake news. The comments and arguments of the users against the news are used to detect the user's action towards it. The most obviously fake news contains inflammatory language in their comment sections. Social media is an amazing platform to understand this sentiment of the users over the news and simple classifiers can be used to say whether it's fake or not. However, this method is very labor-intensive and takes a lot of time. The Third module is to find the source of the news article. The method involves checking the URL, background verifying the publisher and checking the post score. The main limitation of these methods is the hand-crafted feature selection for the classification. To overcome this, the author has proposed a Deep neural network CSI model that can automatically select the features on its own and perform the classification and provide the result as the fake or true article. (Ruchansky, Seo & Liu, 2017).

Conroy, Rubin, and Chen propose to find the veracity of a document by using machine learning and network analysis approaches. This paper discusses different deception assessment methods and their results with the aim of developing a hybrid method. The two main approaches have been widely looked upon such as linguistic and network approach. The linguistic method involves training a machine learning algorithm on the text to classify them as fake or not based on textual features and language patterns. Whereas, the network method involves analyzing the metadata and queries. The most common methods used in the linguistic approach are finding the n-grams and grouping the words, probability context-free grammars for categorizing based on rules and rhetorical structure and discourse analysis. Social network behavior is widely used in the classification of the fake news article. Finally, the author proposes a method that incorporates a highly sophisticated model that performs classification on linguistic features using multiple layers to attain the highest performance and both linguistic and network approaches have to be combined. (Conroy, Rubin & Chen, 2015).

Shu, Silva, Wang, Tang, and Liu (2017) wish to present a survey, giving a comprehensive review to detect fake news on social media, including fake news characterizations on psychology and social theories, existing algorithms from a data mining perspective, evaluation metrics and representative datasets. The fake news gives a negative impact on society and entities and spoils the reputation of people or brands. Detecting fake news based on the news body is very difficult therefore the background information of the publisher has to be analyzed. The paper describes the news into two phases namely Characterisation and Detection. The first phase in characterization where it explains about a technological shift in the newsreaders. The readers rely heavily on social media for news rather than the traditional news channels and newspapers. Therefore, social media is a widely used platform for the spreading of unreliable news articles. It is less expensive, quick, feedback for each news can be given in the comment section and shared as well. These media platforms are utilized for generating fake news for intentions like political gains, false marketing, and financial purposes. These types of fake news can influence the users who falsely believe them. The second phase is the detection phase, where data mining techniques and feature extractions are used to identify whether the news is false or not. (Shu et al., 2017).

Granik and Mesyura intend to show a modest approach for fake news detection using the simple artificial intelligent algorithm called Navie Bayes. The Spam filtering technique is taken as an analogy for this fake news detection. The data from Facebook API is used for the model building and new labeled data is then used for testing. The model had an accuracy of 76% even for the simple classifier algorithm. (Granik & Mesyura, 2017).

Bourgonje, Schneider, and Rehm wish to deal with fake news and related online phenomena with technological means, by providing means to separate related from unrelated headlines and further classifying the related headlines. This paper discusses the determination of attitude of the headlines over the news body. A robust methodology coupled with lemmatization and n-gram classification is used to classify the news articles between fake or not. A series of classification techniques called fine-grained classifiers were used in building the model. The weighted accuracy of 89% was achieved using this technique. (Bourgonje, Schneider & Rehm, 2017).

Hai et al. propose to exploit the relatedness of multiple review spam detection tasks and readily available unlabeled data to address the scarcity of labeled opinion spam data. The supervised classification algorithms and feature engineering techniques have the main disadvantage like it requires ample amount of ground data and require expert opinions with the domain knowledge. To overcome this, a new approach is proposed with multi-task learning based on the logistic regression technique. This technique works on the knowledge sharing formula which is automatically learned during the model creation. In order to overcome the absence of labeled data graph a laplacian regularizer is used in the model. Finally, to improve the performance of the model, semi-supervised multi-task learning integrated with laplacian regularizer and logistic regression. Experiments were done on the real world unlabeled review data to check whether its fake or not (Hai et al., 2016).

2.1.2 Linear Support Vector Machine Classifier for Text Classification:

The Improved Global Feature Selection Scheme is used for selecting equal quantity of features for automatically classifying text documents. As this method may exclude some important features, a Variable Global Feature Selection Scheme is proposed to identify variable quantity of features from a class by calculating the distribution of features/terms in these classes, which is applicable for both balanced and unbalanced datasets. This VGFSS is an ensemble algorithm combining a filter-based feature selection by making use of a local and global score computation of a feature. This is built mathematically using a Linear Support Vector Machine classifier. (Agnihotri, Verma, & Tripathi, 2017).

A Linear SVM was used to perform binary classification on a corpus of German spoken radio documents. The LSVM performs well by controlling input space complexity provided by higher order combinations of sub word features. A combination of syllables and phonemes are fed as an input to the LSVM. The highest accuracy achieved by LSVM is while classifying audio documents for topic politics and using syllable 2-grams as an input, with an accuracy of 63.1% (Larson et al., 2002).

On comparing effectiveness between five automatic text categorization algorithms, in terms of accuracy, training time, real time classification speed. Training dataset size and document representations are also examined. Linear Support Vector Machine performed the best compared against Decision tree, Naïve Bayes, Bayes Nets and Find Similar. LSVMs are highly accurate, gets trained quickly in lesser time and evaluates quickly (Dumais, Platt, Heckerman & Sahami, 1998).

Eight linear support vector machine variants were investigated and compared for text classification. The experiments used two benchmark text datasets: ohsumed and reuters-21578. Results revealed that Linear SVM and Proximal SVM performs better in terms of F1 scores and Break Even Point scores (BEP) (Kumar & Gopal, 2010).

2.1.3 TF-IDF text representation technique for text analysis:

A normalized TF-IDF classifier with weight=1, trained on 12,000 news articles in Bahasa Indonesia, to classify into 15 different categories, resulted in a very high accuracy of 98.3% (Hakim et al., 2014).

An analysis was done for examining the effects of applying TF-IDF to determine the most appropriate words present among a corpus of documents, that can be utilized in a query. A word with high TF-IDF score indicated a positive relationship with the document that they are present in, implying that if this word appears in a query, that associated document should be suggested to the user as it could be of interest to them. Using this TF-IDF based query retrieval resulted in a high amount of documents containing relevant information, being returned for a query, compared to a Naïve brute force query retrieval approach (Ramos, 2003).

An improved TF-IDF technique for enhancing precision and recall scores for classifying texts, by using support, confidence and characteristic words was developed. Predefined lexicons are processed to make use of the synonyms that are already defined, in this new TF-IDF technique. The experiments revealed that the new TF-IDF technique highly improves the recall and precision scores for textual classification in science and technology domains, compared with the traditional TF-IDF approach (Yun-tao, Ling, & Yong-cheng, 2005).

2.1.4 Word embedding text representation technique for text classification:

Clinical texts are automatically classified using deep convolutional neural networks at a sentence level. A word2vec based CNN outperforms other approaches such as: Sentence Embeddings and Mean word embeddings by at least a 15% higher accuracy (Hughes, Li, Kotoulas & Suzumura, 2017).

An architecture for creating continuous representations of words trained on large sized datasets of 1.6 billion words was developed. Words having degrees of similarity helps in performing operations on word vectors. A continuous bag of words architecture and skip gram architecture based word vector representation known as ‘word2vec’ was thus created (Mikolov, Chen, Corrado & Dean, 2013).

An extension of the originally proposed Skip-gram model was developed. Sub-sampling frequently occurring words while training, have resulted in remarkably low running times and also improves accuracy of representation of words that do not appear frequently. An extension was applied by using phrase-based models over single word-based models, and these phrases are considered as a single token during training.

Vector addition by adding two vector representation of different words, seemingly produced a third meaningful word (Mikolov et al., 2013).

It was identified that most models only learn the syntactic meaning of word in the context it is present. To avoid losing on the sentiment polarity of a word, a word embedding for learning sentiment by learning from tweets having negative and positive emoticons, which were distant-supervised, was developed. The sentiment based word embedding model outperformed existing Neural network models such as C&W, word2vec with an accuracy of 77.3% (Tang et al., 2014).

The negative sentiment levels in Austrian parliamentary speeches was analysed using word embeddings in a supervised learning approach. An advantage found by using word embeddings was that unseen words, words that are not in context that were not in the training data were detected and accurately classified. This resulted in a higher accuracy than a traditional bag of words model. A realistic class distribution was also recognized well by word embeddings (Rudkowsky et al., 2018).

A domain specifically built word embeddings and a generally built word embedding were implemented and compared, which revealed that domain specifically trained word embeddings did not necessarily improve over the generally built word embeddings model, partially because generally built word embeddings are normally trained on a vast corpus of words. The generally built word embeddings also achieve good computational power (Major, Surkis & Aphinyanaphongs, 2018).

2.1.5 Sentence embedding text representation technique for text classification:

Sentence embeddings trained on 30 million records of articles from PubMed and clinical notes in MIMIC-III database known as BioSentVec was developed as there were no pre-trained sentence embedders for biomedical texts. BioSentVec was evaluated in a sentence pair similarity classification task. A deep learning algorithm with BioSentVec achieves an accuracy of 85% for supervised method and an accuracy of 82% for unsupervised method (Chen, Peng, & Lu, 2019).

An unsupervised sentence embedding: that is built using word embeddings computed on unlabelled corpus of Wikipedia text, represents the sentence by taking the weighted average of word embedding vectors which when modified with a PCA technique results in performance improvement by 10-30% increase becoming a very good baseline for sentence embeddings (Arora, Liang & Ma, 2016).

A paraphrastic sentence embedding was learned by making use of averaging word vectors and by updating the standard word embedding vectors based on supervision on a Paraphrase Database (Ganitkevitch et al., 2013). This supervision obtained from paraphrase pairs was used during initialization and training. Complex architectures like Long Short Term Memory (LSTM) and Recurrent Neural Networks (RNN) performed well on in-domain features. For out-of-domain data, simple word averaging architecture outperforms the complex LSTM architecture (Wieting et al., 2016).

'Universal Sentence Encoder' models for converting and encoding entire sentences into embedding vectors were created, which outperformed baseline models using word embedding level transfer learning and baseline models that do not use any transfer learning. Good performance was achieved by transfer learning - sentence embedding models, that were trained only on a small size of supervised training data. A Deep Averaging Network was formulated to encode sentences into vector embeddings (Cer et al., 2018).

A 2-Dimensional matrix was used instead of creating vectors to represent the sentence embeddings, where each row of the matrix represents a part of the sentence. Evaluating this model on author profiling task, highest accuracy of 64.21% was achieved, an accuracy of 84.4% was obtained for textual entailment task and an accuracy of 80.45% was achieved for sentiment classification. This paper's sentence embedding model was built in two parts: One being a bidirectional LSTM and the second one with a self-attention mechanism, this provides summed weighted vectors from the hidden LSTM states. This summed weighted LSTM sentence embedder outperformed all the other models like 300 Dimensional LSTM encoder, 600 Dimensional BiLSTM encoders, 300D Tree-based CNN encoders, 300 Dimensional SPINN-PI encoders, 300 Dimensional NTI-SLSTM-LSTM, 1024 Dimensional GRU encoders with SkipThoughts pre-training, 300 Dimensional NSE encoders (Lin et al., 2017).

2.2 State of the art approaches in Fake News Text Detection

(Ruchansky, Seo & Liu, 2017) have used Recurrent Neural Network and an implicit user graph are used for the first two modules and finally combines the response, text, and source information from the first two modules to classify each article as fake or not.

(Hai et al., 2016) have used a multi-task learning method based on logistic regression (MTL-LR), which can boost the learning for a task by sharing the knowledge contained in the training signals of other related tasks. A novel semi-supervised multitask learning method via Laplacian regularized logistic regression (SMTL-LLR) was created to further improve the review spam detection performance.

(Granik & Mesyura, 2017) implemented the Nave Bayes classifier and implemented it as a software system and tested against a data set of Facebook news posts and achieved classification accuracy of 74 percent.

An analysis on the Signal Media dataset containing 11,000 articles using a bi-gram TF-IDF making use of Probabilistic Context Free Grammar detection technique, with Stochastic Gradient Descent classification algorithm resulted in an accuracy of 77.2%, compared against SVM, Gradient Boosting, Random Forests and Bounded Decision Tree algorithms. An n-max of 500 articles resampling was followed. A very detailed pre-processing of data was followed in this research paper, paying key attention to performing the steps based on the political domain/nature of the dataset. Named Entity Recognition was performed to remove the mention of any political personality's name, organization. The article's source, twitter handle names and email IDs were also removed. Although 77.2% accuracy could be in the mid-range, based on the pre-processing steps performed, this could be a very reliably built model, as the data is very much generalized.

A fake news detection model that uses n-gram analysis with six different ML models were developed. Term frequency-Inverse Document Frequency (TF-IDF) and Term Frequency (TF) N-grams such as Uni-gram, Bi-gram, Tri-gram, Four-gram with feature size of 1000, 5000, 10000 and 50000 were developed and compared between the following machine learning algorithms: SVM, LSVM, KNN, Decision Tree and Stochastic Gradient Descent. The comparisons were investigated. The Uni-gram TF-IDF with top 50000 features using Linear Support Vector Machine Classifier (LSVM) yielded a very high accuracy of 92% (Ahmed, Traore & Saad, 2017). **This model**

produced the highest result amongst all the other research papers developed for fake news detection.

2.3 Research Gaps

On analysis of the existing literature carried out by other authors, good accuracy results have been achieved in the models that were used. On the modelling done by (Granik & Mesyura, 2017), they have implemented the Naïve Bayes classifier on a very small dataset with just 2000 instances and achieved classification accuracy of 75 percent and if a larger dataset had been used instead, better results might have been achieved.

A novel semi-supervised multitask learning method via Laplacian regularized logistic regression (SMTL-LLR) used on unlabelled dataset had achieved an accuracy of 87 percent (Hai et al.,2016). Although the model performs well on unlabelled data, if an higher number of unlabelled data are added, it might result in an increase in noise and decrease the performance of SMTL-LLR.

The research work done by (Gilda, 2017) using Stochastic Gradient Descent classification algorithm, used TF-IDF as the text representation technique, which resulted in an accuracy of 77.2% and the work by (Ahmed, Traore & Saad, 2017) again used an uni-gram TF-IDF as the text representation technique with LSVM classifier model, which achieved an accuracy of 92%.

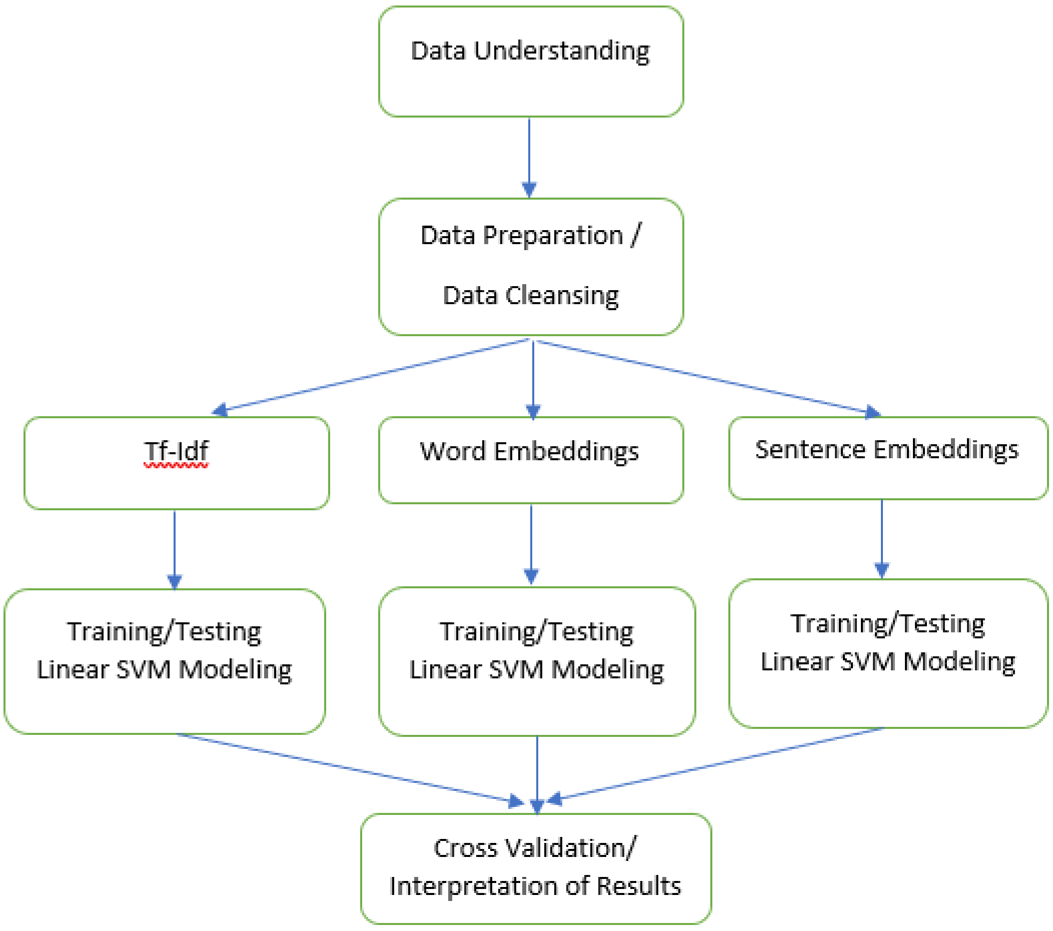
In all the existing literature papers, all the researchers had performed the text feature extraction technique using TF-IDF method. A TF-IDF value or score represents the relative importance of a term in the document based on its frequency of occurrence with respect to the total number of words in that document, inverted by the total number of documents that contain this word in the entire corpus. However, it was discovered that word embeddings and sentence embeddings were never used to perform the text feature extraction technique. Embedding is a form of representing words and documents using a dense vector representation with pre-specified n-dimensions.

This research will attempt to utilize word embeddings and sentence embeddings as the text feature extraction technique, with Linear Support Vector Machine Classifier, and will compare and present the results of the embeddings

based LSVM model against a TF-IDF text feature extraction technique based LSVM model developed by (Ahmed, Traore & Saad, 2017).

CHAPTER 3
DESIGN AND METHODOLOGY

This section will define all the steps that will be carried out in this thesis. The figure 3.1 gives a pretty much clear outline of how this thesis is implemented. The CRISP-DM methodology will be followed for the implementation of this project, as it is a very structured approach to carry out any data science project.



Data Understanding
Data Preparation / Data Cleansing
Tf-Idf
Word Embeddings
Sentence Embeddings
Training/Testing Linear SVM Modeling
Training/Testing Linear SVM Modeling
Training/Testing Linear SVM Modeling
Cross Validation/ Interpretation of Results

Figure 3.1: Thesis implementation diagram

3.1 Data Understanding:

This thesis utilized three datasets for carrying out the experiments. The following section describes each of the dataset and how & where they were gathered from.

3.1.1 First dataset description:

The first dataset was picked from a Kaggle Competition for classifying fake and real news. This dataset consists of 20,761 rows and 4 columns, with the column 'label' containing 0 – reliable and 1 – fake. Title, Author, Text and Label are the columns of this dataset. Only the 'text' column was analyzed and this becomes the dependent variable and the 'label' column is the variable to be predicted. The interesting fact about this dataset was that the number of fake and reliable articles were equally distributed, hence there was no question of an imbalanced dataset. This dataset will be referred as the Kaggle Competition dataset in the later sections of the thesis.

Type of News Article	**Distribution**	**Average count of words in 'text' column**	**Average count of words in 'text' column after data cleansing**
Reliable - (0)	10387	878	751
Fake - (1)	10374	641	537

Table 3.1 - Distribution of fake and reliable articles in the Kaggle Competition dataset

The fake news articles had texts in Icelandic, Russian, Spanish, Arabic, Chinese, Sindhi, Ukranian, Greek, Galician, Turkish, German, Mongolian. For example, the Arabic words found in the article which means 'لتأمين' 'To Secure' as given by Google Translate and 'مؤسسة' means 'Corporation'. The Ukranian word 'ядерному' means 'Nuclear' in English. Removal of these words were handled and will be explained in detail in the Data Preparation part.

Top 20 Frequent word count plot for the Kaggle Competition dataset:

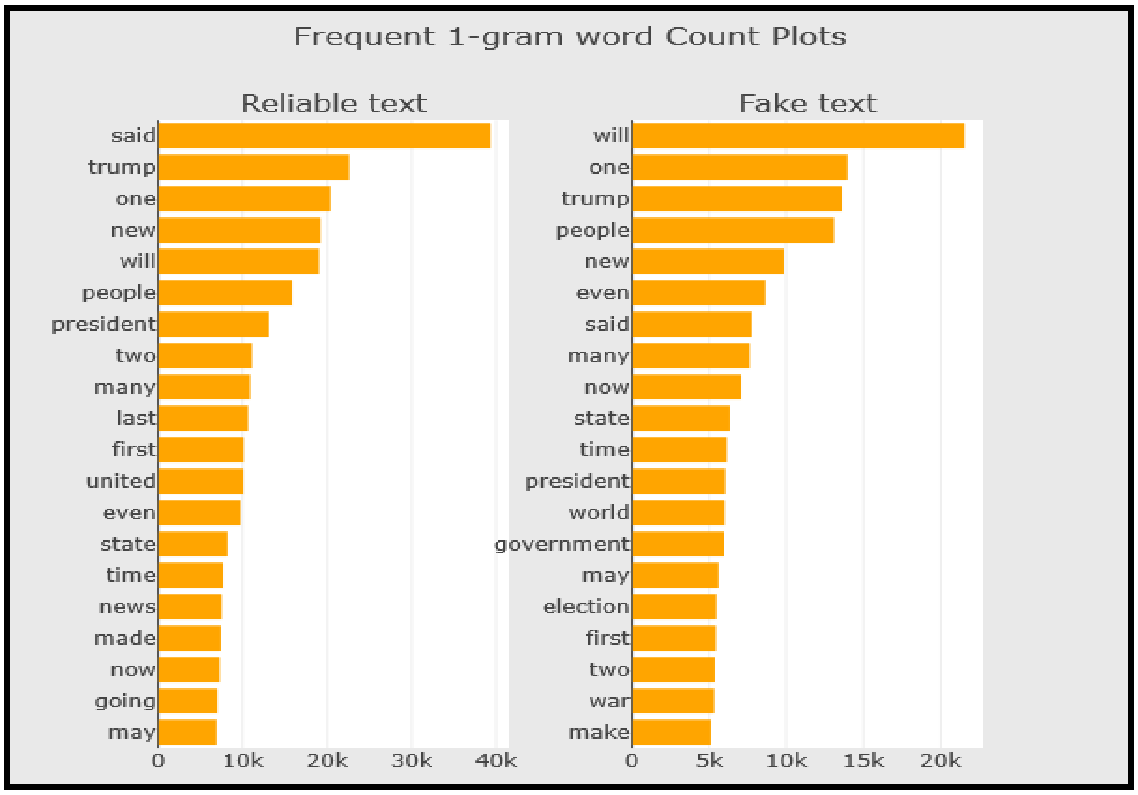


Figure 3.2 - Top 20 Frequent word count plot

Top 20 Frequent Bi-gram word count plot for the Kaggle Competition dataset:

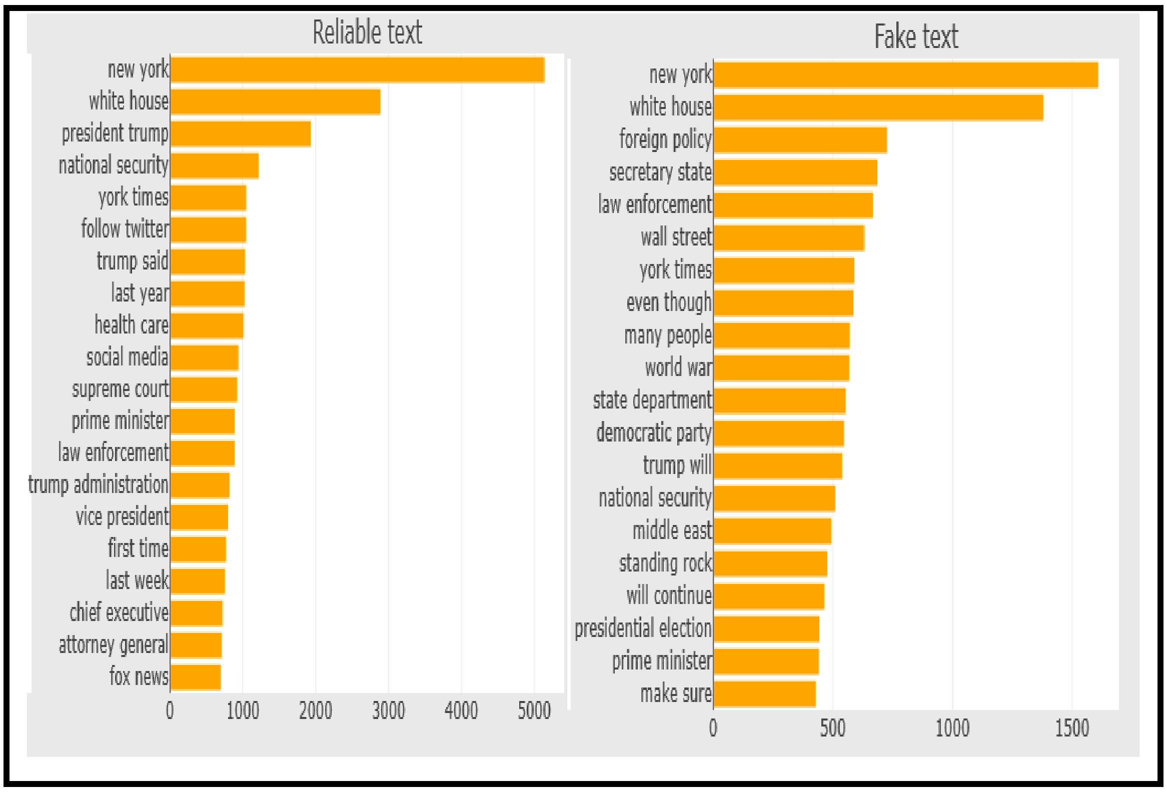


Figure 3.3 - Top 20 Frequent Bi-gram word count plot

Figure 3.4 shows the word cloud generated for reliable ‘text’ column in Kaggle Competition dataset:

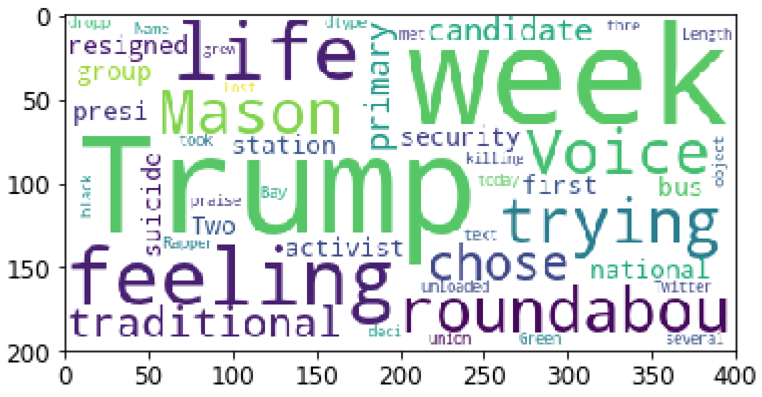


Figure 3.4 - Reliable word cloud

Figure 3.5 shows the word cloud generated for fake ‘text’ column Kaggle Competition dataset:

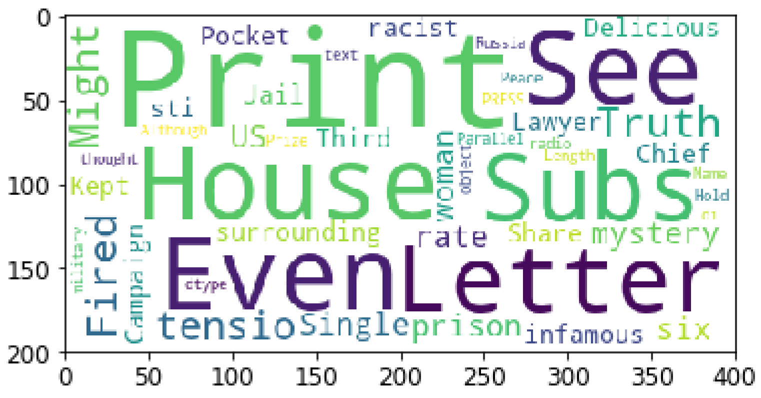


Figure 3.5 - Fake word cloud

3.1.2 Second dataset description:

The second dataset was collected from two separate Kaggle dataset kernels and combined together to form a single dataset. Two separate Kaggle datasets had to be used because one dataset consisted of only fake news articles belonging to the period of October to December 2016, which contains text collected from 244 websites and consists of 12,999 articles. This was scraped from websites that was tagged as "bullshit" by the BS Detector Chrome Extension. This dataset contained the following columns: uuid, ord_in_thread, author, published, title, text, language, crawled, site_url.

Only the ‘text’ column was extracted from this dataset. As all the articles from this dataset are fake articles, a new column ‘label’ was created with every article labeled as 1 for indicating it is fake.

For gathering reliable articles, to match with this fake articles dataset, another Kaggle dataset was considered that consisted of reliable published articles in famous news publications. This dataset was scraped using Beautiful Soup from renowned publications such as, New York Times, Business Insider, Breitbart, the Atlantic, CNN, Fox News, Buzzfeed News, Talking Points Memo, National Review, the Guardian, New York Post, Reuters, NPR, Vox and Washington Post. But this dataset contained articles from the year 2016 to July 2017, thus, only those articles which were published from October to December 2016 were filtered to match with the previous fake dataset time frame. This dataset had the following columns: id, title, publication, author, date, year, month, url, content. Again only the column ‘content’, renamed as ‘title’ for ease of use, were considered. As all the articles from this dataset are reiable articles, a new column ‘label’ was created with every article labeled as 0 for indicating it is reliable.

These two datasets were then combined together which will be referred as the Kaggle Combined dataset in the later sections of the thesis. A random shuffling was performed to ensure that reliable and fake articles are split evenly across the length of the dataset.

Table 3.2 shows the distribution among the fake and real articles for the Kaggle Combined dataset:

Type of News Article	**Distribution**	**Average count of words in ‘text’ column**	**Average count of words in ‘text’ column after data cleansing**
Reliable - (0)	15712	957	822
Fake - (1)	12953	640	535

Table 3.2 - distribution of fake and real articles for the Kaggle Combined dataset

3.1.3 Third dataset description:

The third dataset was also collected from a Kaggle website. Here the fake news articles were collected from BuzzFeed site and reliable articles were collected from Politifact site. This is a relatively very small dataset of just 422 articles in total. This dataset has the following columns: id, title, text, url, top_img, authors, source, publish_date, movies, images, canonical_link, meta_data. Again only the 'text' column was used for analysis. A new column 'label' was added with articles from Buzzfeed being labeled as 1 and articles from Politifact were labeled as 0. This dataset will be referred as the Politifact Buzzfeed dataset in the later sections.

Table 3.3 shows the distribution among the fake and real articles for the third politifact buzzfeed dataset:

Type of News Article	**Distribution**	**Average count of words in 'text' column**	**Average count of words in 'text' column after data cleansing**
Reliable - (0)	211	624	532
Fake - (1)	211	563	482

Table 3.3 distribution of fake and real articles for the third politifact buzzfeed dataset

All three datasets had almost equal number of articles proving to have a very balanced nature of data.

3.2 Data Preparation:

It is a well-known fact in any data science lifecycle that the data preparation part will often take up to 50% of the entire project's time and that this is the most important step, as clean data will ensure that the data is consistent and reliable. There is no point in building models with an unreliable and inconsistent data with a lot of missing values, outliers, etc.

The following section describes the data preparation steps that were performed:

3.2.1 Handling missing data and Lowercasing

Missing values from the ‘text’ column were removed completely, as text data cannot be assumed or treated the way a missing numeric data could have been dealt with.

The first and foremost step was to lowercase all the text. It is applicable to most NLP problems and helps with maintaining consistency of expected output, in order to guarantee that all tokens/text map to the same corresponding feature irrespective of their casing. (Gokulakrishnan, Priyanthan, Ragavan, Prasath, & Perera, 2012). For Example, A word embedding model that is trained for similarity lookups, which has different variation in input capitalization (e.g. ‘Canada’ vs. ‘canada’) will give different types of output or no output at all. This could probably happen because the dataset has mixed-case occurrences of the word ‘Canada’ and there could be insufficient instances for the neural-network to learn the weights of the uncommon version. This type of issue is bound to happen when the dataset is fairly small, and lowercasing is a great way to deal with sparsity issues.[6]

3.2.2 Dealing with Duplicates

Next step was to check if there are any duplicates in the text. Having duplicates will only make the classifier to learn more about the same text and assign more weight to the words that are occurring often in the documents. 539 records were found to be duplicated, out of which 382 records were dropped, with the first occurrence of every duplicate being retained. It is better to remove duplicates in the beginning, before beginning any pre-processing steps as it will shorten the time taken to perform the other steps.

3.2.3 Regular Expressions – Punctuation, Numbers and Alphanumeric texts removal

A regular expression is a pattern consisting of a sequence of characters that matches with a given text. For example, two occurrences of the same word - ‘possible’ without punctuation, and ‘possible!!’ with punctuation could make the model to figure out that they are two separate tokens and will also result in a lot of noisy data. Presence of numbers and alphanumeric texts could make our model to overfit this dataset and not the actual problem in the real world. Moreover, numbers and alphanumeric texts will not add any real value to the actual context of news.

3.2.4 Named Entity Recognition – Spacy

During the Sixth Message Understanding Conference (MUC-6) (R. Grishman & Sundheim 1996), while performing Information Extraction (IE) tasks where structured information of company activities and defense related activities were extracted from unstructured text, it was essential to categorize information in units if it is a person, or a location, or an organization, or an absolute or relative date such as Tuesday, November
which are absolute dates, whereas ‘yesterday, ‘today’, ‘last year’ can be
considered as relative dates. Identifying a category or reference to these entities in text came to be known as “Named Entity Recognition and Classification” (Nadeau & Sekine, 2007).

3.2.5 Removal of Stop words and Non-English words

If non-english words and junk words that have absolutely no meaning are present in the data before modelling, it will only make the model to overfit and over-learn this particular dataset, as all the other datasets will contain only English words. Thus, removing non-english word becomes an important step.

6 https://www.kdnuggets.com/2019/04/text-preprocessing-nlp-machine-learning.html

Stop words are frequently occurring words such as ‘a,are,the,is’ which are only connector words but won’t really add any value to convey if an article is fake or not. The only time it is not advisable to remove stop words is during sentiment analysis. The word ‘not’ is a very valuable indicator while performing sentiment analysis.

3.2.6 Lemmatization

Lemmatization involves the use of a vocabulary and analyses words morphologically. This aims to remove inflectional endings and returns the base or root form of a word, which is the *lemma*. For example, consider the word ‘better’ and ‘best’, the root form of the word ‘good’ will be returned. The word ‘saw’ could be returned as ‘see’ depending upon the verb tense of the word. There is another process known as Stemming, but this is the process of chopping off the end of the words to their root form (eg. trouble, troubled, troubles) will simply be cut off to (e.g. troubl). The “root” in the case of Stemming may not be a real root word. It is always better to use Lemmatization compared to Stemming.

3.2.7 Text Feature Extraction Techniques:

Usually in machine learning, we have numeric data as inputs for training the models, as machine learning models can only deal with numeric inputs. Therefore, before beginning to apply machine learning techniques on text data, text has to be first transformed in a way that can be handled by the algorithm. Text data are converted into vectors, which are lists of numbers with some encoded information within them. This process of converting text to vectors is known as vectorization.

3.2.7.1 Term Frequency – Inverse Document Frequency

This method tells us the relative importance of the word in a corpus of documents, based on its frequency of occurrence in the document. Term frequency is the count or measure of how frequently a word has occurred in one document. Inverse Document frequency is used for finding out importance of the word across all the documents in a corpus. There could be words that occur rarely but which are informative of the context, but the importance of it could be lost if we base the importance of a word only on the number of times it has occurred. There is a possibility that frequently occurring words could have lesser value. Words that occur across all documents equally indicate an higher importance and thus it will get a high TF-IDF value.

3.2.7.2 Word Embedding - Word2vec

Word Embeddings are vector representations of a text in n-dimensional space. Each word is encoded as a vector. The main objective behind word embedding is to cluster words having similar meaning together, that is, if we wish to visualize the words in a feature vector space, words having similar meaning will have close spatial positions. The angle between these similar word vectors will be close to 0.

Word2vec converts the input text and produces a corresponding vector space for each word, each consisting of n-dimensions. These word vectors will be positioned in the vector space in such a way that words that have similar contexts/meaning in the text, to be closely located in the vector space (Rexha, Kröll, Kern, & Dragoni, 2017). Traditional classification models use bag of words technique, which actually reduces text into frequency of occurrence counts per document in a corpus of documents. Whereas, word2vec learns semantic similarity between words and the actual meaning of a word, in an unsupervised method, making use of a contextual window. It also performs way faster compared to other methods. (Major, Surkis & Aphinyanaphongs, 2018). Two use cases of word2vec can be achieved. Given a set of continuous words in a sentence, the next possible word could be guessed using Word2vec and the reverse is also possible, given one word, the consecutive set of words in a sentence can also be found. In Word embeddings, the position of a word within the vector space is learned from text and is based on the words that surround the word when it is used. For fake-news detection, using word embeddings makes more sense, as understanding the contextual meaning of a word becomes highly important. A word separately could be of a different meaning and a word surrounded by a group of words in a paragraph, could have slightly different meaning.

3.2.7.3 Sentence Embedding – Universal Sentence Encoder

Sentence embedding is a seemingly extended version of word embeddings, where sentences are mapped to vectors of real numbers. This method embeds or rather transforms an entire sentence into vector space. Sentence embeddings retain some of the features from the underlying word embeddings. The Universal Sentence Encoder converts group of texts into high n-dimensional vector representations, which can be used for clustering, semantic similarity, text classification, , and other NLP tasks. The model is trained and optimized for greater-than-word length text, such as sentences, phrases or short paragraphs. It is trained on a variety of data sources and a variety of tasks to automatically perform a large variety of NLP tasks. The universal sentence encoder is built on a deep averaging network architecture encoder.

3.3 Modeling

The model that was chosen to carry out in this thesis was Linear Support Vector Machine Classifier, as this model had performed the best in terms of accuracy after reviewing previous literature in this area of research. The results obtained by (Ahmed, Traore & Saad, 2017) while using TF-IDF as text feature representation technique, and LSVM as a classifier, was an accuracy of 92%, which was the highest accuracy achieved compared to all the other research works in fake news detection. Thus, using TF-IDF along with LSVM forms as the baseline model for this research, to recreate the previous researchers work. LSVM classifiers are known for generalizing well in high dimensional spaces (Larson et al., 2002). LSVMs are promising as they are highly accurate, gets trained quickly in lesser time and evaluates quickly. (Dumais, Platt, Heckerman & Sahami, 1998).

The next models as proposed in this study, were built using word embedding word2vec as a text feature extraction technique with LSVM and sentence embedding model Universal Encoder as a text feature extraction technique with LSVM.

The accuracy results obtained by all three models and were compared which will be discussed in the next section.

3.4 Evaluation

The results of the research are evaluated using Accuracy measures, ten k-fold cross-validation scores, classification error, Recall, Precision, performance metrics calculated from the confusion matrix. To compare the differences obtained by each model, normality and statistical tests were performed to see if the differences were actually statistically significant, in order to accept or reject the hypothesis. The normality testing was carried out on the results of the cross validation score of each model, using Shapiro-Wilk test. Based on the result of the normality test, if the distribution of scores were normal or Gaussian, student's t-tests were performed. If distribution was not normal, Mann-Whitney U test was performed.

Accuracy

It is a measure of the correct number of predictions to the total number of prediction in the data. Accuracy is highly reliable for balanced datasets.

Accuracy = Correct Predictions/Total Predictions

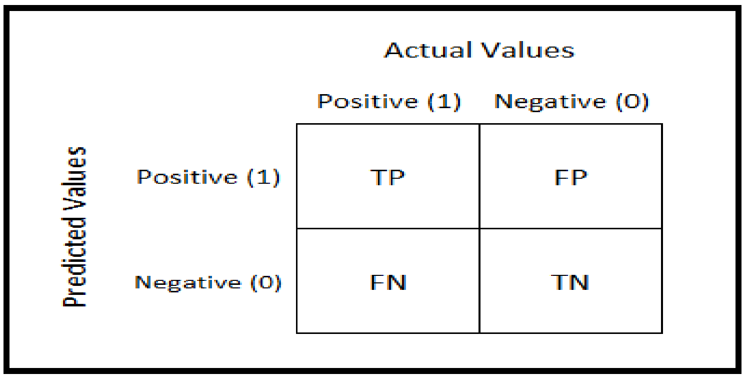
Cross-validation

The main purpose of using cross-validation is to test the model using validation dataset during training phase to curb the problems of underfitting, overfitting and to see how the model generalizes on testing dataset and also to see the model's performance on a completely new unseen dataset.

Here, the K-fold cross-validation technique (k =10) was used to evaluate the models. It is nothing but a repeated holdout method and the scores are averaged after all the holdouts are completed. In this method, every chunk of data goes into the validation set exactly once, and also goes into the training set k-1 times (9 times). This helps in reducing underfitting as all the data is used for fitting, and reduces overfitting as every data is used in validation set.

Confusion Matrix – Precision, Recall, F1-Measure Scores

It is a performance measurement table with four combinations of predicted and actual values of a classifier model. It displays the number of correct (predicted and actual value being the same) and incorrect (predicted and actual value are not same) predictions made by the model.



		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

Figure 3.7 – Confusion Matrix

Recall

$$\boxed{\textit{Recall} = \frac{TP}{TP + FN}}$$

Figure 3.8 – Recall

It is also known as True Positive Rate. It is the measure of all the positive classes that were predicted correctly by the model. Recall is referred to as the completeness of the model.

Precision

$$\boxed{\textit{Precision} = \frac{TP}{TP + FP}}$$

Figure 3.9 – Precision

It is a measure of the number of positive classes that were predicted correctly, how many are actually positive classes.

F1 Score

F1 score is an harmonic mean of recall and precision. A high F1 score indicates that the accuracy level of the model is very good.

3.5 Software Used

The thesis was completely written using Python. For most parts, Jupyter Notebook was used to run the experiments, as it was very convenient to code locally. For some reason, a privacy error kept occurring while running the sentence embedding using Universal Sentence Encoder in Jupyter Notebook. Hence, only for running the LSVM model using sentence embedding, Google Collaboratory was used to run the experiment. Libraries such as Pandas, NumPy, SciPy, Scikit-learn, NLTK, Gensim, Tensorflow hub, matplotlib, seaborn were primarily used to run the scripts used in this thesis.

3.6 Strengths and Limitations

One of the major strengths of this thesis were that the datasets used were all balanced, which means that the model has the capacity to learn both the scenarios equally and has the capability to learn the instances correctly. Type I and Type II errors are mostly the result of having samples that don’t really represent the real world, ideally occurs more because datasets are usually imbalanced.

Word embeddings and sentence embeddings were never used in fake news detection research area before, which was why this was identified as the major research gap after a thorough literature review. Because embedding techniques usually capture the contextual meaning of a word in a corpus of text, exploring the usage of these text representation techniques gives a good area of scope for performing meaningful text analysis.

Ample amount of time was spent in data understanding, to ensure that the data preparation actually makes sense to the datasets at hand, rather than just performing the data preparation that is done usually, which may not apply commonly to every dataset.

One major limitation that was identified was that Linear SVM’s default hyperparameters were only used, as there was a necessity to recreate exactly what the previous researchers had done in their paper. Another concern was that if any hyperparameter were tweaked, the accuracy usually went down, hence the hyperparameters were left to be default.

CHAPTER 4
IMPLEMENTATION AND RESULTS

This chapter describes how each step that was already defined in Chapter 3, were carried out to implement the research question.

4.1 Data Preparation:

This section intends to continue from the data preparation steps that were explained in chapter 3, to give the reason why these particular data preparation steps were performed. As the answer for everything in the world of Information Technology is usually "It depends on the situation at hand", this research also performed these following data preparation steps based on exploring the dataset in detail.

4.1.1 Removing Null values and Lowercasing

'NA' values were discovered in text column and these values were completely removed by using the drop.na() method.

The first and foremost step after treating null values, was to lowercase all the text. It particularly helps while performing TF-IDF, word embeddings and sentence embeddings tasks, as two words having different casing style might be treated differently.

4.1.2 Removing Duplicates

Next step was to remove the duplicate records in the text. 539 records were found to be duplicated, out of which 382 records were dropped, with the first occurrence of every duplicate being retained.

4.1.3 Regular Expressions – Removing Punctuation, Numbers, Alpha-numeric texts

While exploring the dataset, a lot of inconsistent text were found, such as numbers/digits, meaningless alphanumeric texts, words with punctuations, non-english words were present in the data. Punctuations were removed with the help of regular expressions. Any symbol except full stop '.' that is not a letter were removed. As a full stop indicates the ending and beginning of a sentence and being able to recognize sentences was important while performing word and sentence embeddings. Thus, full stops were not removed.

Numbers and alphanumeric texts were removed next. Before the numbers were removed, an interesting insight was found after executing the baseline model using TF-IDF, that the year '2016' and months 'october' and 'november' were among the top weighted words, found using the package 'eli5' – 'explain like I am 5 years old'. If this same model is used on other such datasets which have news belonging to years before 2016 or after 2016, it could potentially make our model to learn only for 2016 in the data and could result in misclassification for other years. Hence, it makes little sense to have years and dates in the text.

4.1.4 Named Entity Recognition – Spacy

It is best to limit the model's knowledge of the people and organizations mentioned in the article text. Otherwise, there is a risk with the model simply learning patterns of text in this dataset, for example, 'Clinton stressed upon' which describe the topic and viewpoint of the text, rather than focusing on the main outcome (is this text fake or not). Additionally, these patterns will be highly sensitive to the particular news cycle. To overcome this scenario, Spacy [7] is used for performing Named Entity Recognition to replace all mentions of named entities with its entity tag, e.g. Hilary Clinton will be replaced as PERSON, Google will be replaced as ORG. (Gilda, 2017)

[7] (Explosion, Spacy, Sep. 2017) https://github.com/explosion/spaCy

4.1.5 Removal of Non-English words and Stop words

After performing named entity recognition, as mentioned in the data understanding section, many non-english words and junk words that have absolutely no meaning were found. Any word that was not found in nltk package's English words corpus collection was considered to be a non-english or meaningless word and were thus removed. Consider texts like "happpppiiiiiieeeeeee" "haffunn" cannot be converted to their actual meaning and so they have to be removed. An additional regex to retain only English alphabets is also performed before this step.

While creating the tfidf using TfidfVectorizer method available in sklearn.feature_extraction.text package, stop words can be removed directly by including it as parameter stop_words='english'.

4.1.6 Lemmatization

The WordNetLemmatizer package from nltk.stem library was used to perform the lemmatization of words.

4.1.7 An experimental example:

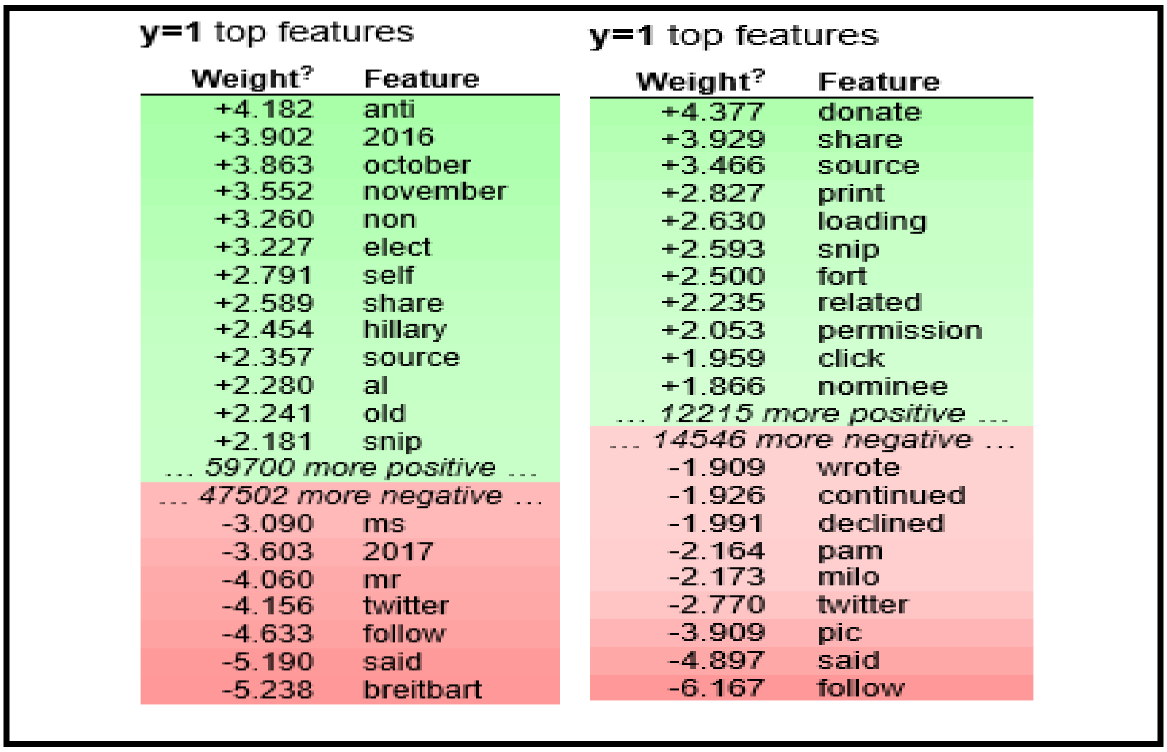
This section shows how important the data pre-processing step is and the significant effects of overfitting in accuracy. Accuracy result achieved using TF-IDF before and after data-preprocessing using Linear SVM model on the Kaggle Competition dataset can be seen in Table 4.1:

Pre-processing Stage	**Accuracy**
Before	0.96
After	0.90

Table 4.1

Figure 4.1 displays the features (texts) and their weights that contributed the most to classifying the news as fake or reliable. This was generated by using the eli5 (explain like I am 5) library. y=1 indicates that the top positive scores indicated in green color, belongs to fake articles and the red color belongs to reliable articles.

Before: **After:**



y=1 top features

Weight?	Feature
+4.182	anti
+3.902	2016
+3.863	october
+3.552	november
+3.260	non
+3.227	elect
+2.791	self
+2.589	share
+2.454	hillary
+2.357	source
+2.280	al
+2.241	old
+2.181	snip
... 59700 more positive ...	
... 47502 more negative ...	
-3.090	ms
-3.603	2017
-4.060	mr
-4.156	twitter
-4.633	follow
-5.190	said
-5.238	breitbart

y=1 top features

Weight?	Feature
+4.377	donate
+3.929	share
+3.466	source
+2.827	print
+2.630	loading
+2.593	snip
+2.500	fort
+2.235	related
+2.053	permission
+1.959	click
+1.866	nominee
... 12215 more positive ...	
... 14546 more negative ...	
-1.909	wrote
-1.926	continued
-1.991	declined
-2.164	pam
-2.173	milo
-2.770	twitter
-3.909	pic
-4.897	said
-6.167	follow

Figure 4.1

A potential reason behind this drop in accuracy could be the famous problem of overfitting. Overfitting occurs when the model learns too much about the dataset and less about the underlying problem that the data represents in the world. Figure 4.2 shows a clear differentiation between underfitting and overfitting of a model:

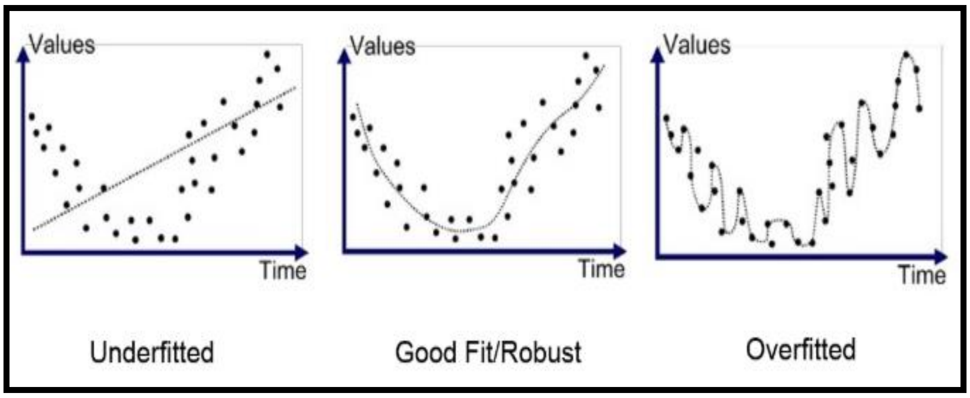


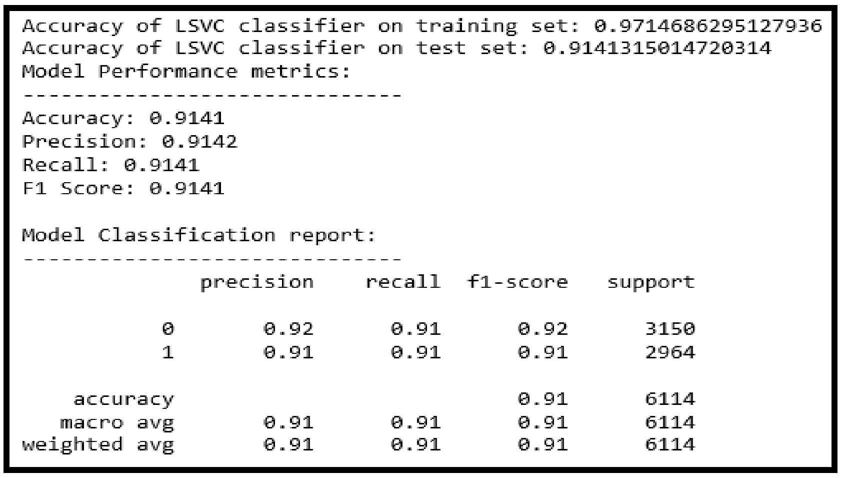
Figure 4.2

4.2 Modeling

4.2.1 Baseline Model

The baseline model for this research is to reproduce the results from the previous research by (Ahmed, Traore & Saad, 2017) which consists of performing Linear SVC model with TF-IDF as the text representation technique. Linear SVC The researchers achieved an accuracy of 92% by using Linear SVC by using TF-IDF with uni-gram technique by selecting only the top 50,000 features. Almost similar results of 91% testing accuracy and 97% training accuracy is achieved in this research by the baseline model on the Kaggle competition dataset, by using Linear SVC and TF-IDF uni-gram as text representation technique. This was implemented using sklearn.svm's LinearSVC method. The second Kaggle combined dataset achieves an accuracy of 90%. With the third politifact buzzfeed dataset although the training accuracy is 85%, prediction with the test set is very poor with an accuracy of 43%, this could be because the dataset size is very small, compared to the other two datasets. A 70:30 split was done to divide the datasets into train and test dataset, by using sklearn.model_selection package's train-test-split method.

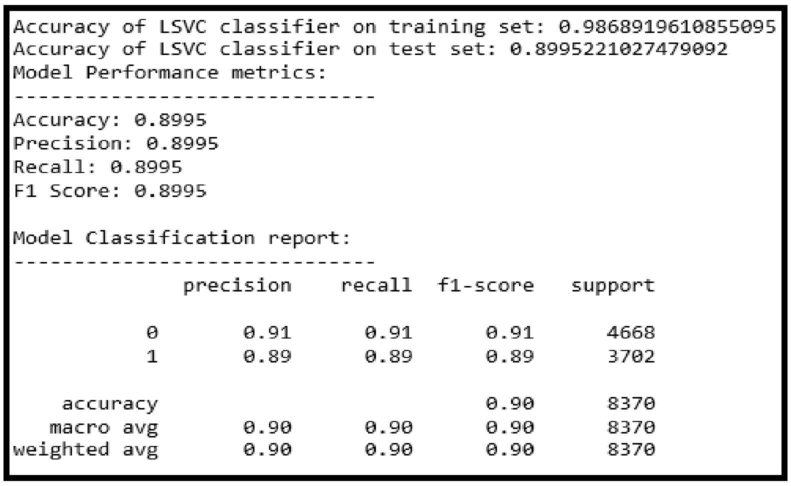
Classification Report for Kaggle Competition dataset's TF-IDF based LSVM model can be seen in Figure 4.3:



```
Accuracy of LSVC classifier on training set: 0.9714686295127936
Accuracy of LSVC classifier on test set: 0.9141315014720314
Model Performance metrics:
------------------------------
Accuracy: 0.9141
Precision: 0.9142
Recall: 0.9141
F1 Score: 0.9141

Model Classification report:
------------------------------
              precision    recall  f1-score   support

           0       0.92      0.91      0.92      3150
           1       0.91      0.91      0.91      2964

    accuracy                           0.91      6114
   macro avg       0.91      0.91      0.91      6114
weighted avg       0.91      0.91      0.91      6114
```

Figure 4.3

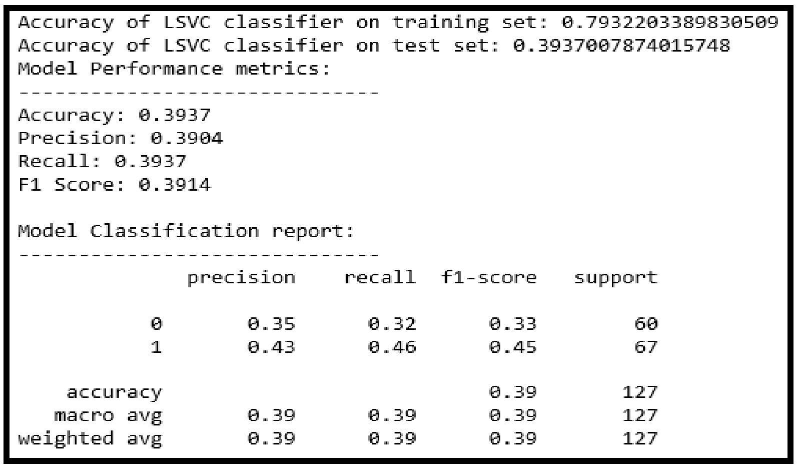
Classification Report for Kaggle Combined dataset's TF-IDF based LSVM model can be seen in Figure 4.4:



```
Accuracy of LSVC classifier on training set: 0.9868919610855095
Accuracy of LSVC classifier on test set: 0.8995221027479092
Model Performance metrics:
------------------------------
Accuracy: 0.8995
Precision: 0.8995
Recall: 0.8995
F1 Score: 0.8995

Model Classification report:
------------------------------
              precision    recall  f1-score   support

           0       0.91      0.91      0.91      4668
           1       0.89      0.89      0.89      3702

    accuracy                           0.90      8370
   macro avg       0.90      0.90      0.90      8370
weighted avg       0.90      0.90      0.90      8370
```

Figure 4.4

Classification Report for Politifact Buzzfeed dataset's TF-IDF based LSVM model can be seen in Figure 4.5:



```
Accuracy of LSVC classifier on training set: 0.7932203389830509
Accuracy of LSVC classifier on test set: 0.3937007874015748
Model Performance metrics:
------------------------------
Accuracy: 0.3937
Precision: 0.3904
Recall: 0.3937
F1 Score: 0.3914

Model Classification report:
------------------------------
              precision    recall  f1-score   support

           0       0.35      0.32      0.33        60
           1       0.43      0.46      0.45        67

    accuracy                           0.39       127
   macro avg       0.39      0.39      0.39       127
weighted avg       0.39      0.39      0.39       127
```

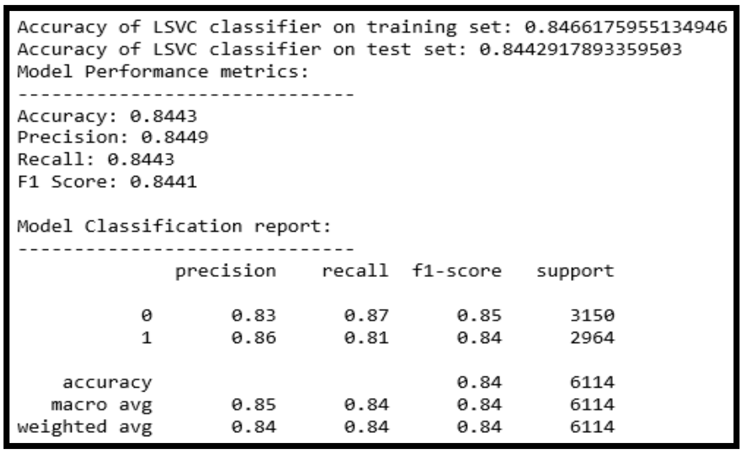
Figure 4.5

4.2.2 Linear SVM with Word Embeddings

The main goal of this research was to see if using word embeddings as the text representation technique will perform better compared to using TF-IDF technique. The word2vec model is trained on the overall 'text' column data. The text data should be in the form of list of lists. First, every document is split into sentences. A full stop indicates the beginning and end of a sentence. Then, every word inside each of the sentences are tokenized, thus forming a list of lists. For each word of the text, the Word2Vec vector representation is extracted. The training data is then constructed by creating an average vector over the entire news text (Rexha, Kröll, Kern, & Dragoni, 2017). Each word is an object and a sentence is a set of these objects. If vectors of this news set is close to each other in space, then the average of this will be a good representation of the tweet. This average vector is then used as input for training the LSVM classifier.

After creating word vectors using Gensim's Word2Vec which was trained on Google news, the training accuracy was 85% and test accuracy was 84% for the Kaggle Competition dataset. The training accuracy was 82% and test accuracy was 83% for the Kaggle Combined dataset. The training and test accuracy was 56% for the Politifact buzzfeed dataset.

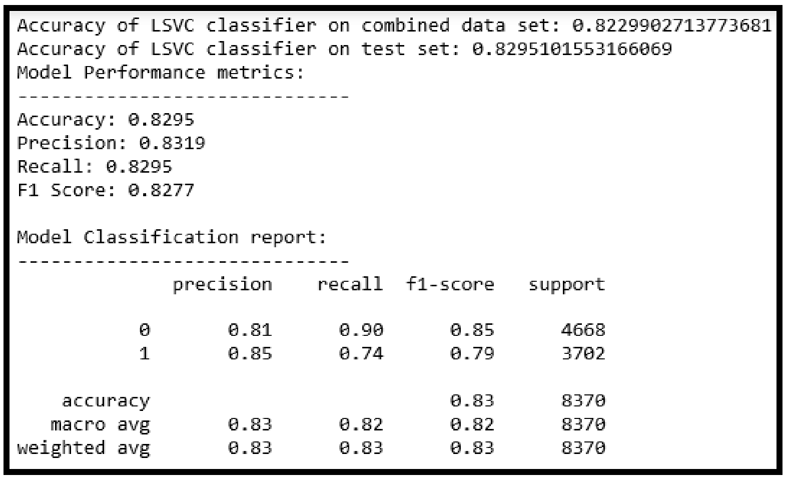
Classification Report for Kaggle Competition dataset's word embeddings based LSVM model can be seen in Figure 4.6:



```
Accuracy of LSVC classifier on training set: 0.8466175955134946
Accuracy of LSVC classifier on test set: 0.8442917893359503
Model Performance metrics:
------------------------------
Accuracy: 0.8443
Precision: 0.8449
Recall: 0.8443
F1 Score: 0.8441

Model Classification report:
------------------------------
              precision    recall  f1-score   support

           0       0.83      0.87      0.85      3150
           1       0.86      0.81      0.84      2964

    accuracy                           0.84      6114
   macro avg       0.85      0.84      0.84      6114
weighted avg       0.84      0.84      0.84      6114
```

Figure 4.6

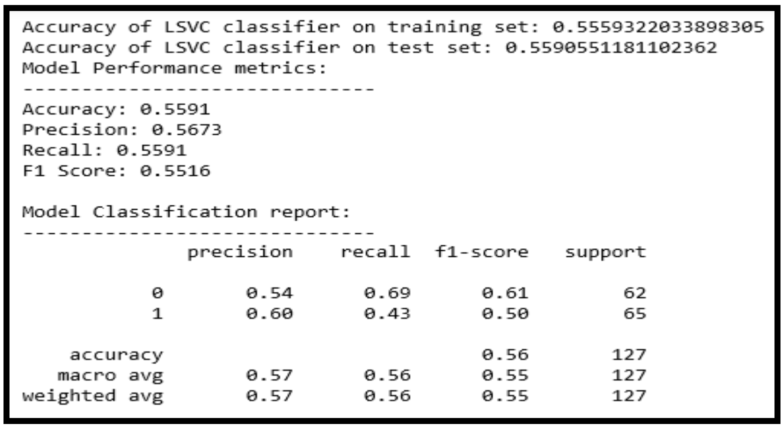
Classification Report for Kaggle Combined dataset's word embeddings based LSVM model can be seen in Figure 4.7:



```
Accuracy of LSVC classifier on combined data set: 0.8229902713773681
Accuracy of LSVC classifier on test set: 0.8295101553166069
Model Performance metrics:
------------------------------
Accuracy: 0.8295
Precision: 0.8319
Recall: 0.8295
F1 Score: 0.8277

Model Classification report:
------------------------------
              precision    recall  f1-score   support

           0       0.81      0.90      0.85      4668
           1       0.85      0.74      0.79      3702

    accuracy                           0.83      8370
   macro avg       0.83      0.82      0.82      8370
weighted avg       0.83      0.83      0.83      8370
```

Figure 4.7

Classification Report for Politifact dataset's word embeddings based LSVM model can be seen in Figure 4.8:



```
Accuracy of LSVC classifier on training set: 0.5559322033898305
Accuracy of LSVC classifier on test set: 0.5590551181102362
Model Performance metrics:
------------------------------
Accuracy: 0.5591
Precision: 0.5673
Recall: 0.5591
F1 Score: 0.5516

Model Classification report:
------------------------------
              precision    recall  f1-score   support

           0       0.54      0.69      0.61        62
           1       0.60      0.43      0.50        65

    accuracy                           0.56       127
   macro avg       0.57      0.56      0.55       127
weighted avg       0.57      0.56      0.55       127
```

Figure 4.8

4.2.3 Linear SVM with Sentence Embeddings

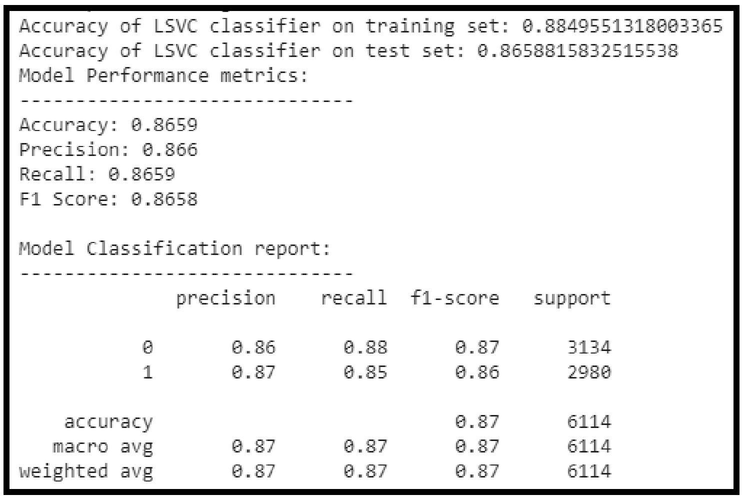
The code to run sentence embeddings alone had to be run in Google Collaboratory, as the tensorflow embedding module runs into privacy blockage issues while running using Jupyter Notebook. Creating sentence embeddings are fairly straightforward compared to creating word embeddings. The embedder was downloaded from [8], using Tensorflow's hub.module. The embedder preprocesses the data by itself, hence it does not require intensive preprocessing of text. For embedding the text, the data was processed as 10 separate chunks and appended in the end, to ensure not into memory and time out issues. Due to too many time out issues, this step was the only way to create sentence embeddings successfully. After creating the embeddings, similar to previous models, the data was split into train and test and the Linear SVC model was run, with the following results:

Classification Report for Kaggle Competition dataset's Sentence embeddings based LSVM model can be seen in Figure 4.9:

Classification Report for Kaggle Combined dataset's Sentence embeddings based LSVM model can be seen in Figure 4.10:

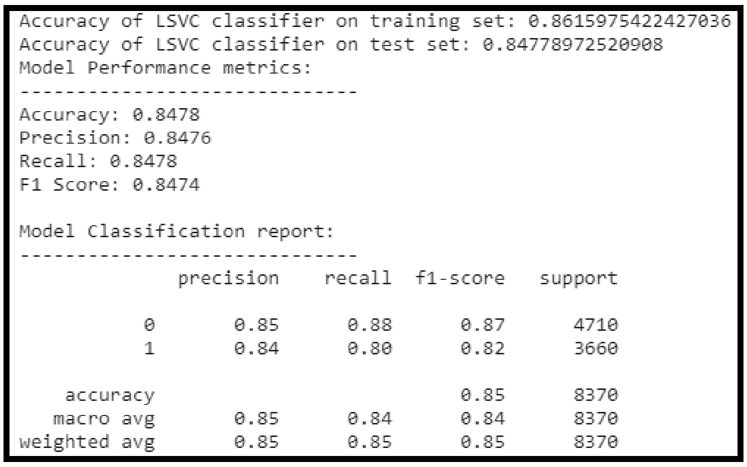
[8] https://tfhub.dev/google/universal-sentence-encoder/2

Classification Report for Politifact Buzzfeed dataset's Sentence embeddings based LSVM model can be seen in Figure 4.11:



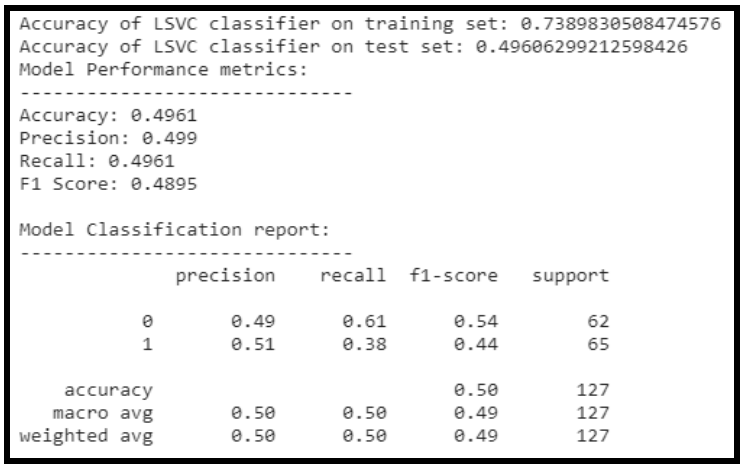
```
Accuracy of LSVC classifier on training set: 0.8849551318003365
Accuracy of LSVC classifier on test set: 0.8658815832515538
Model Performance metrics:
------------------------------
Accuracy: 0.8659
Precision: 0.866
Recall: 0.8659
F1 Score: 0.8658

Model Classification report:
------------------------------
              precision    recall  f1-score   support

           0       0.86      0.88      0.87      3134
           1       0.87      0.85      0.86      2980

    accuracy                           0.87      6114
   macro avg       0.87      0.87      0.87      6114
weighted avg       0.87      0.87      0.87      6114
```

Figure 4.9



```
Accuracy of LSVC classifier on training set: 0.8615975422427036
Accuracy of LSVC classifier on test set: 0.84778972520908
Model Performance metrics:
------------------------------
Accuracy: 0.8478
Precision: 0.8476
Recall: 0.8478
F1 Score: 0.8474

Model Classification report:
------------------------------
              precision    recall  f1-score   support

           0       0.85      0.88      0.87      4710
           1       0.84      0.80      0.82      3660

    accuracy                           0.85      8370
   macro avg       0.85      0.84      0.84      8370
weighted avg       0.85      0.85      0.85      8370
```

Figure 4.10



```
Accuracy of LSVC classifier on training set: 0.7389830508474576
Accuracy of LSVC classifier on test set: 0.49606299212598426
Model Performance metrics:
------------------------------
Accuracy: 0.4961
Precision: 0.499
Recall: 0.4961
F1 Score: 0.4895

Model Classification report:
------------------------------
              precision    recall  f1-score   support

           0       0.49      0.61      0.54        62
           1       0.51      0.38      0.44        65

    accuracy                           0.50       127
   macro avg       0.50      0.50      0.49       127
weighted avg       0.50      0.50      0.49       127
```

Figure 4.11

4.3 Evaluation:

This section will present the mean test accuracy scores and all the individual n-fold scores obtained by using 10-fold Cross Validation method on all three datasets. This was done using the methods: cross_val_score, cross_val_predict and KFold available in sklearn's model_selection library in python. As the precision, recall, F1 scores are almost in the same range as Accuracy scores, which can be seen from the classification reports in Figures 4.3 to 4.11, only the Accuracy score has been considered for evaluation and for comparing all three model's performance. As all three datasets had a balanced nature, Accuracy score can be used as a good metric for comparison.

4.3.1 10-fold Cross-Validation Mean Scores for all three datasets:

From the Table 4.3, it is evident that the TF-IDF based LSVM model overall performs the best in terms of accuracy. The sentence embedding based LSVM model is the second best and the word embedding based LSVM model performs the least. Only for the Politifact Buzzfeed dataset, TF-IDF based model performs poorly and word embeddings based model performs well compared to the other two. The highest accuracy achieved by a model for a particlar dataset is highlighted in bold.

Dataset	**TF-IDF**	**Word Embedding**	**Sentence Embedding**
Kaggle Competition	**0.92**	0.85	0.87
Kaggle Combined	**0.89**	0.82	0.85
Politifact Buzzfeed	0.39	**0.53**	0.50

Table 4.2

4.3.2.1 Individual scores of 10-fold Cross-Validation for Kaggle Competition Dataset:

It is evident from Figure 4.12 that tf-idf performs really well compared to the other two text representation techniques based models. Sentence embedder based model slightly performs better than the word embedder based model.

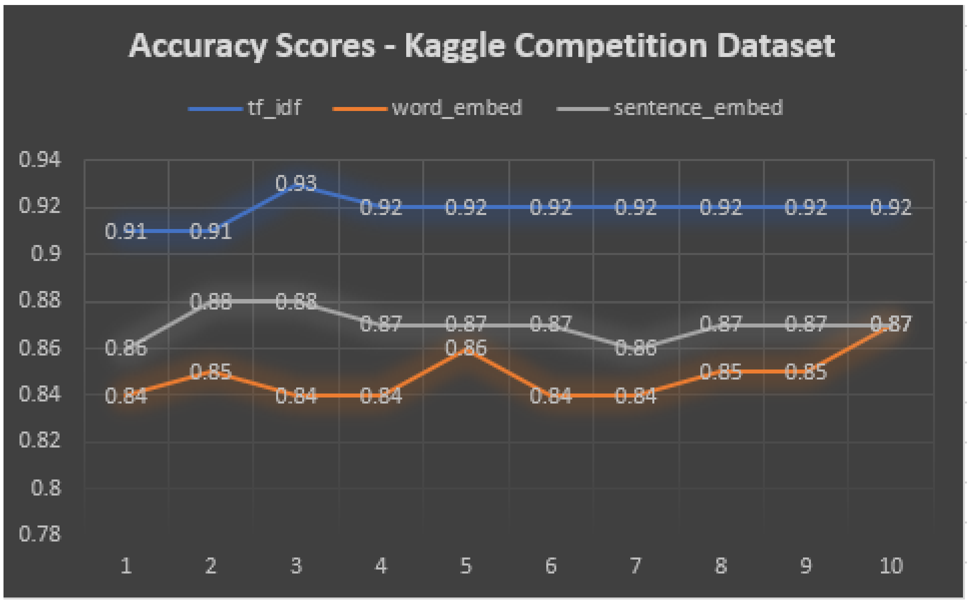


Figure 4.12

4.3.2.2 Individual scores of 10-fold Cross-Validation for Kaggle Combined Dataset:

It is again evident from Figure 4.13 that tf-idf performs well again compared to the other two text representation techniques based models. Again Sentence embedder based model performs better than the word embedder based model.

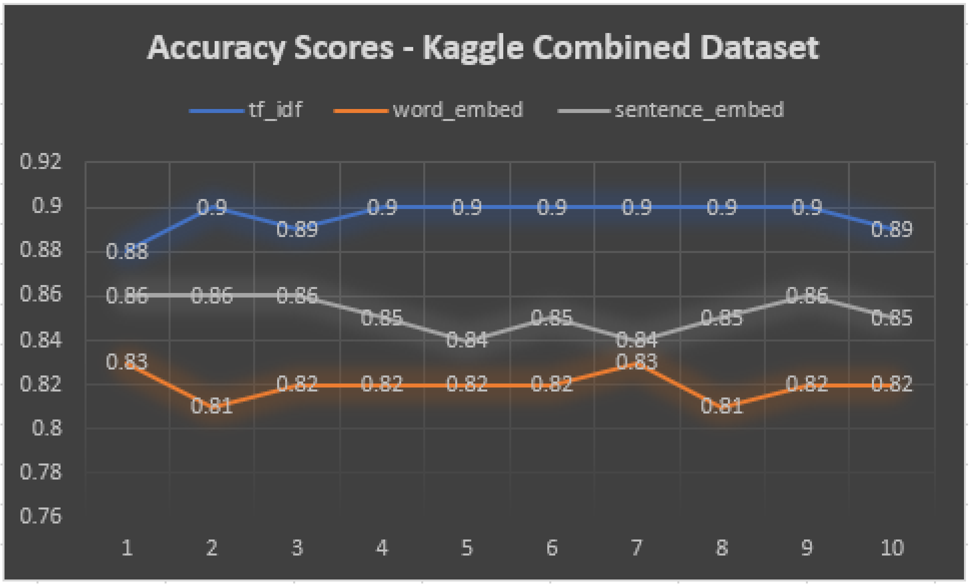


Figure 4.13

4.3.2.3 Individual scores of 10-fold Cross-Validation for Politifact Buzzfeed Dataset:

For a very small dataset such as Politifact Buzzfeed, surprisingly word embedder based model outperforms the other two models. It could be deduced that as TF-IDF technique relies on the frequency of occurrence of a word, for a small dataset, the number of times that a word appeared could be lesser, hence the TF-IDF based model performs the least for this dataset.

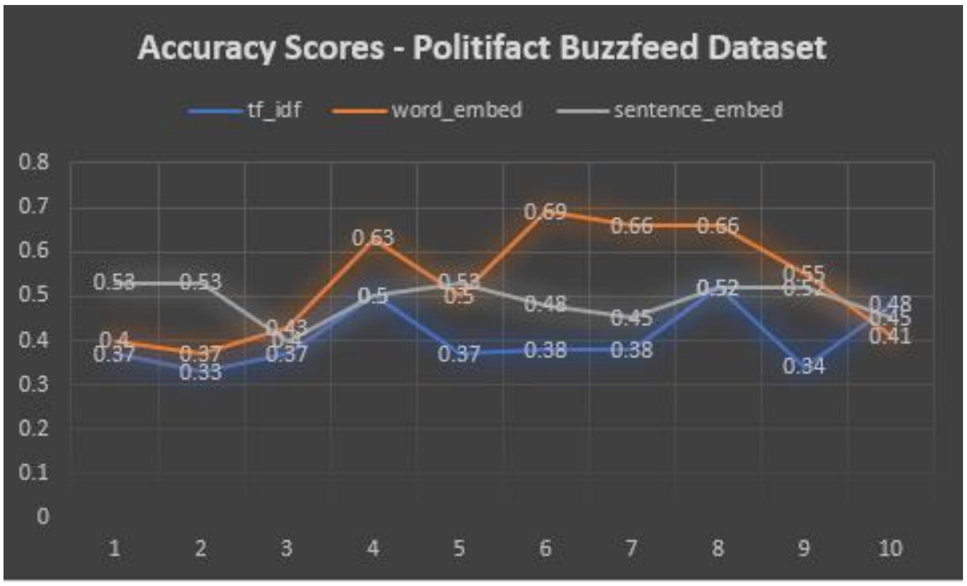


Figure 4.14

4.3.3.1 Confusion matrix scores for the Kaggle Competition Dataset:

From the confusion matrix in Figure 4.15 and Table 4.4, it can be seen that all three models have classified True Positive in more or less the same range, which is class ‘0’ - reliable articles. For True Negative, TF-IDF’s prediction rate is the highest. Word embedder based model seems to classify articles as positive mostly, as it has the highest False Positive rate and a good enough True Positive rate.

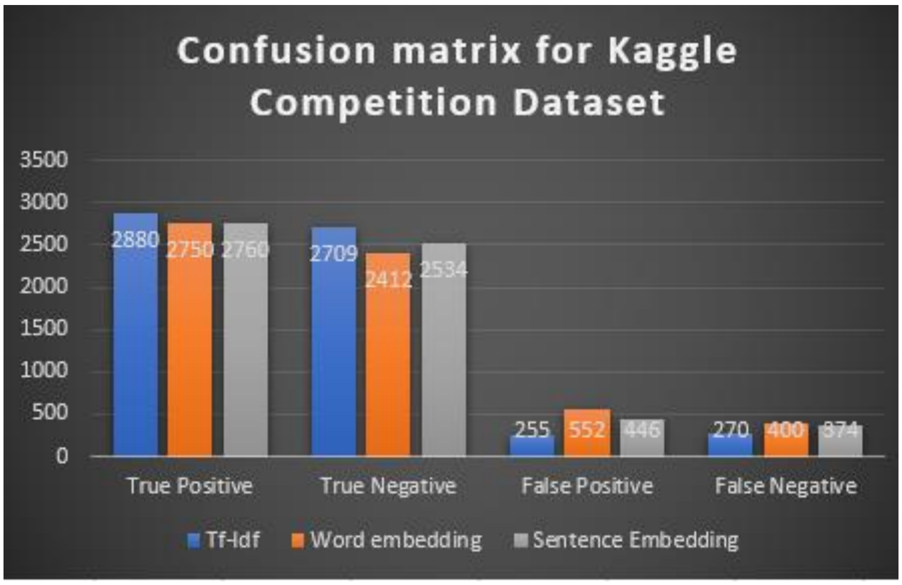


Figure 4.15

Metric	TF-IDF	Word embedding	Sentence Embedding
True Positive	2880	2750	2760
True Negative	2709	2412	2534
False Positive	255	552	446
False Negative	270	400	374

Table 4.3

4.3.3.2 Confusion matrix scores for the Kaggle Combined Dataset:

From the confusion matrix in Figure 4.16 and Table 4.5, it can be seen that all three models have classified articles as True Positive in more or less the same range, which is class ‘0’ - reliable articles. For True Negative, again TF-IDF’s rate is the highest. There is a significant difference between the embedder based models and the TF-IDF based model’s True Negative rate. Word embedder based model again classifies articles as positive mostly, with having a very high False Positive rate and a significantly low True Negative rate.

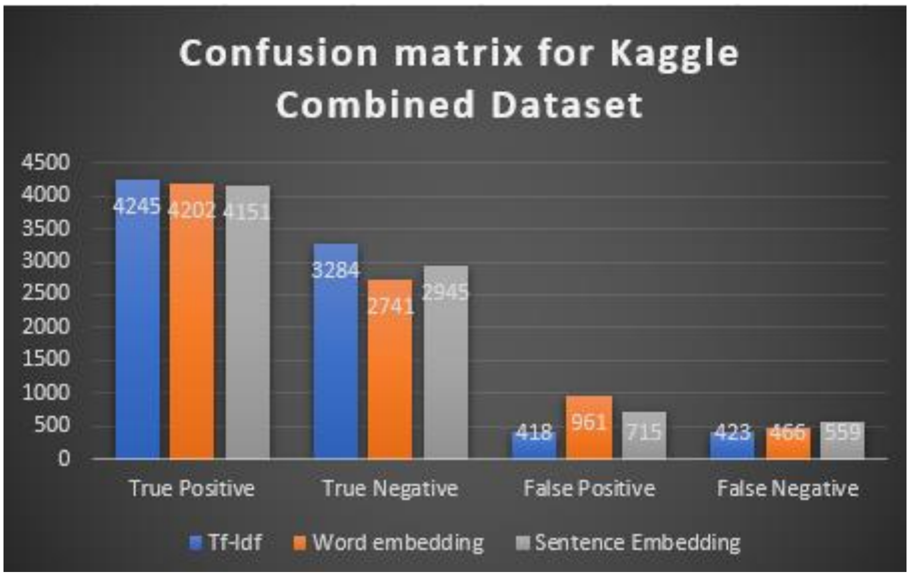


Figure 4.16

Metric	TF-IDF	Word embedding	Sentence Embedding
True Positive	4245	4202	4151
True Negative	3284	2741	2945
False Positive	418	961	715
False Negative	423	466	559

Table 4.4

4.3.3.3 Confusion matrix scores for the Politifact Buzzfeed Dataset:

From the confusion matrix in Figure 4.17 and Table 4.6, Word embedder based model has the highest True Positive rate and TF-IDF based model has the lowest True Positive rate. For True Negative, however TF-IDF’s rate is the highest. But for this dataset, TF-IDF based model has a very high False Positive rate. Word embedder based model again classifies articles as positive mostly, with having a very high False Positive rate and a very high False Positive rate.

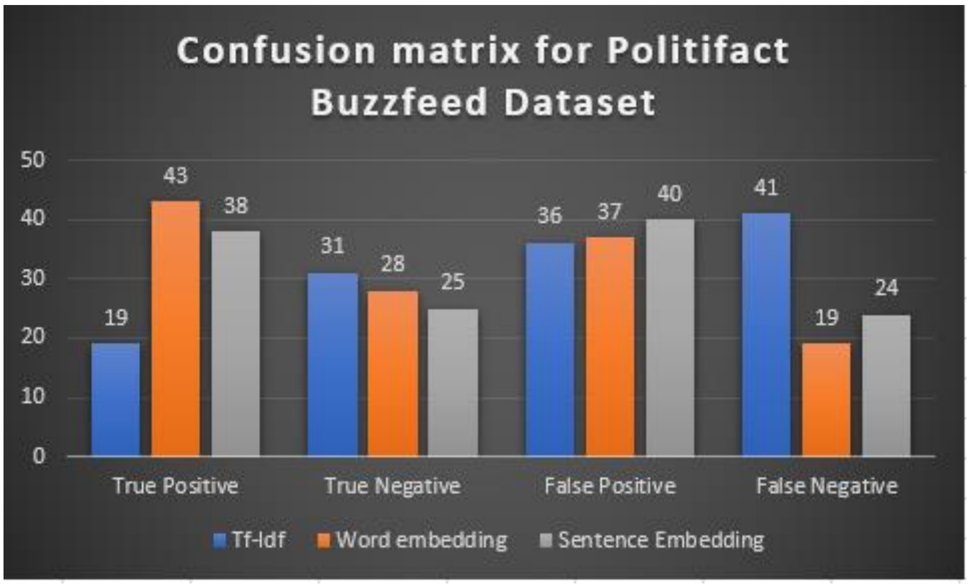


Figure 4.17

Metric	TF-IDF	Word embedding	Sentence Embedding
True Positive	19	43	38
True Negative	31	28	25
False Positive	36	37	40
False Negative	41	19	24

Table 4.5

Summary:

The scores achieved by every model for all three datasets were presented and discussed in this chapter. In chapter 5, the statistical tests that were carried out to see if the differences found were statistically significant and the hypothesis acceptance or rejection will be presented and discussed.

CHAPTER 5
EVALUATION, ANALYSIS AND DISCUSSION

In chapter 4 the experiments were carried out and the results were reported. In order to accept or reject the Null and Alternate Hypotheses, a statistical difference test between the results of different models must be carried out.

5.1 Evaluation

5.1.1 Normality Tests

In order to carry out difference tests, normal distribution or normality of data must be found. As there are two types of difference tests: Parametric tests that assume data is normally distributed and Non-Parametric tests which assume data is not normally distributed, normality distribution of data should be calculated with the help of Shapiro-Wilk Test. The Shapiro-Wilk test is generally used to evaluate whether data samples have a normal (Gaussian) distribution or not. A normal distribution can be defined as that the data's samples are symmetric around the mean, meaning that data appears more frequently near the mean value than data far away from the mean value. In a graphical form normal distribution will look like a symmetric bell curve. The Shapiro-Wilk test was carried out on all three datasets and for each of the three models. The results are presented in Table 5.1.

In the Shapiro-Wilk test, the Null Hypothesis is that the distribution is normal and the Alternate Hypothesis is the distribution is not normal. A p-value less than 0.05 indicates that the distribution is not normal. The tests indicate that the accuracy scores of the Kaggle Competition dataset and Politifact Buzzfeed dataset have a Gaussian or Normal Distribution, with p-value > 0.05, hence we do not have sufficient statistical evidence to reject the Null Hypothesis and thus indicating that the distribution is normal. For the Kaggle Combined dataset, all three models have a p-value < 0.05, so there is enough statistical evidence to reject the Null Hypothesis and thus indicating that the distribution is not normal.

Dataset	**Feature Extraction**	**Shapiro-Wilk Test**		
		W-Statistic	**p-value**	**Normal**
Kaggle Competition	TF-IDF	0.918	0.341	Yes
	Word Embedding	0.855	0.066	Yes
	Sentence Embedding	0.943	0.585	Yes
Kaggle Combined	TF-IDF	0.650	0.000	No
	Word Embedding	0.815	0.022	No
	Sentence Embedding	0.820	0.025	No
Politifact Buzzfeed	TF-IDF	0.858	0.073	Yes
	Word Embedding	0.887	0.155	Yes
	Sentence Embedding	0.847	0.053	Yes

Table 5.1

5.1.2 Statistical Difference Tests

After obtaining the normality test results, parametric Student’s t-test will be performed on normally distributed data and non-parametric Mann-Whitney U-test will be performed on non-Gaussian data. The results are tabulated in Table 5.2.

The p-value is less than 0.05 for all datasets and for all the models, compared against the baseline model of TF-IDF based Linear SVC model. There is statistically sufficient evidence to reject the Null Hypothesis and accept the Alternate Hypothesis for all the models. Thus, there is a statistically significant difference in the mean accuracy value between the baseline TF-IDF based Linear SVC model and the embeddings based Linear SVC models. A positive statistic value indicates that the baseline TF-IDF based Linear SVC model has a higher significant mean accuracy than the embeddings based Linear SVC models. Whereas, a negative statistic value indicates that the embeddings based Linear SVC models achieved a higher significant mean accuracy than the baseline TF-IDF based Linear SVC model.

Dataset	**Feature Extraction Comparison**		**Difference Test Results**		
			Difference Test	**p-value**	**Statistic**
Kaggle Competition	TF-IDF	Word Embedding	Student’s t-test	0.000	20.687
		Sentence Embedding	Student’s t-test	0.000	17.426
Kaggle Combined	TF-IDF	Word Embedding	Mann-Whitney U test	0.000	0.000
		Sentence Embedding	Mann-Whitney U test	0.000	0.000
Politifact Buzzfeed	TF-IDF	Word Embedding	Student’s t-test	0.042	-2.191
		Sentence Embedding	Student’s t-test	0.048	-2.121

Table 5.2

5.2 Acceptance/Rejection of Null and Alternate Hypothesis

Based on the 10-fold mean cross validation results presented in Table 4.2, it was clear that the baseline TF-IDF based Linear SVC model had a higher accuracy compared to the embeddings based Linear SVC model for the Kaggle competition and Kaggle Combined datasets. For the Politifact Buzzfeed dataset, interestingly the word embeddings based Linear SVC model performed best. To prove that this difference was indeed statistically significant, difference tests were performed.

Based on the results obtained in Table 5.2, acceptance or rejection of hypothesis stands as:

Kaggle Competition and Kaggle Combined datasets:
Experimental evaluation revealed that there was sufficient statistical evidence to reject the Null Hypothesis and accept the Alternate Hypothesis.

A LSVM classifier model built on ‘word2vec’ word embedding and ‘Universal Sentence Encoder’ sentence embedding text feature representation technique does not achieve a statistically significant higher accuracy for predicting fake news, than a LSVM classifier model built on traditional TF-IDF based text feature representation technique.

Politifact Buzzfeed dataset:
Although there was sufficient statistical evidence to reject the Null Hypothesis and accept the Alternate Hypothesis, the statistic value of the difference test for this dataset indicated that the word embeddings & sentence embedding based text feature extracted Linear Support Vector Machine Classifier model achieved a higher statistically significant accuracy, than the Term Frequency-Inverted Document Frequency (TF-IDF) based text feature extracted Linear Support Vector Machine Classifier model.

5.3 Summary of Results and Discussion

The TF-IDF based LSVM Classifier model trained on the Kaggle Competition dataset achieved the highest accuracy of 92% for fake news classification, amongst all the other models used in this research. The bag of words based TF-IDF text representation technique seems to be performing really well compared to the embeddings based models. A possible explanation could be that the word and sentence embedders were built for using with Deep Learning Neural Network models. Recently text classification problems are addressed more using neural networks, because they do not use bag of words approach and use the local context text window representations using word embeddings, and these capture semantics of the word at a greater scale (Major, Surkis & Aphinyanaphongs, 2018). This could be an area to explore for future work, to compare the accuracies of embeddings based Machine Learning model’s accuracy against an embeddings based Deep Learning model’s accuracy.

Although the aim of this research was to utilize the power of contextual meaning capturing word and sentence embedders to better predict and classify text articles as fake and reliable, through the results and findings of performing the research, it can summarized that the TF-IDF based LSVM model predicts news articles as reliable and fake more accurately than the word embedding and sentence embedding based LSVM model.

5.4 An additional experiment of Sentiment Analysis for Future Work

As an inspiration for future work in the field of fake news detection, an interesting area to explore was to see if negative sentiment were more prevalent to among fake news, compared to the reliable articles that are usually written with neutral sentiment.

There are many popular sentiment analysis lexicons available today, to automatically classify a text as having either positive, negative or neutral sentiment based on polarity scores. To explore this are of research, the Kaggle competition dataset was used. The dataset was split into two entities as Fake and Reliable based on their label values. Label values with 1 were split as Fake and label values with 0 were split as Reliable.

The main goal was to see if Fake news had more number of negative sentiment texts. Popular sentiment lexicons such as Afinn (Nielsen, F. Å., 2011), Text Blob and Vader were used to classify the text into positive, negative and neutral automatically.

Afinn has a wrapper library available in python known as ‘afinn’ which can be imported to use to code in python. The method Afinn.score() generates the polarity scores. This lexicon has more than 3300 words having a polarity score associated with each word.

The package TextBlob is imported from the library texblob to use this lexicon in python. The method TextBlob(text).sentiment.polarity generates the polarity score as a float in the range [-1.0, 1.0].

The VADER lexicon (Valence Aware Dictionary and sEntiment Reasoner) is a rule-based sentiment analysis tool which is mainly built to work on sentiments expressed in social media. It is available in python in the library vaderSentiment and from the package SentimentIntensityAnalyzer. The polarity scores can be obtained using the method SentimentIntensityAnalyzer.polarity_scores(text).

The results from the three lexicons on Fake and Reliable articles can be found in tables 5.3 and 5.4:

Sentiment scores on Fake articles:

Lexicon	**Negative**	**Neutral**	**Positive**
Afinn	4886	753	4353
Vader	4389	436	5167
Text Blob	1605	696	7691

Table 5.3

Afinn is the only lexicon that classifies more articles as negative in the fake articles data. TextBlob classifies a very high number of articles as positive and Vader also classifies the articles more with positive sentiments.

Sentiment scores on Reliable Articles:

Lexicon	**Negative**	**Neutral**	**Positive**
Afinn	4563	210	5614
Text Blob	1278	55	9054
Vader	3939	18	6430

Table 5.4

For the reliable articles, all three lexicons classify more articles as to having a positive sentiment, with Text Blob recording the highest number of positive sentiment classification.

Summary:

Although the primary goal of this exploration using Sentiment analysis was to see if Fake articles had more negative sentiments, the interesting yet meaningful insight that was obtained was that the Reliable articles had a higher number of positive sentiments. Although using just the lexicons is by no means a way to prove the findings are statistically significant, performing this sentiment analysis with a clustering machine learning model such as K-Means clustering algorithm with K=3 (positive, neutral, negative) will be an area to explore for future work.

Chapter 6
CONCLUSION

6.1 Research Overview

Application and processing the concepts of Natural Language Processing and Text Analytics are in greater demand today than ten years ago. As the access to digital devices and spread of social media increases day by day, developments in this research area are very much needed. Employing the latest embeddings based text representation techniques such as ‘word2vec’ word embedder and ‘Universal Sentence Encoder’ sentence embedder along with the traditional bag of words technique of TF-IDF, to classify news articles as fake or reliable, has been the main focus of this research.

6.2 Problem Definition

This research was carried out to perform experiments on the research gaps that were found during literature review, which was: only traditional bag of words based TF-IDF text representation technique were used to classify news articles as fake or reliable. Techniques that actually capture the meaning of a word based on the words surrounding it such as the word embedding based ‘word2vec’ technique and sentence embedding based ‘Universal Sentence Encoder’ technique were never used in fake news text detection. Experiments were conducted to determine if a Linear Support Vector Machine Classifier model using an embeddings based text representation technique would achieve higher significant accuracies compared against a TF-IDF based LSVM Classifier model.

6.3 Experiments, Evaluation and Results

This thesis followed the CRISP-DM design to carry out the experiments. Three existing fake news datasets were collected from Kaggle website. A thorough analysis and data understanding was performed. Based on the findings from data understanding section, the datasets were cleansed and prepared accordingly. A TF-IDF text representation technique based LSVM model was identified as the baseline model from an existing research work (Ahmed, Traore & Saad, 2017) and was compared against word embeddings and sentence embeddings based LSVM model identified from the research gap found during literature review. These models were applied on all three datasets and the results were recorded in the form of classification reports and confusion matrices. Results were validated using 10-fold Cross validation method. Shapiro-Wilk's Normality test was conducted on these 10-fold cross validation scores to see if the distribution of these scores were Gaussian or not. This Normality test was performed to identify which statistical difference test should be used. If the distribution was normal, then Student's t-test was performed and Mann-Whitney U test was performed for non-normal distribution. Difference tests were performed to record if the differences found in the accuracy between two models were statistically significant or not, in order to accept or reject the null hypothesis. An additional experiment was also identified to perform a sentiment analysis on the articles to see if negative sentiment gave rise to more number of fake articles. This was implemented using Afinn, TextBlob and Vader sentiment lexicons, available in Python.

On summarizing the research findings, it was concluded that the traditional TF-IDF text representation based LSVM model performed better than the word & sentence embeddings based LSVM models. The Null Hypothesis was rejected and Alternate Hypothesis was accepted.

6.4 Contribution and Impact

This research intended to find if the utilization of word embeddings and sentence embeddings based LSVM model will achieve an higher accuracy compared to TF-IDF based model. Although the embeddings based LSVM model did not outperform the TF-IDF based model, both the word embedding and sentence embedding based models had given a good reliable accuracy of 85% and 87% for the Kaggle Competition dataset respectively, and an accuracy of 82% and 85% for Kaggle Combined dataset respectively. For the very small Politifact Buzzfeed dataset, both the embeddings based model had performed better than the TF-IDF based model.

This research used three different datasets for performing the same models, to see if there were any drastic changes in accuracy between the datasets. Only the results of the Politifact Buzzfeed dataset were remarkably low, because of the small size of this dataset.

The Kaggle Combined dataset was actually picked from two different Kaggle kernels which had a separate Fake articles dataset and a separate Reliable dataset, which were pre-processed accordingly and combined into a single dataset.

Sentiment Analysis was performed on the fake articles and reliable articles using Afinn, Vader and Text Blob lexicons, to identify if negative sentiment gave rise to more fake articles. It was found that reliable articles had a high positive sentiment.

6.5 Future Work and Recommendation

- A possible recommendation for Future Work is that this thesis had used only a Machine Learning algorithm to classify the news articles. TF-IDF performs really well with Machine Learning algorithms, whereas the word embedding and sentence embedding techniques performed comparatively lesser. The word and sentence embedding techniques can be used with Deep Learning algorithms such as CNN, RNN in the future to see if they outperform the accuracy achieved by TF-IDF with LSVM. There is a possibility that a deep learning algorithm could make much better sense of the embeddings based text representation techniques.
- Performing a detailed and reliable sentiment analysis on fake and reliable news articles using clustering algorithms is another recommended area for future work. This research had only used already available sentiment lexicons to classify articles into negative, neutral or positive.
- Also, this research had used only the ‘text’ column or the body of the news article for classifying it as fake or reliable. Future research could possibly make a comparison between the title of the news and the body of the news and see if the title is written in a controversial way to invite people to click into the websites or links, commonly known as ‘clickbait’ articles, only to see that the body of the article is a completely different story.

- As the precision, recall, F1 scores of all the models were almost the same, they were not considered as the main evaluation metric. Future work could possibly make more use of these metrics to make a more thorough comparison.
- This research used Linear Support Vector Machine Classifier as the machine learning model as this was identified as the best performing model from literature review, as this model had achieved the highest accuracy in comparison to all the other models used by other researchers. Another recommendation for future work could be that, various model comparisons can be made and the best performing model based on the metrics identified can be selected for fake news detection.

BIBLIOGRAPHY

Agnihotri, D., Verma, K., & Tripathi, P. (2017). Variable global feature selection scheme for automatic classification of text documents. *Expert Systems with Applications*, *81*, 268-281. doi: https://doi.org/10.1016/j.eswa.2017.03.057

Ahmed, H., Traore, I., & Saad, S. (2017, October). Detection of online fake news using N-gram analysis and machine learning techniques. In *International Conference on Intelligent, Secure, and Dependable Systems in Distributed and Cloud Environments* (pp. 127-138). Springer, Cham. doi: https://doi.org/10.1007/978-3-319-69155-8_9

Arora, S., Liang, Y., & Ma, T. (2016). A simple but tough-to-beat baseline for sentence embeddings. In *ICLR.*
Retrieved from: https://openreview.net/forum?id=SyK00v5xx

Bourgonje, P., Schneider, J. M., & Rehm, G. (2017, September). From clickbait to fake news detection: an approach based on detecting the stance of headlines to articles. In *Proceedings of the 2017 EMNLP Workshop: Natural Language Processing meets Journalism* (pp. 84-89).
doi: http://dx.doi.org/10.18653/v1/W17-4215

Cer, D., Yang, Y., Kong, S. Y., Hua, N., Limtiaco, N., John, R. S., & Sung, Y. H. (2018). Universal sentence encoder. *arXiv preprint arXiv:1803.11175.*
doi: https://arxiv.org/abs/1803.11175

Chakraborty, A., Paranjape, B., Kakarla, S., & Ganguly, N. (2016, August). Stop clickbait: Detecting and preventing clickbaits in online news media. In *2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)* (pp. 9-16). IEEE.
doi: https://doi.org/10.1109/ASONAM.2016.7752207

Chen, Q., Peng, Y., & Lu, Z. (2019, June). BioSentVec: creating sentence embeddings for biomedical texts. In *2019 IEEE International Conference on Healthcare Informatics (ICHI)* (pp. 1-5). IEEE. doi: https://doi.org/10.1109/ICHI.2019.8904728

Conroy, N. J., Rubin, V. L., & Chen, Y. (2015). Automatic deception detection: Methods for finding fake news. *Proceedings of the Association for Information Science and Technology*, *52*(1), 1-4. doi: https://doi.org/10.1002/pra2.2015.145052010082

Dumais, S., Platt, J., Heckerman, D., & Sahami, M. (1998). Inductive learning algorithms and representations for text categorization. In *7th International Conference on Information and Knowledge Management*, 148-152. Retrieved from: https://www.microsoft.com/en-us/research/publication/inductive-learning-algorithms-and-representations-for-text-categorization

Ferreira, W., & Vlachos, A. (2016, June). Emergent: a novel dataset for stance classification. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: Human language technologies* (pp. 1163-1168). doi: http://dx.doi.org/10.18653/v1/N16-1138

Gilda, S. (2017, December). Evaluating machine learning algorithms for fake news detection. In *2017 IEEE 15th Student Conference on Research and Development (SCOReD)* (pp. 110-115). IEEE. doi: https://doi.org/10.1109/SCORED.2017.8305411

Gokulakrishnan, B., Priyanthan, P., Ragavan, T., Prasath, N., & Perera, A. (2012, December). Opinion mining and sentiment analysis on a twitter data stream. In *International Conference on Advances in ICT for Emerging Regions (ICTer2012)* (pp. 182-188). doi: https://doi.org/10.1109/ICTer.2012.6423033

Granik, M., & Mesyura, V. (2017, May). Fake news detection using naive Bayes classifier. In *2017 IEEE First Ukraine Conference on Electrical and Computer Engineering (UKRCON)* (pp. 900-903). IEEE. doi: https://doi.org/10.1109/UKRCON.2017.8100379

Hai, Z., Zhao, P., Cheng, P., Yang, P., Li, X. L., & Li, G. (2016, November). Deceptive review spam detection via exploiting task relatedness and unlabelled data. In *Proceedings of the 2016 conference on empirical methods in natural language processing* (pp. 1817-1826).doi: http://dx.doi.org/10.18653/v1/D16-1187

Hakim, A. A., Erwin, A., Eng, K. I., Galinium, M., & Muliady, W. (2014, October). Automated document classification for news article in Bahasa Indonesia based on term frequency inverse document frequency (TF-IDF) approach. In *2014 6th International Conference on Information Technology and Electrical Engineering (ICITEE)* (pp. 1-4). doi: 10.1109/ICITEED.2014.7007894

Hughes, M., Li, I., Kotoulas, S., & Suzumura, T. (2017). Medical text classification using convolutional neural networks. *Stud Health Technol Inform*, 235, 246-250. Retrieved from: https://www.semanticscholar.org/paper/Medical-Text-Classification-using-Convolutional-Hughes-Li/d5c67808fe08f15c3e54acbfd206c41f1e8e8e03

Jiang, M., Cui, P., & Faloutsos, C. (2016). Suspicious behavior detection: Current trends and future directions. *IEEE Intelligent Systems*, *31*(1), 31-39. doi: https://doi.org/10.1109/MIS.2016.5

Kumar, M. A., & Gopal, M. (2010, February). An investigation on linear SVM and its variants for text categorization. In *2010 Second International Conference on Machine Learning and Computing* (pp. 27-31). IEEE. doi: https://doi.org/10.1109/ICMLC.2010.64

Larson, M., Eickeler, S., Paaß, G., Leopold, E., & Kindermann, J. (2002). Exploring sub-word features and linear support vector machines for german spoken

document classification. In *Seventh International Conference on Spoken Language Processing*. Retrieved from:
https://www.isca-speech.org/archive/archive_papers/icslp_2002/i02_1989.pdf

Lin, Z., Feng, M., Santos, C. N. D., Yu, M., Xiang, B., Zhou, B., & Bengio, Y. (2017). A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*. doi: https://arxiv.org/abs/1703.03130

Ma, J., Gao, W., Mitra, P., Kwon, S., Jansen, B. J., Wong, K. F., & Cha, M. (2016, July). Detecting rumors from microblogs with recurrent neural networks.
In *Ijcai* (pp. 3818-3824). Retrieved from: https://885d47c6-a-62cb3a1a-s-sites.googlegroups.com/site/iswgao/home/ijcai16.pdf?attachauth=ANoY7cr0jdIQODFyy4klfpFdqDbqWvZEeRWSLTdpGagMH3SympWlnmIOfDASfhreGwd5s4X-_13maCEEnyauRjqv55LBX2g_BROivo1IGUwBZnyhmRnph0CK4BmaHQ6A3hG2IM7L8jFAsBfRMdsUCyRxo0E6RAs2AwU_7Z7Y2pxKY-kvElHnmsnOTciL_Qj_PZUpgaHhY_KuMQmyhT8ISzAXgFNajW34IA%3D%3D&attredirects=0

Major, V., Surkis, A., & Aphinyanaphongs, Y. (2018). Utility of general and specific word embeddings for classifying translational stages of research. In *AMIA Annual Symposium Proceedings* (Vol. 2018, p. 1405). American Medical Informatics Association.
Retrieved from: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6371342/

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.
doi: https://arxiv.org/abs/1301.3781

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. *In Advances in neural information processing systems* (pp. 3111-3119). Retrieved from:
http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf

Mihalcea, R., & Strapparava, C. (2009, August). The lie detector: Explorations in the automatic recognition of deceptive language. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers* (pp. 309-312). Association for Computational Linguistics. doi: https://dl.acm.org/doi/10.5555/1667583.1667679

Nadeau, D., & Sekine, S. (2007). A survey of named entity recognition and classification. *Lingvisticae Investigationes*, *30*(1), 3-26. doi: https://doi.org/10.1075/li.30.1.03nad

Nielsen, F. Å. (2011). A new ANEW: Evaluation of a word list for sentiment analysis in microblogs. *arXiv preprint arXiv:1103.2903.* doi: https://arxiv.org/abs/1103.2903v1

Ott, M., Cardie, C., & Hancock, J. T. (2013, June). Negative deceptive opinion spam. In *Proceedings of the 2013 conference of the north american chapter of the association for computational linguistics: human language technologies* (pp. 497-501). Retrieved from: https://www.aclweb.org/anthology/N13-1053

Ramos, J. (2003, December). Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning* (Vol. 242, pp. 133-142). Retrieved from: https://www.cs.rutgers.edu/~mlittman/courses/ml03/iCML03/papers/ramos.pdf

Rashkin, H., Choi, E., Jang, J. Y., Volkova, S., & Choi, Y. (2017, September). Truth of varying shades: Analyzing language in fake news and political fact-checking. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing* (pp. 2931-2937). doi: http://dx.doi.org/10.18653/v1/D17-1317

Refaeilzadeh, P., Tang, L., & Liu, H. (2009). Cross-validation. *Encyclopedia of database systems*, 532-538.

Rexha, A., Kröll, M., Kern, R., & Dragoni, M. (2017). An embedding approach for microblog polarity classification. In *Proceedings of the 3rd International Workshop on Emotions, Modality, Sentiment Analysis and the Semantic Web co-located with 14th ESWC*. Retrieved from: http://ceur-ws.org/Vol-1874/paper_7.pdf

Rubin, V. L., Chen, Y., & Conroy, N. J. (2015, November). Deception detection for news: three types of fakes. In *Proceedings of the 78th ASIS&T Annual Meeting: Information Science with Impact: Research in and for the Community* (p. 83). American Society for Information Science.
doi: https://dl.acm.org/doi/10.5555/2857070.2857153

Ruchansky, N., Seo, S., & Liu, Y. (2017, November). Csi: A hybrid deep model for fake news detection. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management* (pp. 797-806). ACM. Retrieved from: https://dl.acm.org/citation.cfm?id=3132877

Rudkowsky, E., Haselmayer, M., Wastian, M., Jenny, M., Emrich, Š., & Sedlmair, M. (2018). More than bags of words: Sentiment analysis with word embeddings. *Communication Methods and Measures, 12*(2-3), 140-157.
doi: https://doi.org/10.1080/19312458.2018.1455817

Shu, K., Mahudeswaran, D., Wang, S., Lee, D., & Liu, H. (2018). Fakenewsnet: A data repository with news content, social context and dynamic information for studying fake news on social media. *arXiv preprint arXiv:1809.01286*. Retrieved from:https://www.researchgate.net/profile/Suhang_Wang/publication/327464821_FakeNewsNet_A_Data_Repository_with_News_Content_Social_Context_and_Dynamic_Information_for_Studying_Fake_News_on_Social_Media/links/5b97152e299bf14739440f89/FakeNewsNet-A-Data-Repository-with-News-Content-Social-Context-and-Dynamic-Information-for-Studying-Fake-News-on-Social-Media.pdf

Shu, K., Sliva, A., Wang, S., Tang, J., & Liu, H. (2017). Fake news detection on social media: A data mining perspective. *ACM SIGKDD Explorations Newsletter, 19*(1), 22-36. doi: https://doi.org/10.1145/3137597.3137600

Shu, K., Wang, S., & Liu, H. (2017). Exploiting tri-relationship for fake news detection. *arXiv preprint arXiv:1712.07709*. Retrieved from: https://www.semanticscholar.org/paper/Exploiting-Tri-Relationship-for-Fake-News-Detection-Shu-Wang/8fd1d13e18c5ef8b57296adab6543cb810c36d81

Tang, D., Wei, F., Yang, N., Zhou, M., Liu, T., & Qin, B. (2014, June). Learning sentiment-specific word embedding for twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (pp. 1555-1565). doi: http://dx.doi.org/10.3115/v1/P14-1146

Wieting, J., Bansal, M., Gimpel, K., & Livescu, K. (2015). Towards universal paraphrastic sentence embeddings. *arXiv preprint arXiv:1511.08198.* doi: https://arxiv.org/abs/1511.08198

Yun-tao, Z., Ling, G., & Yong-cheng, W. (2005). An improved TF-IDF approach for text classification. *Journal of Zhejiang University-Science A*, 6(1), 49-55. doi: https://link.springer.com/article/10.1007/BF02842477