Detecting Opinion Spam and Fake News Using N-gram Analysis and Semantic Similarity

By
Hadeer Ahmed

B.Sc., University of Ahram Canadian, 2012

A Thesis Submitted in Partial Fulfillment of the

Requirements for the Degree of

Master of Applied Science

in the Department of Electrical and Computer Engineering

Supervisory Committee

Dr. Issa Traoré, Supervisor

(Department of Electrical and Computer Engineering, University of Victoria)

Dr. Lin Cai, Departmental Member

(Department of Electrical and Computer Engineering, University of Victoria)

ABSTRACT

In recent years, deceptive contents such as fake news and fake reviews, also known as opinion spams, have increasingly become a dangerous prospect, for online users. Fake reviews affect consumers and stores alike. Furthermore, the problem of fake news has gained attention in 2016, especially in the aftermath of the last US presidential election. Fake reviews and fake news are a closely related phenomenon as both consist of writing and spreading false information or beliefs. The opinion spam problem was formulated for the first time a few years ago, but it has quickly become a growing research area due to the abundance of user-generated content. It is now easy for anyone to either write fake reviews or write fake news on the web.

The biggest challenge is the lack of an efficient way to tell the difference between a real review or a fake one; even humans are often unable to tell the difference. In this thesis, we have developed an n-gram model to detect automatically fake contents with a focus on fake reviews and fake news. We studied and compared two different features extraction techniques and six machine learning classification techniques. Furthermore, we investigated the impact of keystroke features on the accuracy of the n-gram model. We also applied semantic similarity metrics to detect near-duplicated content. Experimental evaluation of the proposed using existing public datasets and a newly introduced fake news dataset introduced indicate improved performances compared to state of the art.

CONTENTS

Supervisory Committee ..... **i**

Abstract ..... **ii**

Contents ..... **iii**

List of tables ..... **vi**

List of Figures ..... **vii**

Acknowledgements ..... **ix**

Dedication ..... **x**

Chapter 1: Introduction **1**

1.1 Context 1

1.2 Problem statement 2

1.3 Approach Outline 3

1.4 Thesis Contributions 5

1.5 Thesis Outline: 6

Chapter 2: Related Work **7**

2.1 Opinion spam detection 7

2.2 Review Content-based Models 8

2.2.1 Lexical Models 8

2.2.2 Syntactic Models 9

2.2.3 Content and style similarity Models 11

2.2.4 Semantic Models 11

2.3 Reviewers Behavior-based Models 13

2.5 Fake News Detection 18
2.6 Summary and Discussion 19
Chapter 3: Models and Approach 21
3.1 N-gram Based Model 21
3.1.1 What is N-grams 21
3.1.2 Data Pre-processing: 22
Stop Word Removal 23
Stemming 23
3.1.3 Features Extraction 23
Term Frequency (TF) 24
TF-IDF 24
3.1.4 Classification Process 26
3.2 Keystroke Features 27
3.2.1 Editing Patterns 28
3.2.2 Timespan 28
3.3 Semantic Similarity Measurement 29
3.3.1 WordNet lexical Database 29
3.3.2 Similarity Measurement 30
3.3.3 Similarity between words: 30
Path Length Measure: 31
Depth Measure: 31
3.3.4 Similarity between two Texts 32
3.3.5 The similarity between Order Vectors 33
3.3.6 Overall Semantic similarity 34

Chapter 4: Experiments and evaluation 35

4.1 Experiments Outline 35

4.1.1 Datasets 35

Dataset 1 35

Dataset 2 36

Dataset 3 37

4.1.2 Experiment Procedure Overview 38

4.2 Experiment One 39

4.3 Experiment Two 45

4.4 Experiment Three 51

4.5 Experiment Four 52

4.6 Summary 54

Chapter 5: Conclusion and Future Work 56

5.1 Summary 56

5.2 Future Work 57

LIST OF TABLES:

Table 2.1: POS analysis of sentence 10

Table 4.1: Datasets used in the research. 37

Table 4.2: SVM Prediction Accuracy Dataset 1. The second row corresponds to the features size. Accuracy values are in %. 41

Table 4.3:LSVM Prediction Accuracy Dataset 1. The second row corresponds to the features size. Accuracy values are in %. 42

Table 4.4: KNN Prediction Accuracy Dataset 1. The second row corresponds to the features size. Accuracy values are in %. 42

Table 4.5: DT Prediction Accuracy Dataset 1. The second row corresponds to the features size. Accuracy values are in %. 43

Table 4.6: SGD Prediction Accuracy Dataset 1. The second row corresponds to the features size. Accuracy values are in %. 43

Table 4.7: LR Prediction Accuracy Dataset 1. The second row corresponds to the features size. Accuracy values are in %. 44

Table 4.8: SVM Prediction Accuracy Dataset 1. The second row corresponds to the features size. Accuracy values are in %. 47

Table 4.9: LSVM Prediction Accuracy Dataset 1. The second row corresponds to the features size. Accuracy values are in %. 48

Table 4.10: KNN Prediction Accuracy Dataset 1. The second row corresponds to the features size. Accuracy values are in %. 48

Table 4.11: DT Prediction Accuracy Dataset 1. The second row corresponds to the features size. Accuracy values are in %. 49

Table 4.12: SGD Prediction Accuracy Dataset 1. The second row corresponds to the features size. Accuracy values are in %. 49

Table 4.13: LR Prediction Accuracy Dataset 1. The second row corresponds to the features size. Accuracy values are in %. 50

Table 4.14: LSVM Prediction Accuracy Dataset 2. Accuracy values are in %. 51

Table 4.15: The average semantic measurement of each of the ten groups of fake duplicated documents 53

Table 4.16: Comparison of previous works and our work results for opinion spam detection 55

LIST OF FIGURES:

Figure 1: Classification Process 26

Figure 2: The Word distribution in dataset 1. Even though the difference is slight, it can be seen that fake content writers are using more content and filler words. Furthermore, they use more verbs and adjectives than real content writers. 39

Figure 3: Top 30 Bi-gram in Fake reviews from Dataset 1 40

Figure 4: Top 30 Bi-gram in real reviews from Dataset 1 40

Figure 5: The Word distribution in Dataset 3. Even though the difference is slight it can be seen that fake content writers are using more content and filler words. However, in contrast to Dataset 1, fake writers in Dataset 3 used less verb, and adjectives in their text. 46

Figure 6 : Top 30 Bi-gram in real News from Dataset 3 46

Figure 7: Top 30 Bi-grams in Fake News from Dataset 3 47

ACKNOWLEDGEMENTS

I would like to thank my supervisor, Dr. Issa Traoré, of the Faculty of electrical engineering at the University of Victoria for his support and help. The door to Dr. Traoré office was always open whenever I ran into trouble or had a question about my research or writing. He consistently helped me and steered me in the right direction whenever he thought I needed it.

I would like to thank, my committee member, Dr. Lin Cai for her valuable advice and critical feedback. My gratitude is extended to the external examiner Dr. Alex Thomo for putting time and effort into the evaluation of my work. Thanks to Prof. Sherif Saad, who were involved in this research without his participation and advice, I would not have been able to complete this research successfully.

I would like to thank my friend, Amera Yhya for the support and help during my stay in Canada. I would like to thank Neveen Kamal and her three lovely daughters for assisting me during my stay in Victoria, and for providing a friendly environment for me.

I would like to express my gratitude to my family, who always supported me and helped me overcome the difficulties I faced in life. Without their support and love, I would have never finished this thesis

DEDICATION

“I dedicate this work to my mother and my father that always encouraged me to study”

CHAPTER 1: INTRODUCTION

1.1 Context

In the recent years, online content has played a significant role in swaying users decisions and opinions. Opinions such as online reviews are the main source of information for customers to help with gaining insight into the products they are planning to buy. According to e-commerce statistics [1], 88 % of customers rely on reviews as a personal recommendation and 72% of them will blindly trust a business/company with positive reviews. Also, the majority of customers will take immediate action after reading the business reviews. Customers write reviews to provide feedback, by sharing their experience either bad or good with others. Their experience impacts businesses for the long term either positively or negatively. Online reviews can affect businesses from small local businesses to giant e-commerce retailers. Naturally, this creates incentives and opportunities for manipulating customers' decisions by generating false/fake reviews. Such practice is called opinion spamming where spammers will write false opinions to influence others. Opinions/Reviews may be produced either to improve the reputation of a business/product or to damage the reputations of a business/product.

There have been reports about companies hiring spammers to enhance their reputations. The New York Times [2] reported a case where Lifestyle Lift, a cosmetic surgery company, ordered staff members to pretend to be satisfied customers and writes good reviews about their face-lifting procedure/products. ABC News [3] reported the discovery of 50 Google user accounts who repeatedly posted positive reviews about the same businesses. It was suspected that the business owners hired them to write positive reviews to increase their businesses reputations. In contrast, companies can hire writers to write negative reviews to harm a rival company or a brand potentially. BBC [4] reported

in 2013, about the ongoing investigation by Taiwanese officials against Samsung for hiring students to post negative reviews on the website of HTC, a cell phone company. The New York Times [5] reported that fake review is becoming a problem on the web and an issue for numerous businesses and organizations. Recently it became clear that opinion spam does not only exist in product reviews and customers' feedback. In fact, fake news and misleading articles is another example of opinion spamming. One of the biggest sources of spreading fake news or rumors is, of course, social media websites such as Google Plus, Facebook, twitters, etc. [6].

Though the problem of fake news is not a new issue, it can be argued it exists since the beginning of times, as humans tend to believe misleading information [7]. Fake news has been getting more attention in the last couple of years, especially since the US election in 2016. It is tough for humans to detect fake news. It can be argued that the only way for them to identify manually fake news is to have a vast knowledge of the covered topic. Even with the knowledge, it is considerably hard to successfully identify if the information in the article is real or fake.

Trend Micro, a cybersecurity company, analyzed hundreds of fake news services providers around the globe. They reported that it is effortless to buy one of those services. In fact, according to the report, it is much cheaper for politicians and political parties to use those services to manipulate election outcomes and people opinions about certain topics [8,9]. Detecting fake news is believed to be a complex task and much harder than detecting fake product reviews given they spread using social media and word of mouth.

1.2 Problem statement

In our work, we investigate the scope and the impact of opinion spam and try to understand the complex nature of this problem, and develop an approach to detect automatically such occurrence. We explore the two main classes of opinion spam, namely, fake reviews and fake news.

Fake reviews are problematic since they can negatively affect various customers and companies. It can affect businesses negatively when someone uses spammers to damage a business or the business commodity; this could cause damage to the reputation and financial loss of the business/company. Furthermore, it will affect the customers/consumers by either tricking them into buying a product that is entirely different from what was advertised; or they will lose the chance to buy a product they desire. The biggest challenge is the lack of an efficient way to tell the difference between a real review or a fake one; even humans are often unable to tell the difference. As mentioned earlier, opinions posted on online-shopping sites and social media have a strong influence on the consumer's purchase decisions; as such the danger of opinion spam becomes clear. Opinion spam and in particular fake reviews are harmful to the product/service providers, consumers, online stores and even the reputations of the social media that publish these fake reviews. In fact, posting fake reviews online is illegal, and those involved in publishing fake review could be held liable, charged with a hefty fine, and even forced out of business as a result.

Fake news is a serious problem, exacerbated by the open nature of the web and social media in addition to the recent advance in computer technologies, which simplify the process of creating and spreading fake news. While it is easier to understand and trace the intention and the impact of fake reviews, the intention, and the impact of creating propaganda by spreading fake news cannot be measured or understood easily. For instance, it is clear that fake review affects the product owner, customer and online stores; on the other hand, it is not easy to identify the entities affected by the fake news. This is because identifying these entities require measuring the news propagation which has shown to be complex and resource intensive [10,11].

1.3 Approach Outline

Jindal and Liu [12] categorized fake reviews into three groups. First, there are untruthful reviews, whose primary purpose is to present false information about the product either to enhance its reputation or to damage it. Second, reviews that target the

brand but do not express an experience with an individual product. The third group is non-reviews and advertisements that contain text not related to the product. Groups three and two are relatively easy to identify, while the first one is a bit problematic. These reviews are either written by a single spammer hired by the business owner, or a group of spammers who work together in a time window to manipulate the reputation of a product or store.

Fake news could be categorized into three groups. The first is fake news, which is news that is completely fake and is made up by the writers of the articles. The second group is fake satire news, which is fake news whose main purpose is to provide humor to the readers. The third group is poorly written news articles, which have a certain degree of real news but are not entirely accurate. In short, it is news that use for example quotes from political figures to report an entirely fake story. Usually, this kind of news is designed to promote a certain agenda or bias opinion [13].

Our approach to tackling the challenges involved in identifying opinion spams consists of using text analysis using n-gram features and semantic similarity metrics. Specifically, we extract N-gram features from unique reviews and articles written by various users. In addition, we use semantic similarity metrics to calculate the similarity between essays to detect near-duplicate and duplicated reviews. In addition, we explore the benefit of keystroke dynamic features when combined with N-gram features to detect fake reviews.

According to previous research, the n-gram features are effective at detecting content written by different users, while the semantic similarity approach is better in detecting content by same users or users who re-use their content. Our goal is to use and validate the effectiveness of n-gram features to assess the likelihood of the review or article being fake.

Firstly, the n-gram model will be applied to singular reviews consisting of a mix of deceptive reviews and truthful reviews. Four types of n-gram features (unigrams, bigrams, trigram, and four-gram) will be extracted from the data. The extracted features

will be analyzed using machine learning classification. We study and compare six different supervised classification techniques, namely, K-Nearest Neighbor (KNN), Support Vector Machine (SVM), Logistic Regression (LR), Linear Support Vector Machine (LSVM), Decision tree (DT) and Stochastic Gradient Descent (SGD). In addition to the n-gram features, keystrokes based features, such as writing speed per word and numbers of pauses, will be added. According to previous research, lying imposes a cognitive burden, which only increases in real-time scenarios. Similar to real life situations, when humans are lying, certain hints (e.g., avoiding eye contact) help detect such occurrence. By the same token, when they write lies there will be hints in their keystroke typing behavior, which can contribute to distinguishing between truthful and deceptive writings, such as, pauses in typing and time duration of editing (e.g., corrections) patterns.

Secondly, we will apply the semantic similarity metrics in identifying duplicated reviews and near duplicated content. According to previous research, the imagination of spammers/writers is limited. Thus, they will not be able to completely produce different reviews about the same experience each time they write a review. The spammer would reuse parts of the reviews or keep it the same and just change some words and phrases in it to trick the readers.

1.4 Thesis Contributions

The contributions of this research can be summarized as follows.

1. We proposed an n-gram based text classification model to detect fake content. We studied the effect of n-gram length and different feature selection techniques on the accuracy of the classifier. The proposed model outperforms existing work using related approaches. We achieved an accuracy of 90% on Ott et al. [14], which is slightly above the 89% achieved by Ott et al.[14] on the same dataset. Using the dataset of Horne and Adali's [15], we achieved an accuracy of 87%, which is considerably higher than the accuracy of 71 % achieved by Horne and

Adali using text features. By running our model, on our news dataset, we obtain an accuracy of 92 % accuracy.

2. We studied the prospect of using keystrokes features developed by Choi et al. [16] combined with n-gram features to improve the effectiveness of the classifier, and drew interesting conclusions that will be useful for future research in detecting effectively fake contents.
3. We applied a semantic similarity measurement framework developed by Li et al. [17] to detect copied or semi-copied content reused by a spammer, achieving encouraging performance results. In the real world, we can find multiple duplications or near-duplications of the same fake content. The limited imagination of spammer forces them to reuse their previous content.
4. We collected a new dataset for the study of fake news, which contains 12,600 fake news articles and 12,600 legit news articles. The new dataset will represent a useful corpus for researchers working in the emerging field of fake news identification.

1.5 Thesis Outline:

The remaining chapters of this dissertation are structured as follows:

Chapter 2 will give an overview of the literature underlying this research, by providing a quick introduction to Opinion spam detection, and summarize the main approaches proposed to address this problem.

Chapter 3 will describe in detail our proposed approaches to detect fake content and describe how we generated the different features used for semantic measurement.

Chapter 4 will present and discuss in detail the results of the various experiments carried to evaluate the proposed detection methods and the corresponding datasets used.

Chapter 5 will make concluding remarks, by discussing the overall results of the research in the context of the related work. In addition, it will suggest possible improvements for future works.

CHAPTER 2: RELATED WORK

In this chapter, we review works related to opinion spam detection, and the various methods used to detect fake reviews and fake news. This chapter is organized as follows. Section 2.1 will introduce opinion spam detection. Section 2.2 will discuss existing models that detect fake opinion based on review content. Section 2.3 discusses models based on spammer's behavior. Section 2.5 will discuss research conducted to detect fake news and Section 2.6 will summarize and discuss shortcomings of some of the models.

2.1 Opinion spam detection

Opinion spam detection problem was introduced first by Jindal et al. [12] in 2008. They categorized reviews into three types: fake reviews, reviews targeting an individual brand, and reviews advertising a product. They analyzed 10 million reviews from Amazon.com to showcase and detect fake reviews. As mentioned earlier (see section 1.3), they identified three categories of fake reviews, including, untruthful reviews targeting directly the products, reviews targeting brands, and non-reviews/advertisements only indirectly related to the products. Identifying the last two categories (type 2 and type 3) was made easy by simply using standard supervised machine learning classifiers such as Naive Bayes and Logistic Regression. Also, they constructed 36 features based on review content, reviewer behavior, and product description and statistics. However, determining type 1 (fake reviews) was tricky since there was no available labeled data. Thus, they labeled duplicated and near duplicated reviews as fake opinions and the remaining data as truthful opinions. They used logistic regression to build their model, which has the added benefit of producing a probability estimating the likelihood of review to be fake. Other than logistic regression, they tried Support Vector Machine (SVM), naïve Bayes and decision tree. They were able to identify type 2 and type 3 with 98.7% accuracy. As

for the type 1, the accuracy was 78% using all features and 63% using text features only. Based on their analysis of the behavior of the spammers, they claim that most of the top reviews and reviewers on products reviewing website pages such as amazon.com is, in fact, most likely spam. Furthermore, products generating high sales numbers will receive less spam. They were able to uncover interesting information about the relationship between reviews, reviewers, products, ratings, and feedback on reviews. Normally, reviewers do not write a large number of reviews; individual products do not get a significant amount of reviews and reviews do not get much feedback.

Most models designed to detect opinion spam (i.e., fake reviews) tend to follow two directions. The first is to engineer the features from review content; the second uses features engineered from the spammer behavior.

2.2 Review Content-based Models

In this section, we will cover different categories of content-based models. First, we will discuss models built using lexical features such as n-grams in Section 2.1.1; Section 2.1.2 will discuss models built using Syntactic features. Section 2.1.3 will discuss content and style similarity models. Finally, we will discuss models based on semantic similarity in Section 2.1.4.

2.2.1 Lexical Models

Bag of words is one of the most common features used in lexical models. This consists of a group of words extracted from the text, from which n-gram features can be extracted. The second most common features are term frequency (TF), which are similar to bag of words, but are associated with frequency of the words.

Ott et al. [14] used an n-gram term frequency model to detect fake opinions. They created a ‘’gold standard’’ data set by collecting deceptive opinions of hotels from Amazon Mechanical Turk, and honest opinions from Trip Adviser. They divided all the opinions (fake and truthful) into positive and negative. Using SVM classifier, they achieved 86 % accuracy. When they removed the positive and negative separation, the

accuracy of the model dropped from 86 % to 84 %, which implied that separating the data into negative and positive improves the performance. Furthermore, they established the inability of humans to identify fake reviews efficiently. They employed humans to judge the reviews. The highest score for a human judge was 65%.

Mukherjee et al. [18] argued the validity of using pseudo fake reviews such as Ott et al. [14] golden standard dataset to detect fake reviews. Ott et al. [14] were able to achieve 89.6 % accuracy with only n-gram features. They argue that pseudo-reviews, which are generated reviews, do not count as real fake reviews, as these do not represent (unsolicited or spontaneous) real-world fake reviews. Mukherjee et al. decided to test their methods on filtered reviews from Yelp since these will be more trustworthy. They were able to achieve only 67.8 % accuracy when they tested Ott et al.[17] model on Yelp data. Thus, they believed results from models trained using pseudo fake reviews are not accurate, and the methods are useless when used on real-world data. However, they acknowledge that 67.8% is still impressive. Thus n-gram is still useful in detecting fake reviews. In a later study, Mukherjee et al. [19] claimed that Yelp spammers seem to exaggerate faking by using words and phrases which appear genuine and attempt not to reuse certain words a lot.

2.2.2 Syntactic Models

Numbers of researchers turn to Linguistic Inquiry Word Count (LIWC), and Part of Speech (POS) features combined with n-gram to detect fake reviews. LIWC is a software developed to identify cognitive and emotional characteristics in written speech. It relies on a built-in dictionary to classify target words based on different categories, for example, ‘’cries” is in sadness. Some of the categories they created are negative emotion, overall affect, verb, and past tense verb and much more. LIWC2015, the latest version of the software, can categorize word and text into 90 categories such as Personal pronouns (e.g., I and me), Common verbs, Positive emotion and more.

POS tagging, also called grammatical tagging, is mapping a word based on the word position in a sentence, and it’s meaning. For example, we have this sentence:

"John likes the apple tree at the end of the street."

An example of POS tagging for the above sentence is outlined in Table 2.1.

Table 2.1: POS analysis of sentence

Words	Tag	Description
John, end, street, apple	NN	noun, singular or mass
Like	VB	Verb
The	DT	Determiner
At, of	IN	preposition

Ott et al. [14] developed a model relying on the part of speech, Bigram and Linguistic Inquiry and Word Count features. Previous research showed a connection between the frequency distribution of POS and the genre of the text. They wanted to test if this relationship applies here. Since LIWC does not include a text categorizer, they created one from the output of the LIWC software. Version LIWC2007 of the software counts and groups the number of instances of nearly 4,500 keywords into 80 psychologically important dimensions; one feature being assigned for each of the 80 dimensions. The features are divided into four categories: Linguistics processes, psychological processes, personal concern, and spoken categories. Linguistics process refers to features that represent functionality of text such as the average number of words per sentence and the rate of misspelling. Psychological processes target emotional, social process and anything related to time and space. The third category is a personal reference to anything personal such as work, religion, and money. Lastly, spoken category covers features such as fillers (e.g.umm, ah) and words that convey agreements. Furthermore, for the text categorization, they extracted n-gram features sets. They used the aforementioned features to train a naive Bayes and SVM classifiers, respectively. For evaluation, they used a five nested fold cross-validation approach. An accuracy of 89% was obtained when they combined the

bigrams and LIWC features. In general, they discovered that bigram features alone work better than LIWC and POS approaches. Li et al. [20] also used LIWC and POS while building a model based on Sparse Additive Generative Model (SAGE). They achieved 64% accuracy with LIWC features and 63% with POS.

2.2.3 Content and style similarity Models

Stylometric features can be divided into lexical and syntactic features. Examples of lexical features include the total number of characters, the average sentence length, and the ratio of characters in a word. Examples of syntactic features include frequency of punctuations and function words.

Shojaee et al. [21] developed a stylometric-based model for review classification. The researchers used the 'gold standard' dataset created by Ott et al. [14]. They extracted 234 stylometric features, divided into lexical and syntactic features, and used for classification SVM and Naïve Bayes (NB) models. First, they tested the lexical and syntactic features separately and then the combination of both, by computing the F-measure. SVM outperformed NB with the features combined or separated. However, the highest F-measure score was 84% using both lexical and syntactic features.

2.2.4 Semantic Models

Semantic features help with identifying similarity of reviews, even if the spammer changed a word in one of his reviews to mislead the readers into thinking it is not the same opinion. For instance, they could modify the word ''Fabulous' to ''Fantastic", a semantic-based model might still be able to recognize the similarity of these reviews.

Algur et al. [22] proposed a conceptual level similarity model to detect fake reviews. The proposed approach handles similar words by mapping the words to abstract concepts. For example, if there is 'pretty cheap' and 'not expensive' in the text then this will be in price features. Reviews were divided into duplicated review and near duplicated reviews. Near-duplicated reviews were split into two categories: partially related reviews and unique reviews.They used mainly product features which reviewers mentioned in

their reviews to calculate the similarities between reviews. The proposed approach involves three steps. First, they extract the features from the reviews and store them into a features database. Second, they use the features extracted to build a matrix. Finally, the matrix is used as input to calculate the similarity measure between the reviews. If a (minimum) number of features matched between two reviews, then these are categorized as spam/duplicated reviews. According to the model evaluation results, conceptual level similarity scored lower than human annotation when detecting spam reviews. The reviews they used was stored into what they called pros and cons reviews, where the reviewers wrote the review using a pros and cons style that describe their experience with the product.

The human annotation of pros and cons reviews achieved accuracies of 92.91% and 94.37 %, respectively, the conceptual model scored 57.29 % and 30.00 %, respectively. Though it was able to detect a significant amount of spam reviews, its results in comparison with human perception indicate that it is not well suited to detect spam in general.

Lau et al. [23] focused on building their classifier using unsupervised techniques to address the unavailability of labeled data and situations where prominent features are not available. They created a method to calculate the fakeness of a review by estimating the semantic overlap of content between the reviews. The model built on the idea that a spammer is a human in the end, and human lack imaginations, which means they will tend to reuse their reviews. They will attempt to make it appear more honest by changing words, for example, they will change ‘love’ to ‘like.' Their research was the first to apply inferential language modeling and text mining to detect fake reviews. They assumed that if the semantic content of a review is similar to another review, then both are a duplicate of each other, and thus can be considered as spams. For the data, they gathered reviews from Amazon. They manually labeled the data by calculating the cosine similarly between the reviews and making human judges reviewing them. If the cosine similarity between two reviews is above a certain threshold and two human judges labels the review as spam, then it will be labeled as such. The remaining reviews will be labeled as honest opinions.

They also developed a high concept association model to extract concept association knowledge. First, they process documents of review to calculate the BMI (Balance Mutual Information) between terms to discover a conceptual view of concept and to represent them into concept vectors. Second, they calculate the relevance of a term to a concept domain, so if a term appears more in a certain domain it most likely belongs to this domain; for example, ‘pleasing’ will be added to context vector for ‘fabulous.' The final phase is extracting terms associations based on subsumption. They calculate the score for concept *x* being a subclass of concept *y*. They applied Semantic language model (SLM) and SVM algorithm to compare the effectiveness of the models. According to the results, SLM outperformed the other methods by achieving Area Under Curve (AUC) score of 0.998 while SVM scored 0.557.

2.3 Reviewers Behavior-based Models

In this Section, we will discuss Models that analyze the behavior of reviewers to detect fake reviewers and reviews. Many Spammers share the same profile characteristics and patterns. Thus several researchers have studied and shown the effectiveness of detecting fake reviews by identifying spammers.Various features have been engineered from user behavior such as the maximum number of reviews, the percentage of positive or negative reviews, and review deviation (which is a measurement of how much the rating of a reviewer is different from the average score of a product or a store).

Lim et al. [24] proposed a model based on the assumption that a spammer will target certain products or a brand to make the most impact. Also the rating she will give to the product will be different from the average score provided by other reviewers. They gathered reviews from Amazon; each review contains the text of the review and the rating which was simplified to be in the range [0, 1]. They removed reviews from anonymous users and only kept reviews from active users who at least wrote three reviews. They designed models depending on the different spammer behavior, with each model producing a separate score. In the end, all the scores are combined to reflect a final spam score. According to their work, spammers will target and monitor few products, and then

they will write fake reviews when the time is appropriate to tip the value of the product rating; they rely on targeting products in a short period. The rating will be entirely different from other reviewers' ratings, and they will attempt to review products early to sway other reviewers' opinions. The results showed that the model is effective and outperforms other baseline methods based on helpful votes (i.e., votes that others users gave to the review).

Mukherjee et al. [25] built a so-called Author Spamicity Model (ASM) to detect suspicious reviewers based on their behavior footprint. The idea is reviewers can be categorized into two groups, spammers, and non-spammers. Furthermore, each of these groups will have a different behavior distribution. Unlike previous papers on behavioral analysis, this is the first paper to propose detecting fake reviews using unsupervised Bayesian inference framework. Also, it introduced a way to analyze features using posterior density analysis. A set of experiments involving human expert evaluation have been conducted to evaluate the proposed model. The results showcase that the proposed model is efficient and outperforms other supervised methods.

Fei et al. [26] focused on detecting spammers who write reviews in short time windows. They argue that burst of reviews can be either due to the sudden popularity of the product or a spam attack. They built a network of reviewers which appears in different bursts, then represent it in a graph (Markov Random Field). Furthermore, using statistical methods, they classified reviewers as spammers or not. The authors relied on behavior features, such as the rating deviation, and reviewer burstiness; all features were normalized to [0,1] for simplicity. Like others, the authors decided to ignore single reviews since this is not enough to build statistically meaningful reviewer behavior indication. They achieved 77.6% accuracy with the proposed approach. Thus, their method is effective in detecting fake reviews in short bursts.

In contrast to the Mukherjee et al. ASM model [25], Xie et al. [27] decided to focus on singleton reviews. According to their study of available data, more than 90 percent of reviewers write only one review. Also, the size of singleton reviews is huge compared to non-singleton reviews. Thus these reviews can tip the rating of a product.

They observed that spam attack arrival patterns are hectic compared to the stable arrival patterns of regular reviews. They explored the relationship between the volume of the singleton reviews and the rating of the store. Xie et al. [27] explained that if there is a sharp increase in the volume of singleton reviews and sharp fixation (low or high) of the store rating. This means spammers are attempting to manipulate a store rating/reputation. First, for each time window, they calculate the ratio of singleton reviews, average rating and the average number of reviews. Next, they design a multidimensional time series using the three scores mentioned previously by fitting the curve of time on each dimension then applying LCS (longest common substrings) algorithm. They measure the intensity of spam bursts on each point of time on each dimension. Finally, they design a multi-scale anomaly detection algorithm on multidimensional time series based on curve fitting. The evaluation process consists of employing human judges (three of them) and comparing their labeling of the 53 stores to the output of the proposed algorithm. The algorithm was able to detect fake stores with an accuracy of 75.86 %.

Wang et al. [28] designed a multi-edge graph to detect fake reviews, based on exploring the relationship between reviewer, reviews, and store. It is the first paper that questioned the relationship between reviewer, reviews, and stores. They observed that it might be suspicious for reviewers to post multiple reviews on the same product. However, it is normal for reviewers to post multiple reviews on the same store due to multiple purchases. Furthermore, it is acceptable for stores to have near-duplicated reviews since different stores provide same services. The heterogeneous graph contains three nodes: review, reviewer, and store. Each of the nodes is connected to another: a reviewer's node is connected to reviews the reviewer wrote, and reviews are linked to the store node. Also, nodes are attached to features such as the number of reviews and rating. If reviewers share similar rating then they have a supportive relationship, if the rating is different, then they share a conflicted relationship. They introduced three metrics: the trustiness of reviewers, the honesty of a review and the reliability of stores. Trustworthy reviewers will have an amount of honest reviews; a reputable store will have a positive amount of reviews written by trustworthy reviewers. The decrease in review honesty will affect the

reviewers trustworthiness and consequently will impact the store reliability. After computing the three measurements, they used them to rank reviews, reviewers, and stores. According to the evaluation results, top-ranked reviewers and reviews are more likely to be involved in spamming activities.

Mukherjee et al. [29] published the first paper on exploring group spamming and how to solve it. Before their work, previous research focused on detecting spamming done by individuals. They proposed an algorithm that finds groups of spammers who cooperate to promote or target certain products. The first step of the approach consists of processing the reviews to generate transactions. Each transaction contains a product and all the reviewers who reviewed it. Next, they perform pattern mining on the transactions to produce candidate spam groups, which are groups of users/reviewers who have all reviewed a set of products. They used pattern mining based on the logic that cooperated spammers must work together on more than one product to maximize their profit. Thus, they will write multiple reviews on multiple products. However, this does not guarantee that all the groups are spammers groups. Mukherjee et al. [29] used GSRank (Group spam rank), a logical model to exploit the relationships between the groups, the reviewers in the group and the products they reviewed. GSRank relies on spams indicators, such as whether the groups wrote reviews in the same short window of time, whether the reviewers wrote reviews right after the products were published, content similarity, and the rating they give to the products. In the evaluation phase, the researchers employed eight human judges to review the spams groups and manually label them. They used the labeled data for training different models. Also, they applied the models to four types of features group: spam features, individual spam features, linguistic features and a combination of all of the above. GSRank results outperformed all other supervised learning techniques, such as SVM and Logistic regression, by scoring 95 % (AUC) while, for example, SVM scored 86% (AUC). Also, group spam features outperformed other features.

Feng et al. [30] investigated the connection between the distribution anomaly and detecting fake reviews. They hypothesized that if a deceptive business hires spammers to

write fake reviews, the spammers will try to tip the review score. Thus they will distribute according to the natural distributions. For example, assume that over a five-month period, product *x* received three reviews per day, and then in a short period of time product *x* started getting a huge number of reviews and then it stopped and went back to normal. Now this spike in reviews would have created a trace left behind by spammers which can be detected by this model. For evaluation, they used a subset of the gold standard dataset from Ott et al. [14], which contains 400 deceptive and truthful reviews, as training and testing data. They achieved an accuracy of 72.5 % on their test data. Also, the authors were the first to provide a quantitative study that characterizes the distribution of opinioning using data from Amazon and TripAdvisor. The proposed method is effective in detecting suspicious activity within a window of time. However, it is ineffective in detecting whether individual reviews are false or truthful opinions. Maybe such limitation could be addressed by combining it with another method capable of detecting individual deceptive reviews.

Li et al. [31] decided to not focus entirely on heuristic rules like previous researchers. They used manually labeled reviews from Epinions.com. They considered reviews that received high numbers of helpful votes and comments more trustworthy. They assumed reviews with low helpful votes are more suspicious than reviews with a significant number of votes, and more likely fake. Furthermore, they ignored reviews from anonymous users and duplicated reviews. The authors used supervised learning techniques such as SVM, NB, and logistic regression to identify review spam. Two groups of features were extracted. The first group consists of review features such as content features (unigrams, bigrams), content similarity features and sentiment features. The second group consists of reviewers related features such as profile features and behavior features. The Naïve Bayes method achieved much better results compared to other methods that rely on behavior features. It scored 0.583 F-score with all features. They observed when behavior features are excluded from the features set, that the score drops significantly. Also, they noted that helpful votes perform poorly compared to other features.

2.5 Fake News Detection

Research on fake news detection is still at an early stage, as this is a relatively recent phenomenon, at least in the interest raised by society. We review some of the published work in the following.

Rubin et al. [32] discuss three types of fake news. Each is a representation of inaccurate or deceptive reporting. Furthermore, the authors weight the different kinds of fake news and the pros and cons of using different text analytics and predictive modeling methods in detecting them. In this paper, they separated the fake news types into three groups:

- Serious fabrications are news not published in mainstream or participant media, yellow press or tabloids, which as such, will be harder to collect.
- Large-Scale hoaxes are creative and unique, and often appear over multiple platforms. The authors argued that it may require methods beyond text analytics to detect this type of fake news.
- Humorous fake news, are intended by their writers to be entertaining, mocking, and even absurd. According to the authors, the nature of the style of this type of fake news could have an adverse effect on the effectiveness of text classification techniques.

The authors argued that the latest advance in natural language processing (NLP) and deception detection could be helpful in detecting deceptive news. However, the lack of available corpora for predictive modeling is an important limiting factor in designing effective models to detect fake news.

Horne et al. [15] illustrated how obvious it is to distinguish between fake and honest articles. According to their observations, fake news titles have fewer stop-words and nouns, while having more nouns and verbs. They extracted different features grouped into three categories as follows:

- Complexity features calculate the complexity and readability of the text.

- Psychology features illustrate and measure the cognitive process and personal concerns underlying the writings, such as the number of emotion words and casual words.
- Stylistic features reflect the style of the writers and the syntax of the text, such as the number of verbs and the number of nouns.

The aforementioned features were used to build a SVM classification model. The authors used a dataset consisting of real news from BuzzFeed and other news websites, and Burfoot and Baldwin's satire dataset [33] to test their model. When they compared real news against satire articles (humorous article), they achieved 91 % accuracy. However, the accuracy dropped to 71% when predicting fake news against real news.

Wang et al. [34] introduced LIAR, a new dataset that can be used for automatic fake news detection. Though LIAR is considerably bigger in size, unlike other datasets, this dataset does not contain full articles; instead, it contains 12,800 manually labeled short statements from politicalFact.com.

Rubin et al. [35] proposed a model to identify satire and humor news articles. They examined and inspected 360 Satirical news articles in mainly four domains, including civics, science, business, and what they called "soft news" ('entertainment/gossip articles'). They proposed a SVM classification model using mainly five features developed based on their analysis of the satirical news. The five features are Absurdity, Humor, Grammar, Negative Affect, and Punctuation. Their highest precision of 90 % were achieved using only three combinations of features which are Absurdity, Grammar, and Punctuation.

2.6 Summary and Discussion

Opinion mining field examines and analyzes opinions, emotions and evaluates human attitude toward certain products, causes, or services. A basic advantage of social media is it allows people to express their opinions and emotions without being subjected to direct judgment. However, it also allows people with malicious intentions to post fake opinions

to promote or damage a product, a cause or an organization. Opinion spamming is becoming a major issue for society, businesses, and organizations.

The first method to detect fake reviews is by identifying spammer behavior and extracting behavioral features to detect them. Examples of behavioral features include spammer IP address, the number of reviews posted, a rating of the product and the feedback received for reviews. Behavioral models mostly work well when they are detecting a group of spammers who work together or spammers who use the same account to spam multiple products or same product. However, they tend to ignore singular reviewers and spammers that change their behavior constantly, which affect their effectiveness.

The second method consists of using text analysis to detect if a review is fake or honest. Text analysis includes retrieving information from a review using lexical features (e.g. n-gram, part of speech (POS)), style similarity features, or semantic similarity features. The advantage of text analysis is it relies solely on the text. Thus, it will treat group and individual opinion spamming the same . For instance, Ott et al. [14] proposed an N-gram model that performed well when classifying unique singular reviews. However, It must be taken into consideration that spammers re-use their reviews. Either they re-use it completely unchanged, or they change a word and phrase in it and then re-post it. This can be addressed using a semantic similarity model, which will be able to detect duplicated or near-duplicated reviews.

We believe that one of the best ways to address the challenges mentioned above is using text analysis using word-based n-gram feature to detect fake content. Thus, we have decided to apply different variations of N-gram features on unique reviews written by various users. Also, we used a semantic similarity metric model developed by Li [17] to calculate the similarity between two texts to detect near-duplicate and duplicated content. Furthermore, we explored the effect of keystroke dynamics features combined with N-gram model on detecting fake reviews.

CHAPTER 3: MODELS AND APPROACH

In the recent years, online content and opinion posts, such as online reviews, essays, and news articles, have played a significant role in swaying users decisions and opinions. In this chapter, we will discuss our proposed methods to identify fake opinions and honest ones. In section 3.1 we will discuss the text classification methods used to detect fake content. Section 3.2 will discuss the method used to generate keystrokes features and Section 3.4 will outline the semantic measurement approach and model we applied to detect duplicated content.

3.1 N-gram Based Model

3.1.1 What is N-grams

N-gram modeling is a popular feature identification and analysis approach used in language modeling and natural language processing fields. It started with Claude Shannon in 1948 [36] when he investigated the concept of predicting the next letter in a given sequence of letters. Since then, the use of n-gram expanded into several applications such as Statistical Machine Translation, word similarity, Authorship identification, and Sentiment Extraction. In term of text classification, n-gram language models were proven successful when applied to any language or even non-language scripts such as music or DNA [37].

N-gram is a contiguous sequence of items with length *n*. It could be a sequence of words, bytes, syllables, or characters. The most used n-gram models in text categorization are word-based and character-based n-grams. Examples of n-gram models commonly used include unigram (n=1), bigram (n=2), trigram (n=3), etc. For example, the word-based n-gram corresponding to the following sentence is:

"Samsung Electronics expected to forecast a high profit."

Uni-gram: Samsung, Electronics, expected, to, forecast, a high, profit.

Bi-gram: Samsung Electronics, Electronics expected, expected to, to forecast, forecast, a, a high, high profit.

Tri-gram: Samsung Electronics expected, Electronics expected to, expected to forecast, to forecast a, forecast a high, a high profit.

Quad-Grams: Samsung Electronics expected to, Electronics expected to forecast, expected to forecast a, to forecast a high, forecast a high profit.

When building an n-gram based classifier, the size *n* is usually a fixed number throughout the whole corpus. The unigrams are commonly known as "the bag of words" model. The bag of words model does not take into consideration the order of the phrase in contrast to a higher order n-gram model. The n-gram model is one of the basic and efficient models for text categorization and language processing. It allows automatic capture of the most frequent words in the corpus; it can be applied to any language since it does not need segmentation of the text in words. Furthermore, it is flexible against spelling mistake and deformations since it recognizes particles of the phrase/words.

In this thesis, we will be using word-based n-gram model to represent the context of the document and generate features to classify the document. One of the goals of this thesis is to develop a simple n-gram based classifier to differentiate between fake and real opinions.The idea is to generate various sets of n-gram frequency profiles from the training data to represent fake and truthful opinions. We used different values of *n* to generate and extract the n-gram features. We will examine the effect of the n-gram length and the number of features selected to represent the data on the accuracy of the different classification algorithms considered.

3.1.2 Data Pre-processing:

Before representing the data using n-gram and vector-based model, the data need to be subjected to certain refinements like stop-word removal, tokenization, a lower casing,

sentence segmentation, and punctuation removal. This will help us reduce the size of actual data by removing the irrelevant information that exists in the data.

We created a generic data pre-processing function to remove punctuation and non-letter characters for each document; then we lowered the letter case in the document. In addition, an n-gram word based tokenizer was created to slice the reviews text based on the length of *n*.

Stop Word Removal

Stopwords are insignificant words in a language that will create noise when used as features in text classification. These are words commonly used in a lot sentences to help connect thought or to assist in the sentence structure. Articles, prepositions and conjunctions and some pronouns are considered stop words. We removed common words such as, *a, about, an, are, as, at, be, by, for, from, how, in, is, of, on, or, that, the, these, this, too, was, what, when, where, who, will*, etc. Those words were removed from each document, and the processed documents were stored and passed on to the next step.

Stemming

After tokenizing the data, the next step is to transform the tokens into a standard form. Stemming, simply, is changing the words into their original form, and decreasing the number of word types or classes in the data. For example, the words “Running,” ”Ran” and “Runner” will be reduced to the word “run.” We use stemming to make classification faster and efficient. Furthermore, we use Porter stemmer, which is the most commonly used stemming algorithms due to its accuracy.

3.1.3 Features Extraction

One of the challenges of text categorization is learning from high dimensional data. There is a large number of terms, words, and phrases in documents that lead to a high computational burden for the learning process. Furthermore, irrelevant and redundant features can hurt the accuracy and performance of the classifiers. Thus, it is best to perform feature reduction to reduce the text feature size and avoid large feature space dimension. We applied in this research two features extraction methods, namely, Term

Frequency (TF) and Term Frequency-Inverted Document Frequency (TF-IDF). These methods are described in the following.

Term Frequency (TF)

Term Frequency is an approach that utilizes the counts of words appearing in the documents to figure out the similarity between documents. Each document is represented by an equal length vector that contains the words counts. Next, each vector is normalized in a way that the sum of its elements will add to one. Each word count is then converted into the probability of such word existing in the documents. For example, if a word is in a certain document it will be represented as one, and if it is not in the document, it will be set to zero. Thus, each document is represented by groups of words.

In our case, the Term Frequency will represent each term in our vector with a measurement that illustrates how many times the term/features occurred in the document. We used *Count Vectorizer* class from scikit-learn, a Python module to produce a table of each word mentioned, and its occurrence for each class. *Count Vectorizer* learns the vocabulary from the documents and then extracts the words count features. Next, we create a matrix with the token counts to represent our documents.

TF-IDF

The Term Frequency-Inverted Document Frequency (TF-IDF) is a weighting metric often used in information retrieval and natural language processing. It is a statistical metric used to measure how important a term is in a document over a dataset. A term importance increases with the number of times a word appears in the document, however, this is counteracted by the frequency of the word in the corpus.

Let D denote a corpus, or set of documents.

Let d denote a document, $d \in D$; we define a document as a set of words w. Let $n_w(d)$ denote the number of times word w appears in document d. Hence, the size of document d is $|d| = \sum_{w \in d} n_w(d)$.

The normalized Term Frequency (TF) for word *w* with respect to document *d* is defined as follows:

$$TF(w)_d = \frac{n_w(d)}{|d|}$$

The Inverse Document Frequency (IDF) for a term *w* with respect to document corpus *D*, denoted $IDF(w)_D$ is the logarithm of the total number of documents in the corpus divided by the number of documents where this particular term appears, and computed as follows:

$$IDF(w)_D = 1 + log\left(\frac{|D|}{|\{d: D|w \in d\}|}\right)$$

One of the main characteristics of IDF is it weights down the term frequency while scaling up the rare ones. For example, words such as "the" and "then" often appear in the text, and if we only use TF, terms such as these will dominate the frequency count. However, using IDF scales down the impact of these terms.

TF-IDF for the word *w* with respect to document *d* and corpus *D* is calculated as follows:

$$TF - IDF(w)_{d,D} = TF(w)_d \times IDF(w)_D$$

So for example, let say we have a document with 200 words and we need the TF-IDF for the word "people". Assuming that the word "people" occurs in the document 5 times then TF = 5/200 = 0.025. Now we need to calculate the IDF; let's assume that we have 500 documents and "people" appears in 100 of them then IDF(people) = 1 + log (500/100) = 1.69. Then TF-IDF(people) = 0.025 × 1.69 = 0.04056.

We use *TfidfVectorizer* and *TfidfTransformer* from scikit-learn for implementation. *TfidfVectorizer* converts the training and testing data into a matrix of TF-IDF features.

3.1.4 Classification Process

Figure 3.1 is a diagrammatic representation of the classification process. It illustrates the steps that were involved in this research from data collection to assigning class labels to contents. It starts with collecting the different datasets from multiple sources, then removing unnecessary characters and words from the data. N-gram features are extracted, and a matrix is formed to represent each document.

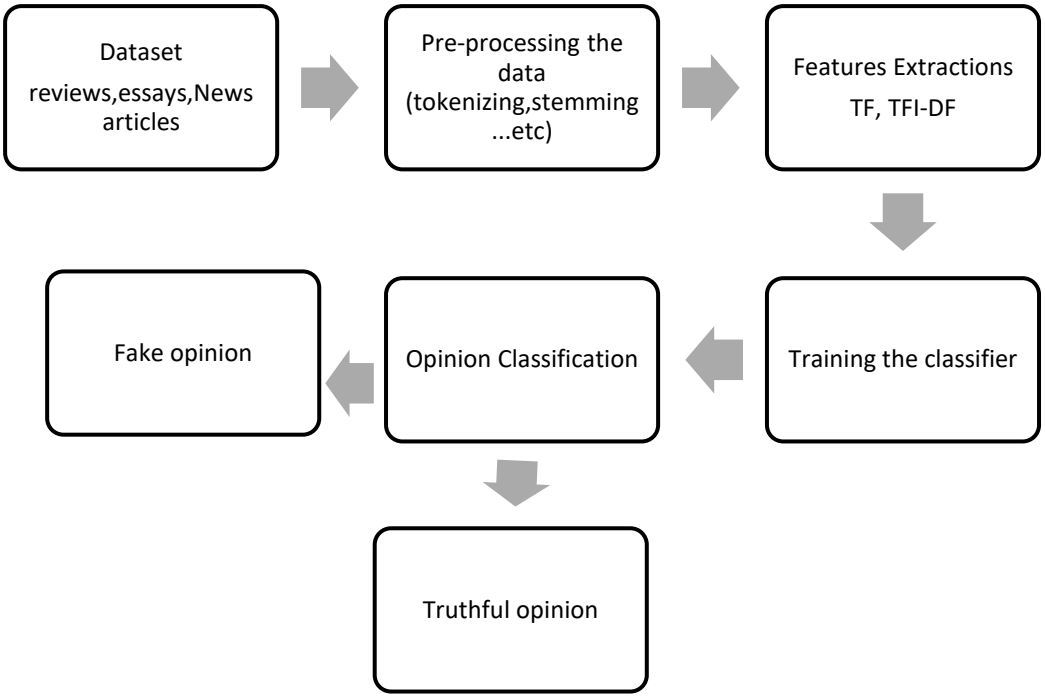

Figure 1: Classification Process

Given a document corpus or dataset, we split the dataset into training and testing sets. For instance, in the experiments presented subsequently, we use 5-fold cross-validation, so in each validation, around 80% of the dataset is used for training and 20% for testing.

Assume that $\Delta = [d_i]_{1 \le i \le m}$ is our training set consisting of *m* documents d_i.

Using a feature extraction technique (i.e., TF or TF_IDF), we calculate the feature values corresponding to all the terms/words involved in all the documents in the training corpus

and select the p terms t_j ($1 \leq j \leq p$) with the highest feature values. Next, we build the features matrix $X = [x_{ij}]_{1 \leq i \leq m, 1 \leq j \leq p}$, where:

$$\begin{cases} x_{ij} = feature(t_j)\ if\ t_j \in d_i \\ x_{ij} = 0\ otherwise \end{cases}$$

In other words, x_{ij}corresponds to the feature extracted (using TF or TF-IDF) for term $t_j$for documentd_i. Such value is null (0) if the term is not in the document.

Using the notation and definition given earlier:

- For TF: $feature(t_j) = TF(t_j)_{d_i}$
- For TF-IDF: $feature(t_j) = TF - IDF(t_j)_{d_i,\Delta}$

The last step in the classification process is to train the classifier. We investigated different classifiers to predict the class of the documents. We investigated specifically six different machine learning algorithms, namely, Stochastic Gradient Descent (SGD), Support Vector Machines (SVM), Linear Support Vector Machines (LSVM), K-Nearest Neighbour (KNN), Logistic Regression (LR) and Decision Tree (DT).

3.2 Keystroke Features

The method discussed in this section is inspired by the work of Choi [16] on detecting deceptive opinions. The features are generated based on the concept that lying imposes a cognitive burden which increases in real-time scenarios. It means users who write fake content will take a longer time to finish writing, in contrast to users who write truthful content. In addition, they will make more mistakes because of the cognitive burden. Two types of features are considered, namely, editing pattern and writing time span. These features are described in the following.

3.2.1 Editing Patterns

The editing pattern features are extracted from a subset of available keys, that record the user's actions when editing the text. These include the ‘Backspace’ and ‘Delete’ keys, the arrow keys and the number of times the mouse is used (represented by the MouseUp event).

These features are represented by a vector E3 as follows:

$$\boldsymbol{E3} = \{\boldsymbol{Del}, \boldsymbol{MSelect}, \boldsymbol{Arrow}\}$$

Where:

- DEL = Number of deletion keystrokes
- MSELECT = number of ‘MouseUp’ events
- ARROW = number of arrow keystrokes

3.2.2 Timespan

Based on keystrokes data aspects, six timespan features are considered that help improve on the result of the text classification. Those features are represented in a vector denoted by T6, consisting of the following:

- δ (D) = The timespan of entire document
- δ (kprv + W) = average timespan of word plus preceding keystroke
- δ (k) = average keystroke timespan
- δ (SP) = average timespan of spaces
- δ (¬SP) = average time span of non-whitespace Keystrokes
- δ (¬W) = average interval between words.

Different people have a differents typing speed and skill. Some will type faster than others. For this reason, all time spans features will be normalized on the corresponding event.

3.3 Semantic Similarity Measurement

3.3.1 WordNet lexical Database

Wordnet [38] is a lexical database spearheaded by George A. Miller, which is available in English. It consists of words, specifically nouns, verbs, adjectives, and adverbs. Words that are synonymous are grouped into synsets. Synsets are connected through semantic and lexical relations.

The relationships between the words are categorized as follows:

- Synonymy is a symmetric relation; it is between words that equivalents to each other.
- Antonymy (opposing-name) relationship between words with opposite meaning such as “wet” and “dry.”
- Hyponymy (sub-name) is the most important relation among synsets; it connects abstract synsets with each other such as “bed” and “furniture; “bed” is a hyponym of “furniture”
- Hypernym is the opposite of hyponymy; for instance, taking the previous example, the hypernym of “bed” is “furniture”
- Meronymy (part-name) is a part-whole relationship, for example, “leg” and “chair” or “hand” and “arm”.The parts are inherited “downward”.
- Troponymy (manner-name) is similar to hyponymy, but it is for the verbs

The hypernym/hyponym relationship is analogous to the generalization/specialization relationship between concepts in ontologies. In lexical databases such as Wordnets, words at upper layers express more general concepts and have less semantic similarity between them. However, the words at lower layers have more semantic similarity and less general concepts. The hierarchy distance of a synset is a measure of how far the synset/node is from the farthest hypernyms (which correspond to the root) [38].

3.3.2 Similarity Measurement

In this section, we present the steps to calculate the semantic similarity between documents. Four main steps are involved in this process as follows:

1. Calculate the similarity between words
2. Calculate the similarity between texts
3. Calculate the Order similarity between texts
4. Calculate the overall Semantic Similarity.

The proposed semantic measurement approach was implemented using Python and the Natural Language Toolkit (NLTK). We used the Wordnet lexical database and the Statistics Package from the Brown Corpus, which are both available in the NLTK's corpus API. The proposed measurement was developed by Li et al. [17] to measure the similarity between words and sentences.

3.3.3 Similarity between words

According to Li *et al.* [17], we can calculate the semantic similarity between two words by using a lexical knowledge base. As mentioned above, we use in our work the WordNet lexical knowledge base. The path length and their depth can determine the similarity between two words.

The semantic similarity between two words w_1 and w_2, denoted $S(w_1, w_2)$, will be calculated as follows:

$$S(w_1, w_2) = f_L(w_1, w_2) \times f_H(w_1, w_2)$$

Where :

- $f_L(w_1, w_2)$ is the path length between two synsets corresponding to w_1 and w_2, respectively.
- $f_H(w_1, w_2)$ is the depth measure between two synsets corresponding to w_1 and w_2, respectively.

Path Length Measure:

The path length function takes two words and returns the measurement of the length of the shortest path in the lexical database between them using the formula developed by Li et la. [17]. The function checks three conditions to calculate the length as follows.

- First, it determines if both synsets are the same and if so it will return 0 since that means we are calculating the path length of the same word.
- Second, it will determine if there is an intersection between the two synsets. For example, if we are trying to get the length of the word "boy" and "girl," both of them have a similar synset which is the synset "Child," the algorithm will return the distance as 1.
- Lastly, it will compute the shortest path distance between the two synsets; if it is none it will return 0, else it will return the corresponding path length.

All values are normalized to range from one to zero.

The following formula is used to calculate the path length :

$$f_L(w_1, w_2) = e^{-\alpha l_{dist}(w_1, w_2)}$$

Where:

- Alpha is a constant, which according to Li et al. [39] equal to 0.2 for the WordNet corpus: $\alpha = 0.2$
- $l_{dist}(w_1, w_2)$ is the length distance between the synsets corresponding to w_1 and w_2, obtained as explained above.

Depth Measure:

The depth function returns a measure of depth between two synsets. It works as follows:

- First, it determines if it is comparing the same synsets, if so it will return the hierarchy distance depth of one of the synsets.

- Second, if the two words are not the same, it will attempt to find the maximum hierarchy distance between them and the farthest common subsumer/ hypernyms. If there is is no distance it will return the hierarchy distance as zero.

Using the hierarchy distance, the depth is calculated using the following formula proposed by Li et al. [17]:

$$f_H(w_1, w_2) = \frac{e^{\beta h_{dist}(w_1,w_2)} - e^{-\beta h_{dist}(w_1,w_2)}}{e^{\beta h_{dist}(w_1,w_2)} + e^{-\beta h_{dist}(w_1,w_2)}}$$

Where :

- Beta is a constant, which according to Li et al. [39] equal to 0.45 for the WordNet corpus: $\beta = 0.45$
- $h_{dist}(w_1, w_2)$ is the distance between the synset and the farthest hypernym.

3.3.4 Similarity between two Texts

We compute the similarity between two texts using the semantic function created by Li et al. [17].

The process to compute the similarity between two texts texts t_1 and t_2 works as follows:

- Tokenize the texts t_1 and t_2, and combine both outputs into a joint list J.
- Calculate semantic vectors $v_{t_1}$and $v_{t_2}$for each of the texts t_1 and t_2, respectively, as follows:
    - For vector v_{t_1}, compare each word in t_1 with the words in J; if the word exists in J it will be represented in the vector $v_{t_1}$with value 1. If a word is not available in J, try to find the most similar word to it in J; then calculate the semantic similarity between the words and add it to the vector. If the

word is unique and there is no similar word to it in J it will be represented by the value 0.

- For vector v_{t_2}, repeat the same process as above with text t_2.

- Obtain the semantic similarity between the two texts by calculating the cosine similarity between the two vectors v_{t_1} and v_{t_2}:

$$Sem\_Sim(t_1, t_2) = \frac{v_{t_1} . v_{t_2}}{\|v_{t_1}\| \times \|v_{t_2}\|}$$

3.3.5 Order similarity between two texts

Given two texts t_1 and t_2, we compute the similarity between order vector based on the order of the words involved. Each word in t_1 and t_2 will be assigned an index to illustrate it's position in the text. Instead of returning a semantic representation of the text, this function will return a vector that contains a comparison of word orders for each text against the other.

The process to compute the similarity between the order vectors ov_{t_1} and ov_{t_2} for two texts texts t_1 and t_2 works as follows:

- Similar to above, tokenize the texts t_1 and t_2, and combine both outputs into a joint list J.

- Calculate the order vectors ov_{t_1} and ov_{t_2} for the two texts t_1 and t_2, as follows:

  a. For vector ov_{t_1}, compare each word in the Joint list J to the words in t_1. If a word exists in t_1, it will be represented by it's position in t_1. If a word is not available in t_1, look for the most similar word in t_1; if found it will take the position of this word. Otherwise, if the word does not exist in t_1 then it will be represented by 0.

  b. For vector ov_{t_2}, repeat the same process as above with text t_2.

- Obtain the ordered similarity between the two vectors using this formula :

$$Ordered_{Sim(t_1,t_2)} = 1.0 - \frac{\left\| ov_{t_1} - ov_{t_2} \right\|}{\left\| ov_{t_1} + ov_{t_2} \right\|}$$

3.3.6 Overall Semantic similarity

To get the overall similarity between two texts t_1 and t_2, we apply the following formula proposed by Li et al. [17]:

$$S(t_1, t_2) = \delta \times Sem\_Sim(t_1, t_2) + (1 - \delta) \times Ordered\_Sim(t_1, t_2)$$

Where:

- Delta is a constant: $\delta = 0.85$

CHAPTER 4: EXPERIMENTAL EVALUATION

We describe in this chapter the experiment evaluation of our proposed approach and discuss obtained results. As part of the evaluation, we will assess the impact of the features models on the performance. In particular, we will study different types of n-grams, by varying *n*, and by comparing fixed n-grams and variable size n-grams. We will examine the effect of the quality of the data used for training on the performance of various classifiers. Also, we will investigate the use of keystrokes features and how it will affect the performance of classifiers combined with n-gram features. Lastly, we will assess the effectiveness of the semantic measurement in detecting near-duplicate content.

This chapter starts by describing the datasets and providing an outline of the experiments. Subsequent subsections describe in detail the experimental procedures and present the results obtained for different text classifiers, training, and testing datasets.

4.1 Experiments Outline

4.1.1 Datasets

We used in our experiments three different datasets outlined in Table 4.1. We describe these datasets in the following.

Dataset 1

This dataset, collected by Ott et al. [14], contains 800 truthful reviews and 800 fake reviews, all written in English. The truthful reviews are from Tripadvisor and cover the pages of the 20 most popular hotels in Chicago. The fake reviews were written by 400 Human-Intelligence Tasks (HITs) hired on Amazon Mechanical Turk. The HITs were asked to pretend that their boss asked them to write either positive or negative fake

reviews for the hotels. 800 fake reviews were produced covering the same 20 hotels as the aforementioned truthful reviews from TripAdvisor. When collecting the reviews, Ott et al. [14] ignored reviews with less than five stars and reviews with less than 150 characters. For every review, the following information is available:

- Hotel Name
- Review text
- Review Sentiment (Positive or negative)
- Review label (Fake or truthful)
- Also, we added the review text length - the total number of words in the text.

Dataset 2

This dataset was collected by Choi et al. [16] using Amazon Mechanical Turk. Turks were requested to write fake and honest reviews regarding three topics: restaurant review, gay marriage, and gun control. In addition to the review context, Choi et al. [16] recorded their keylogging data. They requested that each user write two reviews for one of the topics, one deceptive and one honest. They collected 2600 opinions, 1000 restaurant reviews, 800 gay marriage essays, and 800 gun control essays. The gay marriage and gun control categories are divided into supporting and opposing reviews. For every review, the following information is available:

- Review text
- Review Sentiment (support or oppose)
- Review label (fake or truthful)
- Review text length
- Review keystrokes Data (KeyUp, KeyDown, and MouseUp events, along with the keycode and timestamp)
- Reviewers ID Number

Dataset 3

In contrast to the first two datasets, this dataset was entirely collected from real-world sources as part of this thesis. We collected news articles from Reuters.com (News website) for truthful opinions. As for the fake reviews, they were collected from a fake news dataset on kaggle.com. The collector of the dataset collected fake reviews from unreliable websites that Politifact (a fact-checking organization in the USA) has been working with Facebook to stamp out. We used 12,600 fake news articles from kaggle.com and, 12,600 truthful articles. We decided to focus only on political news article because these are currently the main target of spammers. The news articles from both fake and truthful categories happened in the same timeline, specifically in 2016. Each of the articles length is bigger than 200 characters.

For every review, the following information is available:

- Article text
- Article Type (e.g. political, entertainment, etc.)
- Article label (fake or truthful)
- Article Title
- Article Date

Table 4.1: Datasets used in the research.

Dataset	**Topics in the data**	**Size/Content**	**References**
Dataset 1	Hotels reviews	800 Fake hotel reviews 800 Real hotel reviews	[14]
Dataset 2	Resturant reviews Gun control essays Gay marriage essays	1000 Resturant reviews 800 gay marriage essays 800 gun control essays	[16]

Dataset 3	News Articles	12,600 Fake news 12,600 Real news	Current work

4.1.2 Experiment Procedure Overview

As mentioned earlier, we investigated in this research six different machine learning algorithms, namely, Stochastic Gradient Descent (SGD), Support Vector Machines (SVM), Linear Support Vector Machines (LSVM), Logistic Regression (LR) , K-Nearest Neighbour (KNN) and Decision Trees (DT). We used implementations of these classifiers from the Python Natural Language Toolkit (NLTK).

The experiments started by studying the impact of the size (*n*) of n-grams on the performance. We started with unigram (n = 1), then bigram (n=2), then steadily increased *n* by one until reaching (n=4). Furthermore, each *n* value was tested combined with a different number of features. We conducted four different kinds of experiments. In the first experiment, we tested the n-gram features against the first dataset (Reviews dataset), using two different feature extraction methods. The second experiment reproduced the same setup as the first one but using this time the third data set (News dataset). The third experiment tested a combination of keystrokes and n-gram features against the second dataset (Essay and Reviews). In the fourth experiment, we generated near-duplicate reviews and assessed the efficiency of the semantic measurement. All four experiments were run using 5-fold cross validation; in each validation round the dataset is divided into 80% for training and 20% for testing.

In the first two experiments, the algorithms were used to create learning models and then the learned models were used to predict the labels assigned to the testing data. Experiment results were then presented, analyzed and interpreted. Also, we added an analysis of the first and third datasets, to inspect the distribution of verbs, nouns, adverb in fake and truth content.

4.2 Experiment One

In this experiment, we used Dataset 1, which contains 800 fake and 800 real hotel reviews. From our analysis of sample data, we found out that fake content contains more function words and content words than honest reviews.

- Function words are words with little lexical meaning, which are usually used to express a grammatical relationship with other words (e.g., *the that, she, of*).
- Content words are the opposite, in the sense that they stand on their own by carrying specific purpose or meaning (not necessarily in connection to other words), such as, flew and mountain.

This means that fake reviews writers tend to use more filler/functions and content words than real reviews writers. Furthermore, they use more verbs and adverbs than real reviews writers. In contrast, real reviews contain more nouns and adjectives. This is emphasized by the word distribution in dataset 1, depicted by Figure 2

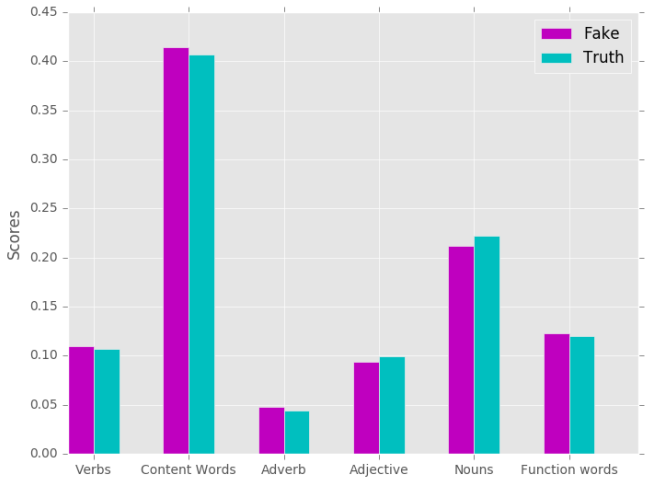

Figure 2: The Word distribution in dataset 1. Even though the difference is slight, it can be seen that fake content writers are using more content and filler words. Furthermore, they use more verbs and adjectives than real content writers.

Figures 3 and 4 depict the 30 most frequently used bi-gram in fake reviews and honest reviews, respectively, from the dataset.

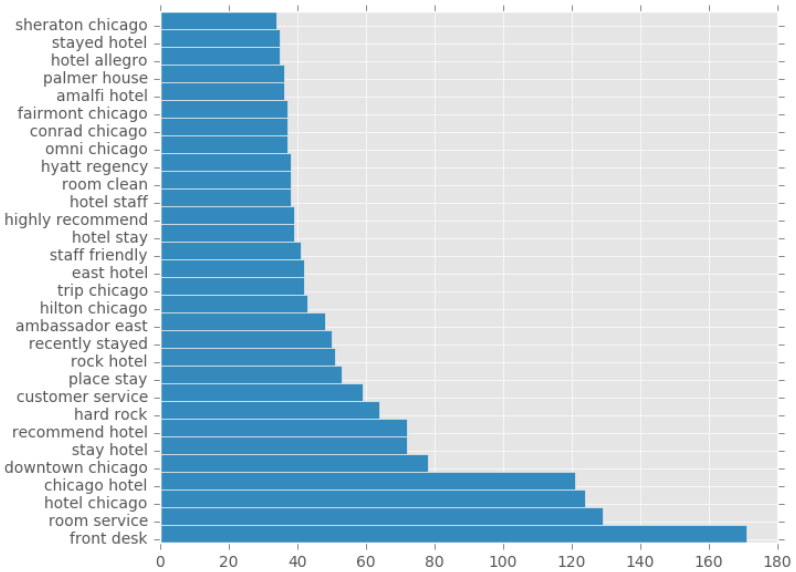

Figure 3: Top 30 Bi-gram in Fake reviews from Dataset 1

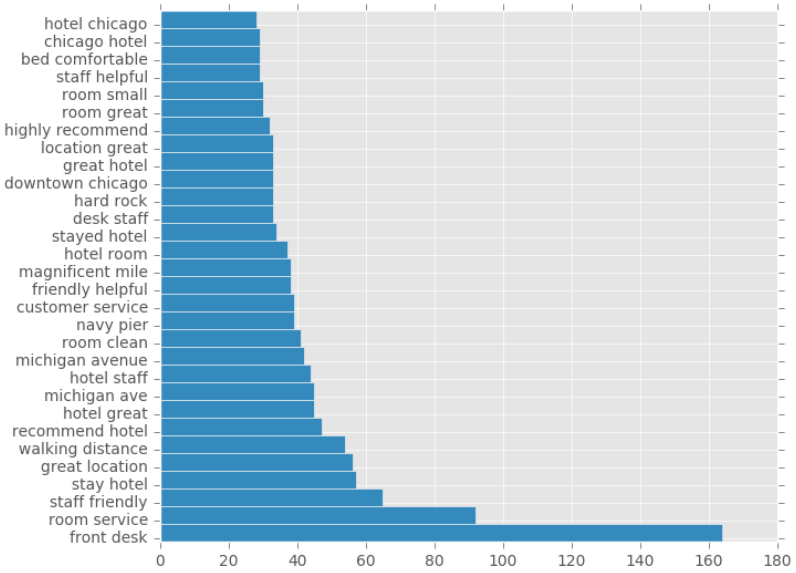

Figure 4: Top 30 Bi-gram in real reviews from Dataset 1

As can be seen, both depictions contain similar terms such as “front desk” and “room service.” However, there are some key differences. For instance, fake reviewers tend to use the hotel’s name more often, while honest reviewers tend to reference locations around the hotel's location to visit.

We run the aforementioned algorithms on Dataset 1, with the goal of predicting whether the reviews are truthful or fake. We studied two different features extraction methods, TF-IDF, and TF, and varied the size of the n-gram from n=1 to n=4. We also varied the number of features *p* (i.e., top features selected), ranging from 1000 to 50,000. The prediction results for the different classification algorithms are depicted in Tables 4.2 - 4.7.

Table 4.2: SVM Prediction Accuracy Dataset 1. The second row corresponds to the features size. Accuracy values are in %.

N-gram Size	**TF-IDF**				**TF**			
	1000	5000	10,000	50,000	1000	5000	10,000	50,000
Uni-Gram	83.0	82.0	83.0	82.0	83.0	82.8	79.0	82.0
Bi-Gram	80.0	83.0	82.0	80.0	80.0	83.0	82.0	80.0
Tri-Gram	73.0	78.0	79.0	75.0	75.0	78.0	77.0	75.0
Four-Gram	76.0	69.0	68.0	47.0	70.0	69.0	68.0	42.0

Table 4.3: LSVM Prediction Accuracy Dataset 1. The second row corresponds to the features size. Accuracy values are in %.

N-gram Size	**TF-IDF**				**TF**			
	1000	5000	10,000	50,000	1000	5000	10,000	50,000
Uni-Gram	85.0	85.0	87.0	86.0	82.0	84.0	84.0	84.0
Bi-Gram	77.0	85.0	90.0	85.0	75.0	79.0	78.0	79.0
Tri-Gram	74.0	80.0	79.0	80.0	76.0	73.0	74.0	75.0
Four-Gram	71.0	75.0	77.0	75.0	71.0	71.0	70.0	70.0

Table 4.4: KNN Prediction Accuracy Dataset 1. The second row corresponds to the features size. Accuracy values are in %.

N-gram Size	**TF-IDF**				**TF**			
	1000	5000	10,000	50,000	1000	5000	10,000	50,000
Uni-gram	75.0	72.0	73.0	71.0	64.0	63.0	60.0	63.0
Bi-Gram	75.0	77.0	78.0	77.0	55.0	57.0	56.0	57.0

Tri-Gram	53.0	72.0	73.0	72.0	55.0	54.0	57.0	54.0
Four-Gram	48.0	55.0	69.0	69.0	56.0	55.0	52.0	47.0

Table 4.5: DT Prediction Accuracy Dataset 1. The second row corresponds to the features size. Accuracy values are in %.

N-gram Size	**TF-IDF**				**TF**			
	1000	5000	10,000	50,000	1000	5000	10,000	50,000
Uni-Gram	69.0	71.0	62.0	69.0	65.0	68.0	66.0	71.0
Bi-Gram	71.0	70.0	69.0	71.0	71.0	73.0	73.0	73.0
Tri-Gram	70.0	70.0	72.0	57.0	71.0	70.0	71.0	0.62
Four-Gram	67.0	67.0	67.0	68.0	64.0	72.0	68.0	67.0

Table 4.6: SGD Prediction Accuracy Dataset 1. The second row corresponds to the features size. Accuracy values are in %.

N-gram Size	**TF-IDF**				**TF**			
	1000	5000	10,000	50,000	1000	5000	10,000	50,000

Uni-Gram	85.0	85.0	85.0	86.0	85.0	83.0	82.0	82.0
Bi-Gram	79.0	84.0	83.0	86.0	76.0	82.0	80.0	80.0
Tri-Gram	71.0	78.0	80.0	82.0	69.0	78.0	80.0	81.0
Four-Gram	72.0	76.0	78.0	76.0	71.0	75.5	76.0	77.0

Table 4.7: LR Prediction Accuracy Dataset 1. The second row corresponds to the features size. Accuracy values are in %.

N-gram Size	**TF-IDF**				**TF**			
	1000	5000	10,000	50,000	1000	5000	10,000	50,000
Uni-Gram	85.0	85.0	87.0	87.0	84.0	84.0	82.0	83.0
Bi-Gram	79.0	86.0	86.0	85.0	79.0	83.0	84.0	84.0
Tri-Gram	76.0	78.0	80.0	79.0	74.0	79.0	79.0	79.0
Four-Gram	72.0	77.0	77.0	76.0	72.0	79.0	77.0	76.0

As it can be seen from the above results, Linear-based classifiers (Linear SVM, SDG, and Logistic regression) achieved better results than nonlinear ones. We obtained 90 % accuracy using Linear SVM. It is also noticeable that they achieved better accuracy with

more feature values since they achieved it using 10,000 and 50,000 feature values. Also, we can observe that increasing the n-gram size lowers the accuracy when using all the classifiers. TF-IDF performed better than TF, which is expected given that fake and honest reviews contained similar words. The lowest accuracy of 47.2% was achieved using KNN with four-gram words and 50,000 feature values.

4.3 Experiment Two

In this experiment, we used Dataset 3 which consists of News articles from different sources. In the start of our research, we applied our model on a combination of news articles from different years with a broader variety of political topics. Our model achieved 98% accuracy when using this type of data. Thus, we decided to collect our dataset so we can require fake and real articles from the same year and even same month. Furthermore, we decided to limit the scope of the articles. Thus we only focused on news articles that revolve around the 2016 US elections and the articles that discuss topics around it. In total, we picked 2000 articles from real and fake articles we collected, 1000 fake articles and 1000 real articles. The 2,000 articles represent a subset of the dataset described in the previous section that focuses only politics. Similar to experiment one, the analysis of sample data corroborates the fact that fake contents contain more function words and content words than honest articles. Also akin to the first dataset, honest news articles contain more nouns and adjectives. The fake news contains more adverbs and verbs. However, as shown in Figure 5 in contrast to Dataset 1, fake writers in Dataset 3 used less verb and adjectives in their content than real content writers.

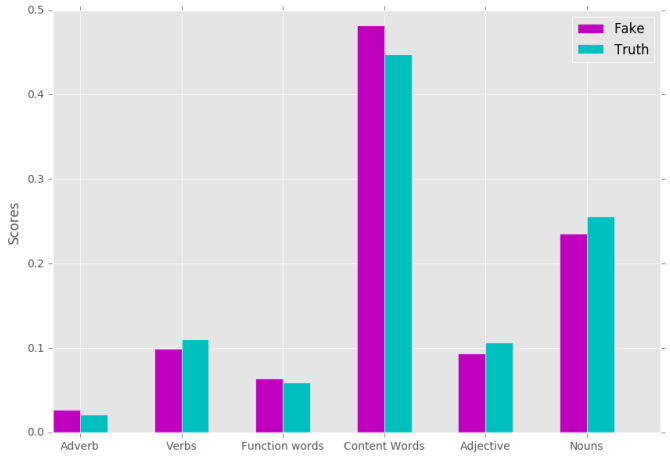

Figure 5: The Word distribution in Dataset 3. Even though the difference is slight it can be seen that fake content writers are using more content and filler words. However, in contrast to Dataset 1, fake writers in Dataset 3 used less verb, and adjectives in their text.

Figures 6 and 7 show the 30 most frequently used bi-grams in fake and honest news from the dataset.

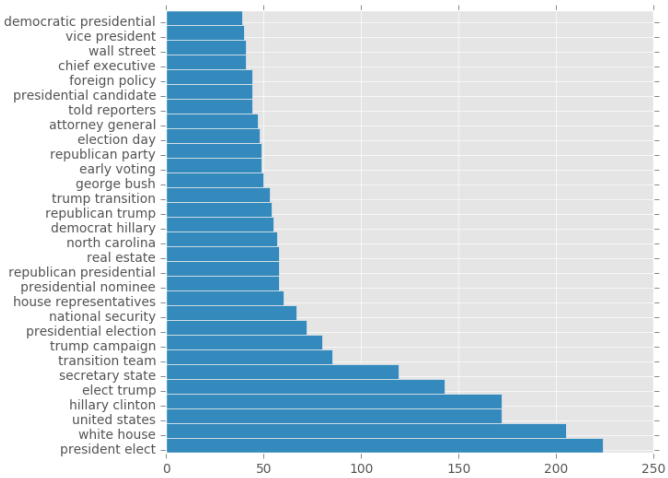

Figure 6: Top 30 Bi-gram in real News from Dataset 3

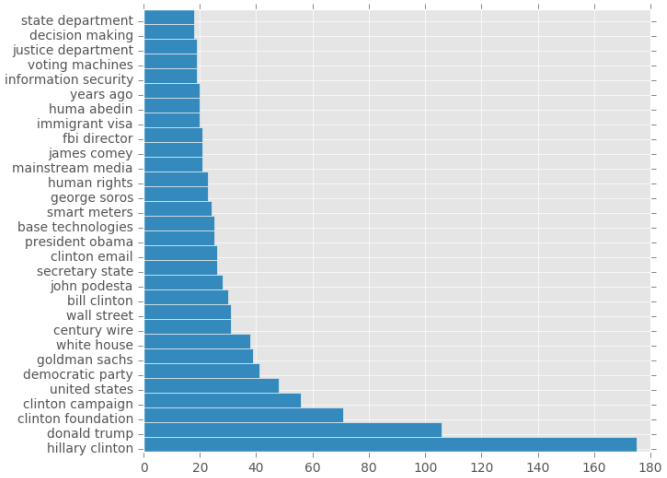

Figure 7: Top 30 Bi-grams in Fake News from Dataset 3

As it can be seen, both types of articles contain similar terms such as "Hillary Clinton" and "United States." However, in contrast to the first dataset, we see more variety of terms in both classes, even though we have targeted articles that cover similar incidents and topics.

We used the same approach as in the previous experiment by running the algorithms on Dataset 3, with the goal of predicting whether the articles are truthful or fake. Tables 4.8-4.13 show the obtained results.

Table 4.8: SVM Prediction Accuracy Dataset 1. The second row corresponds to the features size. Accuracy values are in %.

N-gram Size	**TF-IDF**				**TF**			
	1000	5000	10,000	50,000	1000	5000	10,000	50,000
Uni-gram	84.0	86.0	84.0	84.0	85.0	72.0	69.0	69.4
Bi-Gram	78.0	73.0	67.0	54.0	68.0	51.0	47.0	47.0

Tri-Gram	71.0	59.0	53.0	48.0	53.0	47.0	53.0	47.0
Four-Gram	55.0	37.0	37.0	45.0	47.0	48.0	40.0	47.0

Table 4.9: LSVM Prediction Accuracy Dataset 1. The second row corresponds to the features size. Accuracy values are in %.

N-gram Size	**TF-IDF**				**TF**			
	1000	5000	10,000	50,000	1000	5000	10,000	50,000
Uni-Gram	89.0	89.0	89.0	92.0	87.0	87.0	87.0	87.0
Bi-Gram	87.0	87.0	88.0	89.0	86.0	83.0	82.0	82.0
Tri-Gram	84.0	85.0	86.0	87.0	86.0	84.0	84.0	79.0
Four-Gram	71.0	76.0	76.0	81.0	70.0	70.0	70.0	61.0

Table 4.10: KNN Prediction Accuracy Dataset 1. The second row corresponds to the features size. Accuracy values are in %.

N-gram Size	**TF-IDF**				**TF**			
	1000	5000	10,000	50,000	1000	5000	10,000	50,000
Uni-Gram	79.0	83.0	82.0	83.0	77.0	70.0	68.0	68.0

Bi-Gram	67.0	65.0	68.0	64.0	62.0	55.0	51.0	45.0
Tri-Gram	73.0	68.0	65.0	67.0	76.0	63.0	57.0	46.0
Four-Gram	69.0	68.0	68.0	58.0	67.0	54.0	56.0	43.0

Table 4.11: DT Prediction Accuracy Dataset 1. The second row corresponds to the features size. Accuracy values are in %.

N-gram Size	**TF-IDF**				**TF**			
	1000	5000	10,000	50,000	1000	5000	10,000	50,000
Uni-Gram	88.0	88.0	89.0	89.0	83.0	88.0	88.0	80.0
Bi-Gram	85.0	85.0	85.0	84.0	84.0	87.0	87.0	84.0
Tri-Gram	86.0	86.0	87.0	85.0	86.0	86.0	84.0	86.0
Four-Gram	74.0	74.0	71.0	74.0	67.0	67.0	70.0	67.0

Table 4.12: SGD Prediction Accuracy Dataset 1. The second row corresponds to the features size. Accuracy values are in %.

N-gram Size	**TF-IDF**				**TF**			
	1000	5000	10,000	50,000	1000	5000	10,000	50,000

Uni-Gram	88.0	86.0	88.0	89.0	87.0	86.0	89.0	85.0
Bi-Gram	86.0	85.0	87.0	86.0	85.0	84.0	85.0	84.0
Tri-Gram	84.0	85.0	86.0	86.0	85.0	85.0	87.0	87.0
Four-Gram	70.0	72.0	74.0	80.0	72.0	73.0	72.0	78.0

Table 4.13: LR Prediction Accuracy Dataset 1. The second row corresponds to the features size. Accuracy values are in %.

N-gram Size	**TF-IDF**				**TF**			
	1000	5000	10,000	50,000	1000	5000	10,000	50,000
Uni-Gram	83.0	89.0	89.0	89.0	89.0	89.0	83.0	89.0
Bi-Gram	87.0	87.0	88.0	88.0	87.0	85.0	86.0	86.0
Tri-Gram	86.0	85.0	88.0	87.0	83.0	83.0	83.0	82.0
Four-Gram	70.0	76.0	75.0	81.0	68.0	67.0	67.0	61.0

Similar to the results obtained in Experiment 1, Linear-based classifiers (Linear SVM, SDG, and Logistic regression) achieved better results than nonlinear ones. However, nonlinear classifiers achieved good results too; DT achieved 89% accuracy. The highest

accuracy was achieved using Linear SVM as 92%. This classifier performs well no matter the number of feature values used. Also with the increase of n-gram (Tri-gram, Four-gram), the accuracy of the algorithm decreases. However tri-gram features still performed considerably better on this dataset. Furthermore, TF-IDF outperformed TF. The lowest accuracy achieved is 45% when using KNN with features size of 50.000.

We conducted additional experiments by running our model on the dataset of Adali and Horne [15], consisting of real news from BuzzFeed and other news websites, and satires from Burfoot and Baldwin's satire [33] dataset. We obtained 87% accuracy using n-gram features and LSVM algorithm when classifying fake news against real new, which is much better than the 71% accuracy obtained by the authors on the same dataset.

4.4 Experiment Three

In this experiment, we used Dataset 2, which contains three topics: gun control and gay rights (as essays), and restaurant reviews. Since Linear SVM outperformed all classifiers in the previous two experiments, we decided to use it in this experiment. For this experiment, we used 5-fold cross-validation with parameter tuning. We used a grid search to pass a combination of the parameters to the classifier, and performed the classification using the best combination. Next, we added the time and editing pattern features and re-classified the data. The performance results are presented in Table 4.14.

Table 4.14: LSVM Prediction Accuracy Dataset 2. Accuracy values are in %.

Topic	**TF-IDF only**	**TF-IDF + Time features**	**TF-IDF , E (editing pattern) , time features**
Restaurant	78.0	78.0	79.0
Gay-rights (Support)	92.0	92.0	93.0

Gay-rights (oppose)	88.0	90.0	90.0
Gun-control (Support)	82.0	83.0	84.0
Gun-control (oppose)	90.0	90.0	91.0

Similar to the results obtained by Choi [16], we were able to increase the accuracy of the n-gram model. However, we were only able to achieve a 1% increase. We observed that it is the combination of editing pattern features and the time feature with the n-gram that increases the accuracy.

4.5 Experiment Four

To assess the semantic measurement model, we selected randomly 20 essays from the second dataset (Dataset 2). Then, we created an algorithm that changes a text based on the percentage we provide as input, for example, if we give it 10 % it will change 10% of the text. When we specify the percentage, it randomly selects some words in the text and replaces them with similar words from Wordnet. For example, for 50% it will replace half of the words in the text with words with similar meanings. We used the algorithm and a human editor to generate fake near-duplicated contents from the different essays. By varying the change percentage and applying the algorithm to the different essays, we created ten groups each consisting of 20 fake near-duplicated contents from the different essays.

For each group, we calculated the semantic similarity between each duplicated document and the original document. Table 4.15 illustrates the average semantic score for each of the ten groups of fake duplicated contents.

As it can be observed, the semantic measurement value decreases with the increase of the change amount in the contents. We can see that the score starts to deteriorate after the 55% mark. We were able to sucessfuly identify 70 % of the near duplicated content we created. We had 200 duplicated contents, and the model was able to identify 140 of them.

Table 4.15: The average semantic measurement of each of the ten groups of fake duplicated documents

The percentage of change	**Avg. Semantic Measurement**
10 %	0.95
15 %	0.91
25%	0.85
35 %	0.78
45 %	0.75
55 %	0.64
65 %	0.61
75 %	0.54
85 %	0.47
95 %	0.39

4.6 Summary

This section discusses the results of various experiments. It starts by discussing the effect of the size of n-grams and features space on the classifier performance.

Based on results of this research, the following conclusions were made. The type of data used for training and the size of features affects the classifier performance. We saw an improvement in the accuracy results when we used Dataset 3, which is considered a bigger data set with a variety of words and terms. Furthermore, as observed with the increase of the features, the majority of the classifiers achieved a higher accuracy. The majority of our higher accuracy was achieved using 50,000 and 10,000 features. We also saw that an increase in the n-gram size would cause a decrease in the accuracy. In both datasets three and one, the unigram and bi-gram performed better than the trigram and quad-gram.

The Keystroke features did help with the performance of the classifier, mainly through the editing pattern features. It seems that users who write fake content tend to edit more. We believe that using the writing speed of the users to detect deceptive content is not reliable, given that different individuals have different linguistic proficiency and in turn, their speed is vastly different. Some humans will spend more time creating a lie while other might take more time telling the truth. The semantic measurement achieved rather good results. We were able to identify near-duplicated content with up to 65% of the text changed. In reality, the majority of near-duplicated contents are less than 25% different from the original contents. The spammer tends to report their early work and only change certain words and title most of the time. Thus, we believe that our semantic measurement approach will be able to classify near-duplicate correctly in real-world data.

Comparison of previous works and our results for opinion spam detection models are presented in Table 4.16. As it can be noted, our approach outperforms most of the existing work.

Table 4.16: Comparison of previous works and our work results for opinion spam detection

Dataset	**Classifier**	**Features**	**Performance Metrics**	**Score**	**Reference**
Reviews Amazon website	Logistic Regression	Review and reviewer features	AUC	78 %	[12]
Ott et al. dataset (Dataset 1)	SVM	LIWC + Bigrams	Accuracy	89 %	[14]
Ott et al. dataset (Dataset 1)	SVM	Stylometric features	F-measure	84%	[14]
Dataset 2	LSVM	TF-IDF , E (editting pattern) , time features	Accuracy	93%	Our Results
Dataset 1 (Ott et al.)	LSVM	Bi-gram	Accuracy	90%	Our Results
Dataset 3	LSVM	Uni-gram	Accuracy	92%	Our Results
Buzzfeed news and random new articles	SVM	Text-based features	Accuracy	71 %	[15]

CHAPTER 5: CONCLUSION AND FUTURE WORK

5.1 Summary

In recent years, deceptive content such as fake news and fake reviews have increasingly become a dangerous prospect, for online users. Fake reviews have affected consumer and stores alike. The use of hired writers to produce fake reviews to boost sales is also growing. It has become difficult for consumers to recognize good products from bad ones based on reviews. Furthermore, the problem of fake news has gained attention in 2016, especially in the aftermath of the last US presidential elections. Recent statistics and research show that 62% of US adults get news on social media [40,41]. Most of the popular fake news stories were more widely shared on Facebook than the most popular mainstream news stories [42]. A sizable number of people who read fake news stories have reported that they believe them (Silverman and Singer-Vine 2016) more than news from mainstream media. Dewey [43] claim that fake news played a huge role in the 2016 US election and that they continue to affect people opinions and decisions.

Opinion spam and fake news are a closely related phenomenon as both consist of writing and spreading false information or beliefs. The opinion spam problem was formulated for the first time a few years ago, but it has quickly become a growing research area due to the abundance of user-generated content. It is now easy for anyone to either write fake reviews or write fake news on the web.

In this thesis, we focused on the problem of detecting opinion spam using n-gram analysis. One of the advantages of the n-gram analysis method, we designed is that it worked on multiple type of fake content. The reason is it solely relies on the text of the fake content, and it ignores the behavior of the spammers who publish the content. N-gram models were able to detect fake reviews, fake news, and fake essays with an almost exact efficiency.

The achieved accuracy illustrates that fake content will use different words than real content users. In addition it was able to identify fake content written by different users such as the content that existed in dataset three (fake news), and fake content written by same users such as the content that was in dataset two (essays and restaurant reviews).

Furthermore, we investigated the impact of keystrokes features on the accuracy of n-gram classifiers. In conclusion, we discovered that keystrokes features would improve the accuracy of our n-gram analysis model. However the result was not promising enough to be used as an independent solution. In addition, we think users will not have strong acceptance to this solution, given we are collecting their keystrokes data, and that involves privacy issues.

We also applied semantic similarity metrics to detect near-duplicated content. Experimental results showed that semantic similarity measurement could detect near duplicated deceptive content similar to the ones available on online reviews websites. However, it will not be able to detect unique fake content and that is because having two contents that are semantically similar does not mean they are completely fake, especially if they are covering the same topic. We can improve the semantic measurement, if we build a topic-based database that have semantic relationship between the topics.

The n-gram features performed well on real-world data and pseudo data. Furthermore, they performed better when applied to the fake news data. Lastly, we added the keystrokes features to n-gram, and although it improved the accuracy of the n-gram features, it did not have a significant effect in predicting the deceptive intent of the users.

5.2 Future Work

In our future work, we would like to incorporate in our model, statistical features, and features that reflect the writer's styles such as the number of slang words or filler word in the text. We would like to investigate the effect of adding the additional features on our n-gram model, will the accuracy of the model change and how will it change.

There is lack of accurately labeled real-world datasets. Thus we would like to explore unsupervised and semi-supervised methods to detect deceptive content, even though the available model that applies unsupervised and semi-supervised methods are unable to match the performance of supervised learning methods. I believe that looking into the unsupervised learning approach is worth exploring.

Reviews and articles are a primary source of information. There is a massive amount of data in the real worlds. We would like to apply our models on big data available, to investigate how it will perform. Furthermore, given that high dimensionality issue will arise we would like to explore the effect of feature selection techniques such as information gains and chi-square, in our future work.

REFERENCES

1. Anderson, M. and Anderson, M. (2017). “88% Of Consumers Trust Online Reviews As Much As Personal Recommendations”. [online] Search Engine Land. Available at: http://searchengineland.com/88-consumers-trust-online-reviews-much-personal-recommendations-195803
2. Miller, C. (2017). “Cosmetic Surgery Company Settles Case of Faked Reviews”. [online] Nytimes.com.
3. 7NEWS. (2017). “Woman Paid To Post Five-Star Google Feedback”. [online] Available at: http://www.thedenverchannel.com/news/woman-paid-to-post-five-star-google-feedback .
4. BBC News. (2017). “Samsung probed in Taiwan over 'fake web reviews' “- BBC News. [online] Available at: http://www.bbc.com/news/technology-22166606.
5. Helft, M. (2017). “Reverb P.R. Firm Settles Case on Fake Reviews”. [online] Nytimes.com. Available at: http://www.nytimes.com/2010/08/27/technology/27ftc.html
6. The Verge. (2017). “Your short attention span could help fake news spread.” [online] Available at: https://www.theverge.com/2017/6/26/15875488/fake-news-viral-hoaxes-bots-information-overload-twitter-facebook-social-media .
7. Lemann, N. (2017). “Solving the Problem of Fake News”. [online] The New Yorker. Available at: http://www.newyorker.com/news/news-desk/solving-the-problem-of-fake-news
8. Levin, S. (2017). “Pay to sway: report reveals how easy it is to manipulate elections with fake news”. the Guardian. Available at: https://www.theguardian.com/media/2017/jun/13/fake-news-manipulate-elections-paid-propaganda.
9. Gu, L., Kropotov, V. and Yarochkin, F. (2017). “The Fake News Machine , How Propagandists Abuse the Internet and Manipulate the Public”. 1st ed. [pdf] Trend

Micro, p.81. Available at: https://documents.trendmicro.com/assets/white_papers/wp-fake-news-machine-how-propagandists-abuse-the-internet.pdf?_ga=2.117063430.1073547711.1497355570-1028938869.1495462143.

10. Schulten, K. (2017). “Skills and Strategies | Fake News vs. Real News: Determining the Reliability of Sources”. [online] The Learning Network. Available at: https://learning.blogs.nytimes.com/2015/10/02/skills-and-strategies-fake-news-vs-real-news-determining-the-reliability-of-sources/.
11. Ye, Shaozhi, and Shyhtsun Felix Wu. "Measuring message propagation and social influence on twitter. com." SocInfo 10 (2010): 216-231.
12. Jindal, N., & Liu, B. (2008). “Opinion Spam and Analysis”. In Proceedings of the 2008 International Conference on Web Search and Data Mining (pp. 219–230). New York, NY, USA: ACM. doi:10.1145/1341531.1341560
13. Schow, A. (2017). “The 4 Types of ‘Fake News’”. [online] Observer. Available at: http://observer.com/2017/01/fake-news-russia-hacking-clinton-loss/
14. Ott, M., Choi, Y., Cardie, C., & Hancock, J. T. (2011). “Finding Deceptive Opinion Spam by Any Stretch of the Imagination”. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1 (pp. 309–319). Stroudsburg, PA, USA: Association for Computational Linguistics. Retrieved from http://dl.acm.org/citation.cfm?id=2002472.2002512
15. Horne B.D. and S. Adali (2017). "This Just In: Fake News Packs a Lot in Title, Uses Simpler, Repetitive Content in Text Body, More Similar to Satire than Real News", The 2nd International Workshop on News and Public Opinion at ICWSM
16. Choi, Yejin. "Keystroke patterns as prosody in digital writings: A case study with deceptive reviews and essays". Empirical Methods on Natural Language Processing (EMNLP) 6 (2014).

17. Li, Y., McLean, D., Bandar, Z. A., O'shea, J. D., & Crockett, K. (2006). Sentence similarity based on semantic nets and corpus statistics. IEEE transactions on knowledge and data engineering, 18(8), 1138-1150.
18. Mukherjee, A., Venkataraman, V., Liu, B., & Glance, N. (2013). “Fake Review Detection: Classification and Analysis of Real and Pseudo Reviews”. UIC-CS-03-2013. Technical Report.
19. Mukherjee, A., Venkataraman, V., Liu, B., & Glance, N. (2013). “What yelp fake review filter might be doing”. In Proceedings of the International Conference on Weblogs and Social Media.
20. Li J, Ott M, Cardie C, Hovy E (2014). “Towards a general rule for identifying deceptive opinion spam”. Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 1566–1576, Baltimore, Maryland, USA, June 23-25 2014. ACL
21. Shojaee, S., Murad, M. A. A., Azman, A. B., Sharef, N. M., & Nadali, S. (2013, December). Detecting deceptive reviews using lexical and syntactic features. In Intelligent Systems Design and Applications (ISDA), 2013 13th International Conference on (pp. 53-58). IEEE.
22. Algur, S. P., Patil, A. P., Hiremath, P. S., & Shivashankar, S. (2010, December). Conceptual level similarity measure based review spam detection. In Signal and Image Processing (ICSIP), 2010 International Conference on (pp. 416-423). IEEE.
23. Lau R.Y., Liao S.Y., Kwok RCW, Xu K, Xia Y, Li Y (2011). “Text mining and probabilistic language modeling for online review spam detecting”. ACM Trans Manage Inf Syst 2(4):1–30
24. Lim, E. P., Nguyen, V. A., Jindal, N., Liu, B., & Lauw, H. W. (2010, October). Detecting product review spammers using rating behaviors. In Proceedings of the 19th ACM international conference on Information and knowledge management (pp. 939-948). ACM.
25. Mukherjee, A., Kumar, A., Liu, B., Wang, J., Hsu, M., Castellanos, M., & Ghosh, R. (2013, August). Spotting opinion spammers using behavioral footprints. In

Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 632-640). ACM.

26. Fei, G., Mukherjee, A., Liu, B., Hsu, M., Castellanos, M., & Ghosh, R. (2013). Exploiting Burstiness in Reviews for Review Spammer Detection. ICWSM, 13, 175-184.
27. Xie, S., Wang, G., Lin, S., & Yu, P. S. (2012, April). Review spam detection via time series pattern discovery. In Proceedings of the 21st International Conference on World Wide Web (pp. 635-636). ACM.
28. Wang, G., Xie, S., Liu, B., & Yu, P. S. (2012). Identify online store review spammers via social review graph. ACM Transactions on Intelligent Systems and Technology (TIST), 3(4), 61.
29. Mukherjee, Arjun, Bing Liu, and Natalie Glance. "Spotting fake reviewer groups in consumer reviews". Proceedings of the 21st international conference on World Wide Web. ACM, 2012.
30. Feng, S., Xing, L., Gogar, A., & Choi, Y. (2012). Distributional Footprints of Deceptive Product Reviews. ICWSM, 12, 98-105.
31. Li, F., Huang, M., Yang, Y., & Zhu, X. (2011, July). Learning to identify review spam. In IJCAI Proceedings-International Joint Conference on Artificial Intelligence (Vol. 22, No. 3, p. 2488).
32. Rubin V. L., Chen Y., and N. J. Conroy. 2015. “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 (ASIST '15). American Society for Information Science, Silver Springs, MD, USA, Article 83, 4 pages.
33. Burfoot C., and T. Baldwin, “Automatic satire detection: are you having a laugh?”, Proceedings of the ACL-IJCNLP 2009 Conference Short Papers, August 04-04, 2009, Suntec, Singapore
34. William Yang Wang. " Liar, Liar Pants on Fire: A New Benchmark Dataset for Fake News detection”. arXiv preprint arXiv:1705.00648(2017).

35. Rubin, V. L., Conroy, N. J., Chen, Y., & Cornwell, S. (2016). Fake News or Truth? Using Satirical Cues to Detect Potentially Misleading News. In Proceedings of NAACL-HLT (pp. 7-17).
36. Shannon, Claude E. "A mathematical theory of communication". ACM SIGMOBILE Mobile Computing and Communications Review 5.1 (2001): 3-55.
37. Mason, Jane E. "An n-gram based approach to the automatic classification of web pages by genre". (2009).
38. Miller, George A. "WordNet: a lexical database for English". Communications of the ACM 38.11 (1995): 39-41.
39. Li, Yuhua, Zuhair A. Bandar, and David McLean. "An approach for measuring semantic similarity between words using multiple information sources". IEEE Transactions on knowledge and data engineering 15.4 (2003): 871-882.
40. Gottfried J., and E. Shearer. "News use across social media platforms". Pew Research Center 26 (2016)
41. Gottfried, J., Barthel, M., Shearer, E., & Mitchell, A. (2016). The 2016 presidential campaign–A news event that's hard to miss. Pew Research Center, 4.
42. Silverman, C., and J. Singer-Vine, (2016). "Most Americans who see fake news believe it, new survey says". BuzzFeed News.
43. Dewey, Caitlin. "Facebook has repeatedly trended fake news since firing its human editors". Washington Post (2016)