M.Tech (Information Systems) Vatsala Panwar 2019

DECEPTION DETECTION ON LIES, REVIEWS AND TRIALS

Major-II project

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE

OF

MASTER OF TECHNOLOGY

IN

INFORMATION SYSTEMS

Submitted by:

Vatsala Panwar

(2K17/ISY/15)

Under the supervision of

Dr. KAPIL SHARMA

DEPARTMENT OF INFORMATION TECHNOLOGY

DELHI TECHNOLOGICAL UNIVERSITY

(Formely Delhi College of Engineering)

Bawana Road, Delhi-110042

MAY, 2019

DELHI TECHNOLOGICAL UNIVERSITY

(Formely Delhi College of Engineering)

Bawana Road, Delhi-110042

CANDIDATE'S DECLARATION

I, Vatsala Panwar, Roll No. 2K17/ISY/15, student of M.Tech. (Information Systems) hereby declare that the project Dissertation titled "Deception detection on lies, reviews and trials" which is submitted by me to the Department of Information Technology, Delhi Technological University, Delhi (formerly Delhi College of Engineering) in partial fulfillment of the requirements for the award of the degree of Master of Technology, is original and not copied from any source without proper citation. This work has not previously formed the basis of award of any Degree, Diploma Associateship, Fellowship or other similar title or recognition.

Place: New Delhi

Date:

(VATSALA PANWAR)

DEPARTMENT OF INFORMATION TECHNOLOGY

DELHI TECHNOLOGICAL UNIVERSITY

(Formely Delhi College of Engineering)

Bawana Road, Delhi-110042

<u>CERTIFICATE</u>

I hereby certify that the project Dissertation titled “Deception Detection on lies, reviews and trials”, which is submitted by Vatsala Panwar, Roll No. 2K17/ISY/15, Department of Information Technology, Delhi Technological University, Delhi (formerly Delhi College of Engineering) in partial fulfillment of the requirements for the award of the degree of Master of Technology, is a record of the project work carried out by the student under my supervision.

To the best of my knowledge this work has not been submitted in part or full for any Degree or Diploma to this University or elsewhere.

Place: New Delhi (**DR. KAPIL SHARMA**)

Date**:** **SUPERVISOR**

Professor

Department of Information Technology

Delhi Technological University

(Formely Delhi College of Engineering)

Bawana Road, Delhi-110042

ACKNOWLEDGEMENT

The realization of this project involved a lot of guidance and help from many people and I am really grateful to them for their support. My grateful thanks are extended to Delhi Technological University for providing me the platform and assistance to carry on this work. I would also like to acknowledge the Information Technology department for providing me with useful and constructive resources.

I am grateful and fortunate to get inspiration as well as valuable suggestions from my family, who are my biggest support and my seniors who facilitated me to complete this project by giving their on-the-point tips.

VATSALA PANWAR

ABSTRACT

Since past few years, deceptive contents such as fake reviews posted on online shopping sites, also identified as opinion spam, or deceptive contents in any form either verbal or textual have become a nuisance because of exponential advancement in information technology and communication and hence the ease of creation/distribution of any kind of information.

Fake reviews affect the consumers' decision making abilities and the reputation of stores across all the platforms. Also, the need for tackling and identifying deceptive contents from the day-to-day life using the headway in computation and technology is also being understood. The problem of opinion spamming was discovered not long ago, even then it began to be a promising research front because of the ever-growing abundance of computer generated data, formally, text based computer mediated communication (TB-CMC). Since it has become fairly easy to write and/or propagate any deceptive contents with just a click, so the algorithms tackling such problems need to out-perform them to eradicate them or at least stop them as quickly as they are generated.

The lack of efficient ideas and algorithms, along with the technology to implement them pose a huge hurdle in the way of automating the task of deception detection, since human experts can’t be relied for it anymore, due to various factors such as lack of time, efficiency, knowledge, manpower, great amount of data generated, ever changing forms of deceptive data etc.

In this project, we have developed linguistic, syntactic and semantic models to detect fake/deceptive contents especially fake reviews, lies and deceptive speeches. We experimented using different methods of feature extraction for different categories of features and combined and evaluated them on various classification models of machine learning. The evaluation was done on open domain deception data, product reviews data and real life trial data to further identify the features and classification techniques that suit the domain and purpose of the experiment.

We observed that the greatest results were achieved on real-life trial data while the lowest results were obtained on open-domain deception data. The reason for that might be the lack of domain/target, pseudo lies vs. deception in real situations and large dataset. The features that performed well in general for the task of deception detection were n-grams, word embeddings, POS features, and TF-IDF features. The results were motivating and signify scope of further research in this direction.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS, ABBREVIATIONS AND NOMENCLATURE

SVM	Support vector machine
SGD	Stochastic gradient descent
NB	Naïve bayes
Etc.	Et cetera
KNN	K nearest neighbor
AUC	Area under ROC curve
CNN	Convoluted neural network
RNN	Recurrent neural network
PCFG	Probabilistic context free grammar
MLP	Multi layered perceptron
ROC-AUC	Receiver operating characteristic curve
vs.	Versus
POS	Parts of speech
NLP	Natural language processing
Fig.	Figure
Avg.	Average
w.r.t	With respect to
ML	Machine learning

CHAPTER 1

INTRODUCTION

1.1. GENERAL

Since past few years, the amount of online content has increased significantly with people and machines generating and sharing data on a tremendous scale. These contents mostly played a major role in influencing opinions and minds of the unsuspecting people. These deceptive contents produced for various reasons such as gaining popularity, pushing a particular propaganda, gaining money or other financial profit, defaming an institution or person, can seep into varied forms of communication or platforms like social networks, online shopping, emails or any other sources of alleged mis-information.

This mis-information can be in any form such as fake news, fake opinions or reviews, or just any other information containing deceptive contents. In earlier days, human interaction or print media acted as a medium for spread and propagation of false information, hoaxes and rumors. This resulted in slow diffusion and hence less impact and damage but due to the evolution of the World Wide Web and information technologies, it has become fairly easy to let an information spread to the masses. As indicated by a report by the Jumpshot Tech Blog1 that just 20% of internet traffic directed to reputable websites, while 50% of it went to fake news websites through Facebook links [1].

The consequences of deceptive information, either directly or indirectly can be very damaging and dangerous. Information pieces containing deceptive contents with malicious purposes are a great threat to mankind with people using it for phishing, scamming, fraud, cyber bullying, cyber terrorism, social, economic or commercial propagandas against individuals or organizations etc. Since every dimension of technology is being used in its depth to create and propagate information which might not be true, hence the same powers of technology must be used to segregate true information from the vast ocean of data that contains fabricated contents in huge amounts.

Since last few years the content, propagation, origin and commercialization of information pieces have been taken control majorly by Social media giants [2]. Buyers are increasingly using online reviews as a major source of information for deciding the brand and products to buy. Eventually, these potential customers are a target area for brands and companies that want to influence their minds to increase their sales or downsize their rivals' business. Users give their feedback to share their experience, in form of reviews about a company. Customers' reviews can hugely impact their businesses either positively or negatively. Similarly online reviews also hold this power with an additional advantage of anonymity or genuine verification that isn't guaranteed. Eventually it becomes a way of manipulating customers' decisions and remarks about a specific business by spiking the reviews with false positive or negative reviews. Companies sometimes force their own employees to write positive reviews about the organization or buy spammers, generally called opinion spammers in this case, whose job is to write fabricated reviews either targeting a company or praising their own business services and products.

There have been multiple news items stating that people are brought in by associations to uplift their own reputation. According to BBC news, Samsung reportedly paid students to write fake reviews criticizing its rival company HTC and praising Samsung products [3]. In another report by New York Times [4], an electronic company gives customers offer of $2 per star of rating that they give to its products on Amazon. As indicated by ABC news [5], Accounts of 50 Google users were discovered that were hired by companies to write repetitive positive reviews about them. Some of these actually influence the minds of potential customers. For Example, according to a report, 72% of these buyers are ready to believe the reputation of any company with positive reviews, without any second thoughts while online reviews are considered as good as offline recommendations by 88 % of customers [6].

1.2. PROBLEM STATEMENT

In this project, we had tried to understand the wide area of deception detection of text data and it impact, scope and complexity and propose a solution that can automatically detect deceptive contents using artificial intelligence, or machine learning in general. We take into account 3 types of deceptive texts, namely, fake reviews, lies and truths obtained from people with open domain and trial data containing verbal cues or transcripts.

Fake reviews are a real challenge because of the damage it can do to a business or its consumers. It is a form of cheating in indirect way because a would be customer never gets to know the real feedback of a product or service and if such reviews are huge in number then they can easily modify the buying patterns or decisions of consumers, without them even realizing that they are being fooled to buy or not buy a certain product. For businesses it is equally damaging since it can lead to deep impact on its reputation and financial losses. Posting fake reviews and other contents is illegal and people or business involved in it could face legal action for it, besides their business reputation getting a setback. The other form of opinion spam is fake news since it contains fabricated contents and opinions of someone who wants to push forward certain propaganda or gain financial benefits etc. Fake contents are everywhere, be it, cyber fraud, spamming, phishing, email scams, and any form of cyber crimes or non cyber crimes. Criminals mostly make use of deceptive language to con and dupe people or simply to dis-inform them to achieve their mischievous plans but all this needs to be tackled effectively and strongly.

While it is easier to understand the motive and impact of publishing fake contents, the mindset and propaganda behind it is equally difficult to identify and evaluate. Likewise, it is difficult to completely examine the damages done due to spreading of false contents. The reason for this is difficulty and convoluted nature of propagation dynamics of such false news, reviews and information in general. The biggest hurdle in the solution to this problem is its complexity and ambiguity in the patterns of the deceptive text. Natural language processing in itself is a hugely complex task that is not fully understood by humans, as to how the human brain thinks. Moreover in deception detection, principles of philosophy and psychology also get added to it. Human experts are also not successful in identifying false contents given the prerequisite knowledge it takes. Hence all these factors put a major stake on making use of technology and combining human efforts, expertise, knowledge and technological advances in use to generate a robust, accurate, simple and effective solution to the problem of deception detection.

Many researchers describe deception as follows: deception is a conscious intentional effort to manipulate, hide and fabricate information in any way, by verbal or nonverbal means with the purpose of instilling in another person a belief that the producer of said information himself considers to be untrue. In the terms of language literature, deception is similar to lying or misleading behavior.

However in terms of providing fake information, be it fake news or fake reviews, there is an underlying uncertainty in whether the producer or communicator himself is in the knowledge that the material may contain fake information. There is a possibility that the sharing of such non-credible information may not be a deliberate attempt to misguide anyone. In researchers' terminology it may be termed as satire or humor. The information pieces that contain varying amount of false information with deliberate attempt to mis-inform the masses for a certain intention to be fulfilled are generally termed as hoaxes or propagandas.

In this project, we focused on the linguistic styles, writing style, often called stylometric features, semantic features and syntactic features in text contents of the dataset to extract features that uniquely identify or indicate cues that are common to deceptive behavior. We also tried to include features that indicate the behavior, background or patterns of the producer other than the main text to be classified as fake or real, so that any kind of relationship between them could also be identified. Here we consider the definition of anything to be identified as 'fake', to be- information that contains any amount of false contents, created or distributed for the purpose of misleading anyone.

Hence, the problem of deception detection could generally be defined as a binary text classification problem which could be symbolized as a function f(x, m) , that could be defined as taking an input text sample 'x', which is the main text in which deception is to be found. Other input to the function could be combined under variable 'm', which signifies the metadata information accompanying the main text and which could possibly contain deception cues. This deception function gives as output either 0 or 1, where we defined 1 as being a TRUE information piece i.e. containing no deception at all and 0 as being a FAKE information piece i.e. deceptive text.

$$\mathrm{f(x,m)} = \begin{cases} 1 \;, & \text{'x 'and/or 'm' indicate true} \\ 0 \;, & \text{'x' and/or 'm' indicate false} \end{cases} \qquad (1.1)$$

For deception detection in text where metadata is not present or applicable, only the main text is used as the input to the above said function. The main text and the metadata both are in textual form.

1.3. ORGANIZATION OF THESIS

The outline of the dissertation is as follows.

Chapter 2 briefly discusses the previous research and work done related to the field of deception detection to give a quick overview of the literature background for this research solution. Chapter 3 describes the methodology and approach used in the proposed solution, including the features generated and the models used. Chapter 4 depicts the actual parameters of the experimentation done, highlighting all the details of the implemented solution. Finally, chapter 5 discusses used the results obtained due to the experiments done , its conclusions and scope present in the improvement of the proposed solution in the future research directions in this field.

CHAPTER 2

RELATED WORK

This chapter discusses the previous works linked to deception detection, and the diverse mechanisms used for this purpose. The chapter is grouped as follows. Section 2.1 highlights research done for detecting fake contents. Section 2.2 discusses the background research done previously with the aim of deception detection. Section 2.3 summarized and discussed the drawbacks and improvements required in the past solution to these problems.

2.1 LITERATURE REVIEW ON FAKE CONTENTS DETECTION

The need for steering their research towards the problem of opinion spam and the detection of fake reviews were initially identified and discussed by Jindal et al. [7]. They collected and examined 10 million reviews from Amazon to identify fake reviews and considered them to be divided into reviews that are totally fake, reviews that repeatedly target only a single brand and those reviews that just advertise some products i.e. no reviews. For identifying the later 2 categories of 'fake' reviews they employed ML classifiers such as Logistic Regression and Naive Bayes yielding 98.7% accuracy. They extracted 36 features making use of contents of review, reviewer behavior, and metadata information describing product and sales. For determining totally untruthful reviews they labeled all the duplicate and somewhat duplicate reviews as fake and rest of the reviews in the dataset as truth and used logistic regression, decision trees, naïve bayes and support vector machine (SVM) as classifiers for this model obtaining 78% accuracy using all features. They even claimed to discover some information regarding the presence of fake reviews. Some of them are as follows: - High sales statistics of products indicate that they are likely to receive less number of spam reviews, reviews on individual products do not receive much feedback, individual genuine reviewers are less likely to write a large number of reviews; the top reviews and reviewers are most likely to be spam etc.

Most algorithms aimed at detecting opinion spam are divided into 2 main directions. Some models try to extract features from the review itself while other set of models try to gain deception cues from the reviewer behavior.

Models based on contents of reviews-

Most of the solutions proposed in the past using contents of the review to identify whether it is true or not, make use of lexical features, syntactic features, semantic features and similarity features that depict the deception cues in the writing style of the reviews. Lexical and syntactic features are a part of stylometric features.

The most common features used under lexical features are bag of words approach. This defines the database as a collection of words found in all the documents in the dataset and then converts each of those documents according to the approach chosen and the vocabulary formed. The various approaches used to convert each document to a numerical vector are mostly count vectorizer, hash vectorizer, term frequency vectorizer (TF) and term frequency- inverse document frequency vectorizer (TF-IDF). These take into account the count of words in vocabulary, result of hash function on words in the documents, the frequency of 'terms' in document and the frequency of 'terms' normalized by number of documents in dataset, respectively.

Similarly the TF vectorizer on n-grams of documents were used by Ott et al. [8].They collected a "gold standard" data set which consisted of false reviews of hotels from crowd sourcing service of Amazon Mechanical Turk and true reviews from website of Trip Adviser. They trained their model using SVM classifier achieving an accuracy of 84%, while separating the opinions into positive and negative opinions based on their sentiment analysis yielded 86% accuracy. However the accuracy of a human expert to identify fake reviews was 65%.

Mukherjee et al. [9] shed the light upon the real world effectiveness of the model proposed by Ott et al. [8] as the dataset used by them consisted of pseudo fake reviews generated by crowd sourced spammers that may not be similar to the scenario of real world fake reviews. So they decided to train and test their model on the dataset from Yelp website. They also tested the model used by Ott et al on their Yelp data and obtained 67.8%, hence concluding that these methods were not as efficient when tested on real world data. In a later research, Mukherjee et al. [10] observed that Yelp spammers also seem to over-do writing fake reviews by trying too hard for them to appear genuine. Hence not exactly match those fake reviews written by paid online reviewers.

Semantic features are used to identify the semantic similarity among reviews and hence duplicity of reviews. Lau et al. [11] developed a model to classify reviews using un-supervised learning due to unavailability of labeled dataset. Their approach was build on semantic similarity between reviews due to the belief that spammers tend to reuse their words because they don’t want to invest much time and creativity in writing reviews. They collected their data from Amazon and manually labeled them using cosine similarity and human annotations. They proposed a high concept association model, used text mining and applied SVM and Semantic language model (SLM) attaining AUC score of 0.557and 0.998, respectively.

The researchers using syntactic models used to identify fake reviews/contents generally made use of Part of Speech (POS) features, Linguistic Inquiry Word Count (LIWC), n-grams etc. LIWC software was developed to determine psychological and emotional characteristics in text of verbal speech and uses an inbuilt dictionary to classify the text into different categories to which its words belong to, such as ‘money’ belongs to ‘businesses’. These categories contain negative emotion, positive emotion, optimism, verb etc. while POS tagging maps a word to its value according to its meaning and position in the text. For example: sentence is, “Nothing happens to humans once they die.” The POS tagged sentence is, “Nothing/NN; happens/VBZ; to/TO; humans/NNS; once/IN; they/PRP; die/VBP”. Where the categories are: NN= singular noun, VBZ= 3rd person verb, singular present, TO=to go, NNS=plural noun, IN= preposition/subordinating conjunction, VBP= verb, singular present, PRP=personal pronoun.

The language style and content similarity based models using stylometric features were explored by Shojaee et al. [12]. They used the reviews’ dataset gathered by Ott et al. [8] and used the extracted features on SVM and Naïve Bayes (NB) classifiers and obtained 84% accuracy. A model consisting of POS features, bigrams and LIWC features were also used by Ott et al. [8]. They trained it using a naive Bayes and SVM classifiers along with 5 fold cross validation technique, obtaining 89% accuracy with LIWC and bi-grams features.

<u>Model based on Reviewer pattern-</u>

Many researchers take into account the effectiveness of including spammers’ behavior and patterns to identify opinion spam because generally the spammers share same set of characteristics that help to identify them and eventually their fake reviews. The various features extracted for this outcome include the number of reviews by a person for same brand or category, targeting a particular brand or business, the difference of a review’s ratings from average ratings of the product, use of specific type of language or sentiments etc.

Mukherjee et al. [13] implemented a model to detect spammers based on the different behavioral patterns that genuine and fake reviewers possess. They used unsupervised Bayesian inference framework in their model and used posterior density analysis to analyze features in the dataset. Their model also outperformed some of the supervised learning based models. Lim et al. [14] collected data from Amazon to develop a model that included several behavioral characteristics of the spammers such as; they tend to target specific products/brand, giving deviating ratings, attempting to review products early-on to change genuine reviewers' opinions. They combined models for each spammer characteristic into a single spam review detection model and it outperformed the baseline models.

Feng et al. [15] used the ratings anomaly, burstiness and deviation as features for detecting fake reviews on a subset of the gold standard dataset. They obtained an accuracy of 72.5 % by detecting reviews that were fake by observing them during a specific time window. Similarly, Fei et al. [16] proposed a technique that was based on the reviewer burstiness i.e. the short time window during which a product suddenly gets bombarded with reviews. They developed a reviewer network based on their different time windows and used statistical methods and data analysis on it to detect fake reviews and obtained 77.6% accuracy.

Most of the researchers didn't include reviewers that wrote only single or very few reviews as that wouldn't have contributed to the model development. Unlike, Xie et al. [17] who decided to focus on these type of single reviews because they found that most of the reviewers write only single reviews and hence these reviews decide the accuracy of the model. They believed that a sudden increase in number of singleton reviews and sudden change in ratings of store or brand then it may indicate the work of spammers. Their model resulted in accuracy of 75.86%.

Fake news detection is a similar field aimed at identifying fake news which contains fabricated lies for the purpose of deliberately deceiving people by spreading non –truthful information. Though there are several types of contents when describing fake news like propaganda, humor/satire etc. but many researchers stick to the idea of 'serious fabrications' for defining 'fake news'. Like fake reviews, fake news has also gained momentum since the past few years. Fake news are fabricated and spread with just a click and in few moments it already does the damages, sometimes irreparable. Hence it is equally important to tackle fake news problem by using the same technological advances that are used to create and disseminate it.

Rosas et al. [1] observed that there is a shortage of datasets that cover most of the domains of news as well as fit into the generally accepted definition of fake news. Hence they created two datasets through crowd-sourcing and web scrapping that pass all the criteria of a well structured and informative dataset. They build their classifier model using features like n-grams, LIWC features for punctuation counts and psycholinguistic categories, readability metrics and syntactic features. They achieved accuracies of 78% and 70% for web data and crowd sourced data, respectively. Granik et al. [18] used NLP techniques for identifying fake news. They used buzzfeed data and trained a naïve bayes classifier on it and obtained 74% accuracy. Shlok gilda [19] used a dataset accessed from Signal Media and open sources and applied NLP methods on it to extract features such as TF-IDF on n-grams and PCFG. They fed these features and their combination into a number of classifiers and achieved highest accuracy of 77.2% with SGD classifier using TF-IDF and bi-grams as features.

Singhania et al.[20] developed deep learning solution for fake news identification. They implemented a three level hierarchical attention network (3HAN) working on each structure level of texts i.e. word level, sentence level and headlines level and converted them to vectors in a hierarchical manner and giving different weightage to them based on their importance. They used a large real-world data set consisting of news articles from fact checking websites and achieved accuracy of 96.77% on it. Yang et al. [21] also resorted to deep learning to solve the fake news problem. They proposed a CNN for text and image classification that extracts out features based on text like, word/sentence count, punctuation counts etc. and latent features hidden in the images along with the texts. They obtained precision, recall and f1-score of nearly 92% with their model.

2.2.LITERATURE REVIEW ON GENERAL DECEPTION

Ever since the everyday information sharing in both online and offline mediums turned highly injected with deceptive contents, the need for automatically identifying and combating deception and its originators elevated. Therefore recently many researchers have started to focus their work on the general aim of deception detection that identifies anything and everything fake rather than just trying to create algorithms for solving problems in each sub-field of it.

Similarly, Rosas et al. [22] tried to explore the field of deception detection by creating their own dataset that contained lies, truths, education, country and other such demographic information. The dataset consisted of lies and truths that were freely collected through crowd sourcing services of Amazon mechanical Turk. These lies and truths were not based on some targeted topic, but were instead open domain. They also tried to study the relation between lies and truths that people speak with respect to their demographic information. They also performed age and gender detection on the data. The features they used were n-grams, syntactic complexity, readability metrics, shallow and deep syntactic features like POS, PCFG trees and semantic features like LIWC lexicons. They observed that age and gender are related to the language style that people adopt when they fabricate lies. The highest results that they obtained for deception, age and gender detection were 69.5% using POS, 62.26% using readability metrics and 63.04% using unigrams and semantic features, respectively.

Conroy et al. [23] surveyed the recent state-of-the-art Methods used for deception detection. Acknowledging the damage fake information can do due to the overload of information that lacks strict/certain credibility, structure, form and medium, are circulated by its content generators. They discussed various methods that can be used for examining the credibility or veracity using 2 types of approaches i.e. linguistic cue approaches that take into account the data representation, deep syntax and semantic analysis and network analysis approaches that consider importance of linked data and behavior of network users. They observed that a hybrid approach combining these two systems would do a good job in identifying fake contents.

Ott et al.[8] observed that since people are increasingly reviewing and searching products and services online, hence there is definite need to tackle opinion spam or as they call it- deceptive opinion spam which is completely imaginative and with a deliberate attempt to deceive potential consumers. They used the gold-standard opinion spam dataset and applied 3 approaches based on the theories from psychology and computational linguistics to the task of text categorization, psycholinguistic deception detection and genre identification. They obtained accuracies of 73% with POS features, 76.8% with LIWC features and 89.8% with LIWC and bi-grams for each of the task mentioned above, respectively. They contributed that deceptive contents are linked to imaginative writing while truthful contents are linked to informative writing. They noted that for deception detection it is important to consider the context and purpose for deception, rather than trying to construct a universal set of deception cues.

Almela et al.[24] studied the nature of deceptive language in written communication in the Spanish language. They collected a dataset from 100 participants based on three topics and asked people to prepare a written speech on each topic with their true and false opinions on each of these. They build a classifier model on Support Vector Machines (SVM) using features from LIWC 2001 and obtained 73.4 % accuracy when they used all the categories' dimensions as features. Using bag-of words they obtained 64.8% accuracy for all the topics combined. Since usually research in this field focused only on deception in English language, hence their research is a step forward in the direction of research concentrating on other languages.

Fette et al. [25] studied deception detection in the form of phishing where people deceive users into believing that their interaction is with some ‘reliable’ entity who then carry on their malicious activities. This type of user-targeted deception is increasing these days and hence needs a concrete solution. They used a combination of the ham corpora and the phishing dataset to obtain a final dataset containing roughly 6950 non-phishing emails and 860 phishing emails. Some of the features that they used were-Number of URLs, presence of JavaScript, number of domains, tf-idf vectors etc. and achieved accuracy of over 96% while only mis-classifying 0.1% of the legitimate emails.

Ruiter et al. [26] collected their own dataset containing labeled deceptive texts from 700 game scenarios of mafia, in which players don both a deceptive or truthful role and then exchange messages related to it. They used handpicked linguistic features like word count, sentence length etc. and word vectors such obtained from fast text and glove and input combinations of these features to a logistic regression classifier achieving an average precision of 0.39 and an AUROC of 0.68 on 5000+ word documents. Krishnamurthy et al. [27] implemented multimodal deception detection techniques using neural networks. The used the dataset [29] containing real life videos for deception detection, from which features were extracted for different purposes. Features for Visual deception detection were extracted using 3D CNN that identified facial expressions , frame width , height etc. they used word2vec to extract word embedding vectors and then fed them into 3D CNN to extract textual deception features based on the transcripts of the videos. For audio feature detection they used OpenSMILE tool to extract high dimensional features and then converted them to 300 dimensions using neural networks. For obtaining micro-expression features they used already found 39 facial expression features by Rosas[29] for this purpose. They used all of these features in MLP classifier and obtained 0.9799 as the ROC-AUC score and 96.14% accuracy.

Jaiswal et al. [28] also used the real life trail data [29] for the purpose of multi-modal deception detection. They extracted visual and verbal cues for this purpose including facial features like eyebrow raises, blinking of eyes etc. obtained using OpenFace tool and acoustic patterns such as Prosody features, Energy features etc. using OpenSmile. For text classification they extracted lexical features and input the fusion of these features to SVM model and obtained an accuracy of 78.9%.

Rosas et al. [29] collected a dataset consisting of real life trail data of testimony videos from public courts and labeled them as false or true using 3 parameters- exoneration, non-guilty verdict and guilty verdict. They used non-verbal and verbal modes of data to construct a system for automatic deception detection using multimodal data. They extracted verbal features such as LIWC and n-grams and non verbal features indicating patterns of facial features and hand gestures. They were able to achieve an accuracy of 75.2% using all features combined on decision tree classifier and 76.03% accuracy suing facial features on random forest classifier. They observed that their system was able to outperform human experts for this purpose.

2.3. SUMMARIZATION

Opinion spamming has lately been a topic of research that requires immediate and efficient solution due to development of social media and other such forums that allow people to post or share their opinions without their identity and intentions being verified.

The issue of opinion spamming can be solved either by using reviews as features or extracting features from reviewers. One method of detecting fake reviews is to identify linguistic and style related patterns in the reviews that are written by spammers. Natural language processing offers methods to identify such cues through bag of words approach, n-grams, LIWC features and so on. These methods don’t depend on spammers’ behavioral characteristics and hence is more flexible because it needs only the review text to classify it as real or fake.

Spammers' behavioral features can be sentiments, total reviews posted, IP address location, review deviation and their ratings etc. Behavior based models may not be fully efficient for detecting fake reviews that come from individual spammers that may not have a fixed behavioral pattern in the reviews that they write.

However, sometimes spammers' behavior and identity patterns also help to identify fake reviews along with the text that they write. Hence a hybrid of both reviews and their reviewers can be used to detect fake reviews, to address this issue as the need arises.

Fake news detection has become a tough task to solve automatically due to the escalated volume of information that is shared each second without any guarantee of credibility and truthfulness. Malicious sources are spreading fake news for causing damage to reputation, gaining financial benefits, click bait, fame, spreading propaganda and other such devastating intents.

Hence researchers these days are equally motivated to find an automatic solution for combating fake news. Some of these try to create knowledge bases that can act as fact checking sources while others try to include hybrid approach that also includes human expertise. Some of the features used are those that identify the linguistic patterns hidden in the news articles like word count, sentence count, punctuation count etc. apart from lexical and syntactic features like POS tags, PCFG trees, n-grams etc. Some of them use semantic features like LIWC categories, word embeddings etc. Other common methods employed for this purpose are deep learning implemented through neural networks like CNN, RNN etc.

Deception detection has lately gained popularity among researchers due to the ever increasing deceptive data that is present in the public domain in different forms. Many researchers have tried to use linguistic features for this purpose that try to extract out patterns from the language style of deceivers. Some of the features for this purpose were n-grams, POS features, LIWC lexicons, statistical features etc.

Other type of features for deception detection were semantic similarity features that try to find numerical representation of word vectors based on their similarity in a multidimensional vector space and hence extracting hidden deception cues in language structure, context and style of deceivers. Some of the researchers also extract readability metrics as features for finding semantic and syntactic complexity in the data samples.

Other types of features for deception detection in data types other than just textual data are facial expressions, body gestures, acoustic patterns, audio patterns, prosodic features etc. some of the researchers also make use of deep learning using CNN, RNN etc. for identifying deceptive contents and deceivers’ profile, their behavior patterns, motivations, purpose and target behind deception.

CHAPTER 3

MODELS AND APPROACH

In the past few years, deception detection has become an emerging area of research due to hugely increasing number of cases recently that highlight the need to tackle this issue using technological advancements, since technology is also increasingly being used as a tool for creating hard-to-identify deception techniques. This chapter explores the approach used by us and its corresponding model that we applied for the purpose of deception detection. Section 3.1 discusses the natural language processing methods for text classification. Section 3.2 discusses the semantic features used in the model and Section 3.3 outlines the lexical and syntactic features used in the model to detect deception.

3.1. NLP TECHNIQUES FOR TEXT CLASSIFICATION

We have used supervised machine learning models along with NLP techniques to detect fake reviews and other such deceptive contents in text pieces. Supervised machine learning has a predefined training dataset which has a label or outcome that we want our model to learn and based on that attached a label each to the unseen text samples in the test dataset.

The features that we use as a part of natural language processing convert the text samples into a fixed size numerical vectors that are then fed into the ML classifier algorithms for the purpose of text classification.

Majority of the data to be experimented upon exists in the text form, which is in hugely unorganized form; hence it is necessary to be able to extract relevant information from it without changing its imbibed meaning. This is where NLP is beneficial as it consists of methodical processes to extract and derive information from the text by understanding and analyzing it.

The dataset to be used can be either collected from the web source through web crawling or a predefined dataset which suits the classification task can be used. After separating the label, the dataset is divided into the training and testing data. Features are extracted or engineered from the text in the dataset to help predict the label as either fake or real. The learned model is tested using the test set of the dataset or other types of validation techniques can be used for training and testing split of the data.

The process of text classification contains following basic steps:

1. Dataset Preparation: this step involves loading the dataset into the system and performing pre-processing tasks on it so that it is fit for any analysis to be done on it. This dataset is then split into training and testing datasets.
2. Feature Extraction: the preprocessed dataset is then used to extract meaningful features from it or engineer/mine features that aren't directly present in the dataset and hence need to be hand coded.
3. Model Training: This is the last step in the classification process in which a classifier model is trained using the feature vectors converted training dataset along with its labels and then tested on the test dataset.
4. Performance improvement: using different methods for enhancing the efficiency of text classifiers such as parameter tuning, combining features etc.

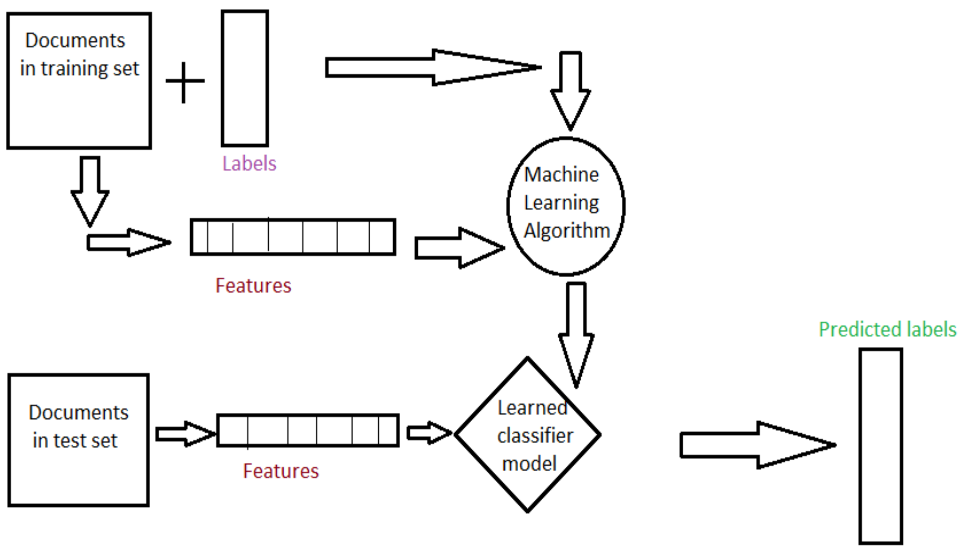


Fig. 3.1: Classification process in machine learning.

<u>Pre-processing of the dataset -</u>

Most dataset contain extra information which is not relevant to the classification process and may even pose a hindrance is correct classification if feature vectors also include such noisy and unrequited information. Any part of raw data which is irrelevant to the context of the program/function and the needed results can be declared as 'noise'. For example – industry/organization particular words, punctuations, social media entities (hash tags, mentions), URLs or links, language stop words (function words of a language like in, of, the, am, is etc.), punctuation removal, tokenization, sentence segmentation, lower/upper casing, and rare/common words removal, spelling correction. The dataset preparation step removes every 'noisy' term found in the text which in turn removes the irrelevant information that is present in the original/unfiltered data thus helping in reducing the amount of actual data that is used as input to the system.

Stop words are those words commonly used in sentences to define its structure and do not hold any special meaning beyond that. Generally considered stop words in the literature of a language are conjunctions, prepositions, articles, and some pronouns. The resulting noise free data is then tokenized i.e. text is separated into tokens which can be words, phrases or sentences. The tokens can be converted into some standard form relative to a language model, after tokenizing the data. This can be done through stemming or lemmatization. Stemming involves converting words or tokens into their standard format by removing the suffices so that the word classes are decreased in number. Lemmatization is similar to stemming but is generally more efficient than the latter since it transforms the word to its root word, instead of removing the suffix by using vocabulary and doing morphological analysis to obtain the root word. For example, words- “Singing,” “sang” and “sung” will be converted to the word “sing”. Stemming/lemmatization further decrease the size of the dataset and make algorithms more efficient. The most commonly used stemming algorithm is porter stemmer because of the results it gives through its accuracy and hence we have also used it.

The difficulty in text processing and classification is to extract meaningful features from the data, while reducing the dimensionality and also not losing its inheriting meaning. We implemented following methods to convert the text columns into numerical feature vectors:

<u>Count vectors-</u> Count Vector is a matrix formation of the data corpus in which every row is representative of a document from the dataset, every column depicts a term from the corpus i.e. the vocabulary prepared by taking each word occurring in the document, and each cell illustrates the frequency count of a specific term in a specific document/text. We have used Count Vectorizer class from scikit-learn module to implement the encoding of text documents into feature vectors.

<u>Term frequency-inverse document frequency (TF-IDF) vectors</u>- the TF-IDF value symbolizes the significance of a term in the document and relative to the entire corpus. TF-IDF score is made up of following 2 terms:
Term Frequency (TF) – it calculates the normalized term frequency in a document.

$$\text{TF (t,j)} = \frac{\text{Number of times term 't' is present in a document 'j'}}{\text{Total number of terms in the document 'j'}} \qquad (3.1)$$

Inverse Document Frequency (IDF) - it calculates the logarithm of, the total number of documents in the entire corpus divided by the number of documents where the specific term is present. Its main feature is that it counteracts or weighs down the frequency of the term meanwhile making it up for the rarely occurring terms.

$$\text{IDF (t)} = \log_e \left(\frac{\text{Total number of documents}}{\text{Number of documents with term 't' in it}}\right) \qquad (3.2)$$

The final TF-IDF score, S (t,j) of a term is the product of its corresponding TF value and IDF value.

$$\mathrm{S(t,j)} = \mathrm{TF(t,j)} * \mathrm{IDF(t)} \qquad (3.3)$$

TF-IDF Vectors can be generated at word level tf-idf scores where each word is taken into account , or n-gram level tf-idf scores where combination of n words together act as a 'term' in the vocabulary , or character level tf-idf scores where combination of n characters together is treated a 'term'. TfidfVectorizer and TfidfTransformer from scikit-learn class are used for implementing the TF-IDF vectors/features.

For example, if there is a document with 400 words and 1000 such documents and if we want the TF-IDF score for the word "apple" which exists in the document 8 times then TF = 8/400 = 0.02. And if it appears in all the documents 370 times then IDF (apple) = $\log_e$ (1000/370) = 0.994. Then TF-IDF (apple) = 0.02 × 0.994 = 0.01988.

Hash vectors- it calculates the value of a one way hash function of words to convert them to integers. No vocabulary is required and an arbitrary-long fixed length vectors are possible, hence it is efficient and space saving. The HashingVectorizer class implements this approach that can be used to consistently hash words and hence encode documents/texts as needed.

N-gram is a contiguous sequence of length ‘n’ composed of terms from the text documents. It could be a made of bytes, characters, words, numerical etc. Unigrams are the terms of length=1, and if it is calculated on word level, then they are just the unique words in the documents. Similarly there are bi-grams; consisting of length=2, tri-grams; consisting of length=3 and so on. For example, the word level n-grams for the following sentence are:
“Ram and Shyam cycle together in the park.”
Uni-grams are- Ram, and, Shyam, cycle, together, in, the, park.
Bi-grams are- Ram and, and Shyam, Shyam cycle, cycle together, together in, in the, the park.
Tri-grams are- Ram and Shyam, and Shyam cycle, Shyam cycle together, cycle together in, together in the, in the park.

The unigrams are actually just the individual words that appear in the dataset and hence are similar to the “bag of words” technique used to convert the text documents into numerical features. The bag of words system doesn’t take into consideration the order of the text samples in the dataset, unlike any other higher order n-gram model, hence unigrams do not consist as much information as its higher order counterparts. The n-gram model is a primitive and efficient model for classification of text and also its categorization. N-grams are based on the basic principle of capturing the language structure/order. The higher the value of n, the longer the n-grams and more the context of the documents are present in the features. Optimum length of n really depends on the application; small n may not be able to catch the context/differences, while larger n may become less general and a little too specific for classification. Moreover it is immune to typographical errors and uneven distributions of the words since it considers only the structure of the phrases/words.

In this project, we had used the word and character-based n-gram model to signify the context of the samples in the corpus and generate numerical features for text classification. The n-grams thus extracted are converted into numerical feature vectors by means of count vectorizer, TF-IDF vectorizer and hash vectorizer. These n-grams are also used in combination with other features like POS tagged features, engineered features etc.

Classification-

Generally the ratio of training data to validation/test data is 4:1 i.e. the test data is 20% while the training data is 80%. Suppose there are ‘m’ documents in the training set and ‘n’ documents in the training set and if the number of features in the feature matrix are restricted to ‘k’, then the resulting feature matrix for training set will be

$$T=\begin{bmatrix} t_{00} & \cdots & t_{0k} \\ \vdots & \ddots & \vdots \\ t_{m0} & \cdots & t_{mk} \end{bmatrix} \qquad (3.4)$$

The resulting feature matrix for test dataset will be

$$S=\begin{bmatrix} s_{00} & \cdots & s_{0k} \\ \vdots & \ddots & \vdots \\ s_{no} & \cdots & s_{nk} \end{bmatrix} \qquad (3.5)$$

The corresponding values in the feature matrix will vary depending on the scheme that is applied to convert the text documents into feature vectors and if the terms don’t exists at all in the vocabulary obtained then its corresponding value in the matrix would be null or 0.

For combining many features together or to include several information columns as features, the feature matrix from different schemes can be combined to obtain a single feature matrix, each for the training and testing data and then fed into the classifier.

The ultimate step in the classification process is to train the classifier through the features extracted in the preceding step. These features represent the texts in the corpus and in a way their individual characteristics and the class that they belong to and hence making the classifier learn that which features are prevalent in which class and hence signify its presence.

We used many different classifiers for the purpose of deception detection through text classification, which are, naïve bayes classifier, passive aggressive classifier, random forest classifier, MLP (multi-layered perceptron) classifier, Stochastic Gradient Descent, Support Vector Machines (SVM), Ada boost classifier, K-Nearest Neighbor (KNN), linear classifier (Logistic Regression), gradient boosting classifier and voting ensemble classifier.

3.2. SEMANTIC FEATURES

3.2.1. WORD EMBEDDINGS AS FEATURES

Word Embedding is the representation of text through the form of numerical vectors. It is based on the idea that 'similar' words will have a lesser distance between their vectors and hence words which are semantically similar can be found and used as features. There may be various numerical representations for the same text based on the method that is used to convert it into numbers. Since most deep learning architectures and various machine learning algorithms are unable to process strings of plain text in their raw form, hence they need numbers as inputs to build any application for the purpose of natural language processing. A Word Embedding model usually tries to map a word/term to a vector using a dictionary which is either pre-trained or trained on the original text corpus.

For example, a sentence is – “Ram and Shyam cycle together in the park.” If suppose the dictionary lists all the unique words in this sentence, then it would look like – [‘Ram’, ‘and’, ‘Shyam’, ‘cycle’, ‘together ’, ‘in’, ‘the’, ‘park’].

If the vector representation of this sentence is done using one-hot encoder, where 1 signifies presence and 0 signifies absence, then the corresponding vector for the word ‘cycle’ in this sentence is- [0,0,0,1,0,0,0,0].

Word Embedding is a type of dense features in low dimensional fixed-size vector space. Hence it develops better features for most of NLP problem. The different types of word embeddings can generally be classified into 2 categories- Prediction based embedding and Frequency based embedding.

There are generally 3 types of vectorization methods that are used in frequency based embedding, namely, Co-Occurrence Vector, TF-IDF Vector and Count Vector. Out of these, we make use of only count vectors and TF-IDF vectors. The prediction based embeddings are generally of 2 types, namely, skip gram model and continuous bag-of-words model (CBOW).

CBOW model- it is a shallow neural network which maps words to target words and learns weights that acts as vector representation. When provided a context, it tries to predict the probability of a word. The context can be just a word or collection of words. The matrix corresponding to a sentence is fed into a shallow neural network with 3 layers, an input layer, a hidden layer and an output layer which performs a softmax function used to sum the obtained probabilities to 1. Advantages of CBOW are -Since it uses probabilities, it generally performs better and also it uses way less memory compared to other such sophisticated means of word embeddings. Disadvantages of CBOW are - It computes the average of the contexts related to a word and uses it as the final input context, which may not always be helpful and sometimes it requires very large amount of time for training.

Skip gram model- Skip-gram model is also based on a shallow neural network same as CBOW but the difference is just in its architecture, which is same up to the hidden layer. It is actually the opposite of CBOW because given a word it predicts the contexts. However the target words/variables are different for the skip gram model which contains two outputs. The vector representation is the weights' vector between the input and hidden layer. The objective function is also same as used in the CBOW model.

Advantages of Skip-Gram Model are- it has 2 encoded target variables for a single word and hence can use a larger context i.e. two semantics for one word and also when used along with negative sub-sampling, it generally performs better than other methods for the same purpose.

Word embeddings can be used for a lot of functions related to the contextual similarity between terms. Such as :- Finding the word different than every other word in the text; Finding the amount of semantic similarity between two words; Developing equations depicting the semantic conclusions in the sentences; using the model for translation and other tasks of natural language processing and Finding probability of a text using the developed model. Word embedding models require a lot of text for it to perform better and learn using a lot of vocabulary, so we can use the word vectors developed and pre-trained by Wiki, Google etc. or we can train it on our training data.

In 2013, Tomas Mikolov et al. popularized Word Embedding which became the state-of-the-art for applications based on NLP. He developed and released the Word2Vec toolkit. Other types of word embeddings include the GloVe released by Stanford, the fastText (extension of Word2Vec) released by Facebook and Wordnet introduced by George A. Miller. All of these are databases offering many languages that depict the lexical and semantic relations between the words and its grammatical characteristics. Using Gensim tool of NLTK these are popularly used as pre-trained models for word embeddings, also called universal embeddings. This is a form of transfer learning because the learning obtained in some different environment is applied on related problems.

However since the word embedding models released by these researchers and companies are trained on general data based on Google news etc. These may not always be suitable for representing domain specific texts or those that do not match with the pre-trained model completely, or for other reasons such as not being freely available and time and space complexity which can be an issue with limited resources. Hence, word embeddings can be developed to suit the application and trained on the problem specific data to get better results.

In this project, we have used embedding layer which is a part of the keras package to develop word embeddings trained on the problem specific data. It requires the corpus to be pre-processed and encoded to integers using one-hot encoder and mapped to word vectors before feeding it into the model. The feature vectors/embedding layer are of fixed dimensions and initially contain random real numbers. This can sometimes be slow, while requiring large data, but will result in embeddings targeted for the specific problem and hence performing better.

The Embedding layer is the first hidden layer of the neural network. It takes 3 arguments, namely, 'input_dim'- size of the vocabulary of the data i.e. the number of different integers used for encoding; 'output_dim'- the size of the vector space for the embedded words; 'input_length'- size/length of input sequences/documents. We have used a dense layer after the Embedding layer hence flattened the 2D vector output of the latter to a 1D vector using the Flatten layer. Then the model is fitted on the training data and uses binary cross entropy loss function along with Adam (stochastic gradient descent) as the optimizer. After this the model is evaluated on the test dataset to see its performance and accuracy.

3.2.2. EMPATH TOOL FOR TEXT ANAYSIS

Word embeddings were used for the purpose of calculating the word similarity between terms and hence using the semantic similarity as features. Similarly, for making use of the semantic features as input to classifier model, we had used Empath for generating semantic categories for words similar to Linguistic Inquiry Word Count (LIWC) tool. LIWC(latest update LIWC2015) is a software tool developed to categorize psychological, emotional parts of speech, cognitive analysis and semantic characteristics in text using a built-in dictionary available for a specific language and providing a percentage of total words in the text that belong to those categories, for example, ‘money’ belongs to category ‘business’. A word can be part of different categories at the same time. But since it is not freely available and also not appropriate if there is lack of resources, we used its similar text analysis tool, called Empath[30], released by Stanford in 2016 and authored by Ethan Fast, which is lightweight and easy to use, install and also freely available.

Empath is an automatic tool for analyzing text across several categories based on its semantic meaning using 200 built-in and human validated categories that are highly correlated to those in LIWC but much more than them and also updated categories as it is mined on latest data from the web, existing knowledge bases and fiction literature. Some of the categories it analyzes on are: social media, hipster, violence, swimming and many more. It can be used to generate user defined categories based on the input seed words or validate new categories. These categories can be based on any of the following models-fiction, NYtimes and reddit. It makes use of deep learning, skip-gram model, word embeddings and crowd sourcing. By default it returns the raw counts of occurrences of categories but it can also be normalized over all the words in the document. Empath is available as a web service at [31], to analyze text and categories it holds while we used it as an open source python library available at [32].

In our project, we made use of Empath’s analyze function to generate categories for each of our documents and then hand coded these categories as feature matrix each for the training data and test data separately. Then we fed them into classifier for prediction of the classes. We also combined these features with TF scores on the documents and then fed them into classifier for deception detection. We had also generated hand coded categories of lies and truths based on the frequency or popularity of their occurrence and the actual ‘true’ or ‘false’ label that the documents held and then used them in a formula to predict the label of unseen document based on the probability of it being deceptive or not.

The total dictionary representing count of occurrences of the categories for each text sample in the training dataset which is classified as a lie, can be denoted by,

$$\mathbf{L} = \sum_{\boldsymbol{x}=\mathbf{1}}^{\boldsymbol{n}} \mathbf{R}_{\mathrm{x}} \qquad (3.6)$$

Similarly, the total dictionary representing count of occurrences of the categories for each text sample in the training data which belongs to the category ‘truth’, can be denoted by,

$$\boldsymbol{T} = \sum_{\boldsymbol{a}=\mathbf{1}}^{\boldsymbol{m}} \boldsymbol{S}_{\mathrm{a}} \qquad (3.7)$$

Where ‘R’ and ‘S’ are the dictionaries returned by the analyzer function of Empath for the input text sample. They can be seen as vectors or matrix where each position describes the category and its corresponding value describes the count of occurrences of that category present in the text. The variables ‘x’ and ‘a’ are the counts of text samples (or their corresponding categories' dictionaries returned by Empath) in the dataset that correspond to lie and truth category, respectively.

The final dictionaries corresponding to lie class and truth class are ‘L’ and ‘T’, respectively. They contain the categories and their occurrence count summed over each dictionary belonging to lie class and truth class.

These final dictionaries are then converted into feature vectors form where the values are the counts corresponding to the categories. Hence these represent the weightage each category holds in lie class and truth class, respectively. These dictionaries are then used to calculate the final weightage of a category in the formula for predicting the class label or a text with respect to lie or truth class, by dividing the count (value) of that category in lie or truth dictionary by the sum of values of that category in both truth and lie dictionary.

$$\boldsymbol{W}_{o} = \frac{\boldsymbol{L}_{o}}{\boldsymbol{L}_{o}+\boldsymbol{T}_{o}} \qquad (3.8)$$

Where ‘W’ is the dictionary representing the final weights of a category w.r.t the lie class and ‘o’ is the variable representing the category for which the weight or value is calculated. The value of ‘o’ ranges from 1 to k where ‘k’ is the total number of categories returned by Empath. The weights wr.t truth class can be obtained by subtracting 1 from $W_{o.}$

This dictionary containing the weights of categories is used in the final formula to compute the probability of a text belonging to lie class or truth class. We then take each data sample from the test dataset and obtain its corresponding Empath categories and their values or occurrences. These values are then multiplied to their corresponding weights obtained earlier. The resulting values obtained after that are summed to calculate a final value ‘P_l’ and ‘P_t’, which represent the probability of a text sample being true or false.

$$\mathbf{P_l} = \sum_{\mathbf{o=1}}^{\boldsymbol{k}} \boldsymbol{W}_{o} * \boldsymbol{Z}_{o} \qquad (3.9)$$

$$\mathbf{P_t} = \sum_{\mathbf{o=1}}^{\boldsymbol{k}} (\mathbf{1} - \boldsymbol{W}_{o}) * \boldsymbol{Z}_{o} \qquad (3.10)$$

Where ‘Z’ is the dictionary of categories and their values obtained from Empath when a text sample is input to it. The class of the text sample is labelled as being ‘truth’ if $P_t > P_l$. Otherwise the class is labelled as ‘lie’. The accuracy of the method employed above is then calculated as usual by checking the actual label against the predicted or assigned label. The purpose of using text analysis tool for deception detection is to understand the language characteristics, structure and emotions used by liars, which had often been proved to be an efficient feature for classifying anything fake.

3.3. LEXICAL AND SYNTACTIC FEATURES

Fake texts have a common characteristic that they are consciously created with the intention of deception and hence carry the linguistic style that liars commonly have in order to pass of their lies as truths. Hence it is efficient to extract features for detecting fake content through linguistic patterns that exhibit the different writing styles. Linguistic features can be extracted from different levels of document organizations such as characters, sentences, words and documents texts itself. The common linguistic features used generally are:

- Lexical features- these include the word level and character level features, such as, characters per word, total words, unique words, frequency of large words, word count, word density etc.
- Syntactic features- these include the phrase or text level features, such as punctuation and parts of- speech (POS) tagging, frequency of function words and phrases like - bag-of-words, n-grams etc., topic modeling etc.

Moreover, many other features can be constructed explicitly to capture the deceptive cues in writing patterns and styles to differentiate fake content from the real one, such as lying detection features.

For making use of lexical features we extracted numerical features such as:
Total number of words in the document-word count; average length of the words used in the documents- Word Density; total number of punctuation symbols in the documents- Punctuation Count; total number of characters in the documents- Character Count; total number of uppercase words used in the documents- uppercase; and many more such engineered features based on the information present in the dataset, either through the main texts or the metadata. We had used combination of these features in different ways, sometimes along with textual data and sometimes along with other such numerical features to generate feature union of these cues and then fed into a classifier model.

We had used pipeline and feature union functions from the Feature Extraction and feature Selection library of python to combine various linguistic features and form a classifier model that identifies deceptive cues for the purpose of detecting unreal contents. Sometimes we have to process the data in such ways as to remove information that may not be relevant to the idea of classification. Hence, we used some basic processing on the text such as removing punctuation, rare words removal, common words removal and then tokenizing it and performing stemming or lemmatization along with count vectorization and tf-idf transformer to generate features. We had also combined various textual information given in the metadata along with the main text to be classified, and then converted them into numerical features , or combined with engineered features to generate a resulting single features that carries the characteristic information deemed helpful in predicting the class of the documents.

For the purpose of extracting syntactic features we used the bag-of-words and the n-gram techniques along with the count vectorizer, TF-IDF vectorizer and the TF scores of documents as features. The TF score of a text is calculated as part of the total TF-IDF score and depicts the normalized frequency of a term in the document. We had used the above vectorization method along with the n-gram approach. If the vectorization is done on word level n-grams then combination of n words together act as 1 feature while if it is done for character level then combination of n characters together act as a feature. Example, if the sentence is "words hold meaning." Then character level n-grams, where n=4, are:['word', 'ords', 'rdsh' , 'dsho', 'shol', 'hold', 'oldm' and so on..]. Parts of speech tagging, often called POS tagging is a form of syntactic features that depict the grammatical structure of documents or more precisely, the parts of text are tagged with the grammatical category that they belong to. The POS function checks each document and returns the category that each word/term belongs to, in a language structure.

Example, nouns have following subcategories-'NN', 'NNS', 'NNP', 'NNPS'; Adverbs are-'RB', 'RBR', 'RBS', 'WRB'; Adjectives are-'JJ', 'JJR', 'JJS'; verbs are-'VB', 'VBD', 'VBG', 'VBN', 'VBP', 'VBZ' and pronouns are-'PRP', 'PRP$', 'WP', 'WP$' . The frequency count of categories present in POS tagged documents can be used as features. The POS tagged documents are then engineered into features to make them suitable to be fed into a classifier model. We had also combined the POS features along with the lexical features like TF vectorizer, word density etc. to obtain a single feature vector that combines these cues.

Topic modeling is another method to classify documents based on the topics or keywords that they have that belong to certain categories. TF-IDF vectorizers can also be seen as a way of generating topics from the documents. Here topics mean the words that represent the documents. These topic distributions over documents can be useful to generate distinguishing features in the corpus.

We had used LDA (Latent Dirichlet Allocation) and LSA (Latent Semantic Analysis) for this purpose. LDA is used to generate topic modeling features for fixed number of topics or classes that the documents contain. It is based on the theory that each topic is made up of some keywords and each document is made up of topics based on the probability of their occurrence. It calculates 2 matrix one for document-topic distribution and another for topic word distribution. It iterates through each word for each document and updates its topic- word probability using a product of 2 probabilities each from both the matrices and converges when a steady state is achieved.

LSA is also an unsupervised learning algorithm like LDA, which is used generally for dimensionality reduction or selecting the best features to represent a document. It is used to convert documents into features based on the topics or groups of text that they possess. The first step in it is to convert documents into their corresponding TF-IDF vectors and then select the better features out of them using the SVD implementation of LSA. In it the words that are similar and within a topic can be given same weight and hence no need to include every single word in the vocabulary and increase the size of vectors, with no improvement in feature quality.

CHAPTER 4

EXPERIMENT RESULTS AND EVALUATION

We explain in this chapter the experiments that we carried out by the implementation of our proposed approach and the evaluation of the results obtained thereafter. We have outlined in this chapter the purpose and results of using the different features, their combination along with other types of features and predictions from different features. We examine the effectiveness of using various categories of features such as semantic features, linguistic features etc.
We also evaluate the structure of datasets used along with the nature of data and its impact on the overall classification process. We tried to conclude the results obtained using various models used for deception detection and their role in driving the classification process.

4.1. DATASETS

4.1.1. DATASET 1 OPEN-DOMAIN DECEPTION DATASET

This dataset was released on August 27, 2015 by Ver´onica P´erez-Rosas and Rada Mihalcea, researchers from University of Michigan [22]. This is a crowd sourced deception dataset containing short one sentence truths and lies collected from 512 users or workers from Amazon Mechanical Turk. They provided 7 lies and 7 truths each by every user. These lies and truths are open domain i.e. they don't belong to any specific category or topic but are just plain lies and truth. The dataset also consists of user's demographic information, such as education level, country of origin, age and gender. Hence it can be used to study the effect of this demographic information on the type of lies and truth that people generally tell. Verification was manually done by the paper authors to avoid spam. The dataset consists of 7168 sentences from 512 unique contributors. The dataset consists of total 3584 truths and 3584 lies.

4.1.2. DATASET 2 REAL LIFE TRIAL DATASET

This dataset was released on June 15, 2016 by Veronica Perez-Rosas, Mohamed Abouelenien, Rada Mihalcea, Mihai Burzo, researchers from University of Michigan [29]. It is a real-life multimodal dataset containing truthful and deceptive testimonies during the court proceedings, which are transcript and annotated manually by the researchers. The dataset consists of total 121 short videos, their transcriptions and facial and body language gesture annotations for the purpose of deception detection, taking into account the verbal and non-verbal cues related to deceptive behavior. It contains 60 truthful texts and 61 are deceptive/false written speeches. This dataset can be used for the purpose of studying human behavior and patterns during false speech through multiple modes of deceptive communication and detecting such fake information and cues that identify it. However we have used only the text transcripts for the purpose of deception detection.

4.1.3. DATASET 3 AMAZON PRODUCT REVIEWS DATASET

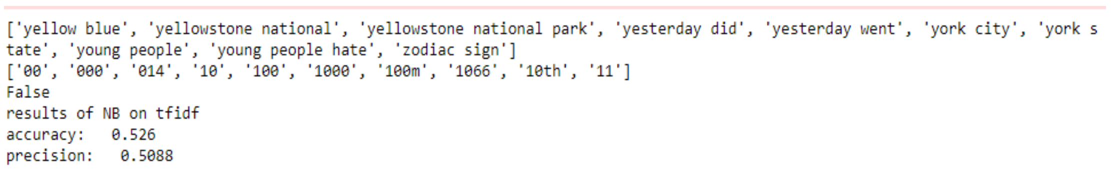
This is a recent dataset of Amazon product reviews provided by kaggle [33]. The reviews are labeled as fake or real corresponding to _label1_ and _label2_ respectively in the dataset. The dataset contains Amazon product reviews across different product categories like kindle, books, electronics etc. along with other metadata information like product category, rating, product ID, user ID, product title, verified purchase, review title etc. The corpus contains a total of of 21,000 text reviews, which are equally distributed across fake and real reviews and product categories. The dataset can be used to identify fake reviews using the reviews itself and the background information available like ratings, sentiments etc. that showcase a certain reviewer behavior/pattern.

Table 4.1: Description of the datasets used in experimentation.

Dataset	**Data contained**	**Content**	**Purpose**
Dataset 1	Lies and truths	3584 lies and 3584 truths	Lie detection
Dataset 2	Trail testimonies	60 truthful and 61 false testimonies	Deception detection
Dataset 3	Product reviews	21000 reviews equally divided among fake and real reviews	Fake information/ reviews detection

4.2. EXPERIMENTS ON OPEN-DOMAIN DECEPTION DATA

We had used Dataset 1, which contains equal amount of lies and truths obtained through crowd sourcing services. First we parsed the dataset that contained a lot of special characters/noise using pandas library. The successfully parsed data was then converted into feature vector using count vectorizer, TF-IDF vectorizer and hash vectorizer. Those resulting feature vectors were then fed into classifier such as multinomial naïve bayes, SVM, stochastic gradient descent, passive aggressive classifier etc.



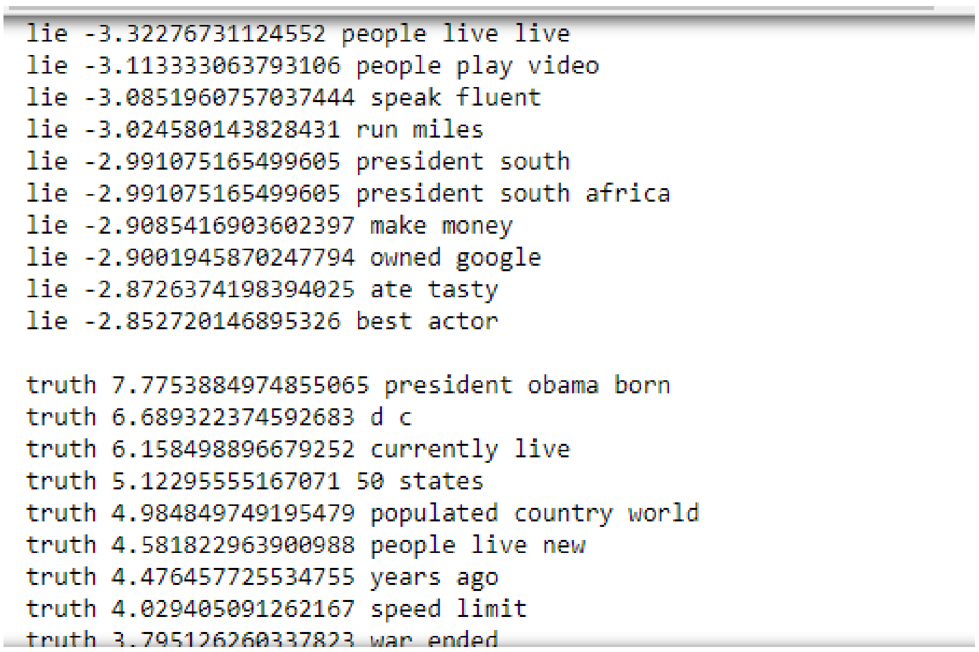
```
['yellow blue', 'yellowstone national', 'yellowstone national park', 'yesterday did', 'yesterday went', 'york city', 'york s
tate', 'young people', 'young people hate', 'zodiac sign']
['00', '000', '014', '10', '100', '1000', '100m', '1066', '10th', '11']
False
results of NB on tfidf
accuracy:   0.526
precision:   0.5088
```

Fig 4.1 Features extracted by count vectorizer and TF-IDF vectorizer, respectively.

We had also used feature union function from feature extraction and feature selection modules to combine more than one feature and classifiers in a pipeline. One set of pipeline included count vectorizer and TF-IDF vectorizer while another set of pipeline made use of chi square function for selecting best features from the count feature vectors and feeding them into classifier yielding 53.26% accuracy. We also included in another pipeline, a snowball stemmer to perform stemming on texts before vectorizing them using count vectors and then modifying them using TF-IDF transformer to obtain final feature vectors which resulted in 54.24% accuracy with Naive bayes classifier.

We made use of n-grams as features and vectroized those using count vectorizer, TF-IDF vectorizer and TF scores. We extracted unigrams, bi-grams, tri-grams and quad-grams as features. However not much difference was seen in the accuracy of the classifiers when using quad-grams. The result of using TF-IDF features with n-grams on MLP classifier was 51.61% accuracy while using TF-IDF features alone on KNN classifier it obtained 51.42% accuracy.

We also extracted the features/terms that included the most weightage among all the terms for each of the lie and truth class to determine what terms are commonly used in the lies and truth and what effect they have on classifying a text as lie or truth respectively. We found that the lies often contain the terms related to politics, money, luxury or strength while truthful texts contain more realistic terms that often convey happiness or normal life in general without making an extremely strong statement that doesn’t match with a common known fact. This holds true in coherence with the previous studies that dictate that liars tend to over exaggerate their emotions along with using less concrete language in order to pass off their opinion as truthful. However truthful texts contain normal language with details and focus on concrete concepts and not just function words.



```
lie -3.32276731124552 people live live
lie -3.113333063793106 people play video
lie -3.0851960757037444 speak fluent
lie -3.024580143828431 run miles
lie -2.991075165499605 president south
lie -2.991075165499605 president south africa
lie -2.9085416903602397 make money
lie -2.9001945870247794 owned google
lie -2.8726374198394025 ate tasty
lie -2.852720146895326 best actor

truth 7.7753884974855065 president obama born
truth 6.689322374592683 d c
truth 6.158498896679252 currently live
truth 5.12295555167071 50 states
truth 4.984849749195479 populated country world
truth 4.581822963900988 people live new
truth 4.476457725534755 years ago
truth 4.029405091262167 speed limit
truth 3.795126260337823 war ended
```

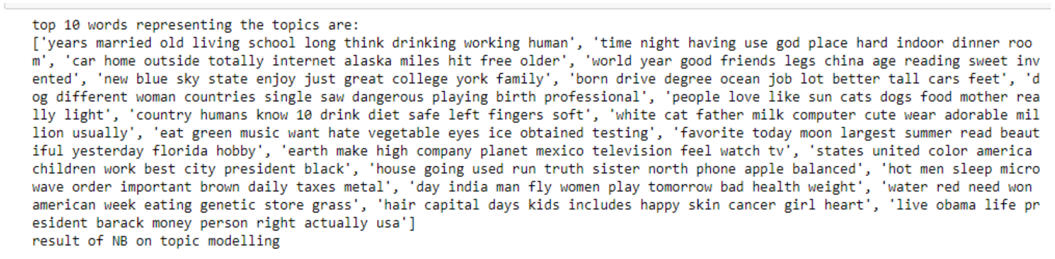
Fig 4.2 Most informative words/features in lie and truth class.

Next we extracted numerical features from the text samples. These engineered features were used as an additional effort to gain insight into the hidden feature space present in the Meta information present in the dataset. The numerical features engineered for this purpose are:

- Word count in each text, depicted by column-‘word_count’
- Average word length in each text, depicted by column-‘avg_word’
- Character count in each text, depicted by column-‘char_count’
- Punctuation count in each text, depicted by column-‘punctuation_count’
- Word density in each text, depicted by column-‘word_density’

These numerical features were used in combination with text and then converted into features together or converted into features separately and then combined as one single feature vector to be fed into the system which resulted in accuracy of around 54.68%. We had also used these features individually as an input to the classifier, based on the understanding through research that the word count, punctuation count, word density etc. reveal deceptive behaviour because liars tend to make their text too long in order to convince their point of view as a fact but tend to use less vocabulary and often repeat their words that aren’t too long in length either. We had also used the main text as feature along with other information present in the dataset like country, education etc.

We used LDA for gathering features based on the topics and keywords used in the main text. The top words used in the topics were also displayed to identify the topics that people lie or tell the truth about since the dataset was open domain and hence lacking any well defined features. However the features based on this function didn’t prove to be much helpful in increasing the accuracy of the model which settled at 53.66% with SGD classifier. We had also used LSA’s implementation i.e. SVD for generating features to be fed into the classifier. This function selects better features and discards the one that don’t hold enough wightage in the classification process, hence improving the accuracy and conserving the time and space resources.

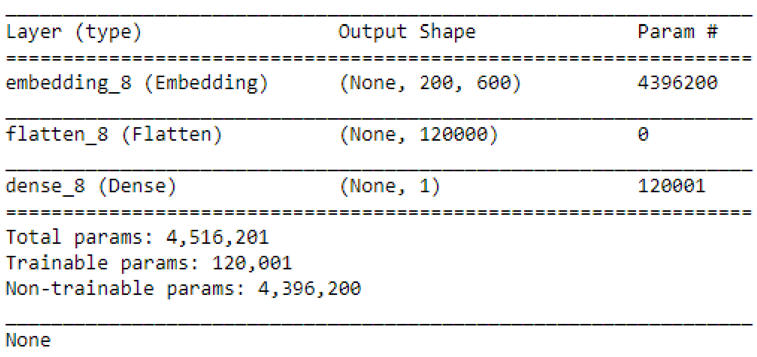


```
top 10 words representing the topics are:
['years married old living school long think drinking working human', 'time night having use god place hard indoor dinner roo
m', 'car home outside totally internet alaska miles hit free older', 'world year good friends legs china age reading sweet inv
ented', 'new blue sky state enjoy just great college york family', 'born drive degree ocean job lot better tall cars feet', 'd
og different woman countries single saw dangerous playing birth professional', 'people love like sun cats dogs food mother rea
lly light', 'country humans know 10 drink diet safe left fingers soft', 'white cat father milk computer cute wear adorable mil
lion usually', 'eat green music want hate vegetable eyes ice obtained testing', 'favorite today moon largest summer read beaut
iful yesterday florida hobby', 'earth make high company planet mexico television feel watch tv', 'states united color america
children work best city president black', 'house going used run truth sister north phone apple balanced', 'hot men sleep micro
wave order important brown daily taxes metal', 'day india man fly women play tomorrow bad health weight', 'water red need won
american week eating genetic store grass', 'hair capital days kids includes happy skin cancer girl heart', 'live obama life pr
esident barack money person right actually usa']
result of NB on topic modelling
```

Fig. 4.3 Top weightage words present in the generated topics.

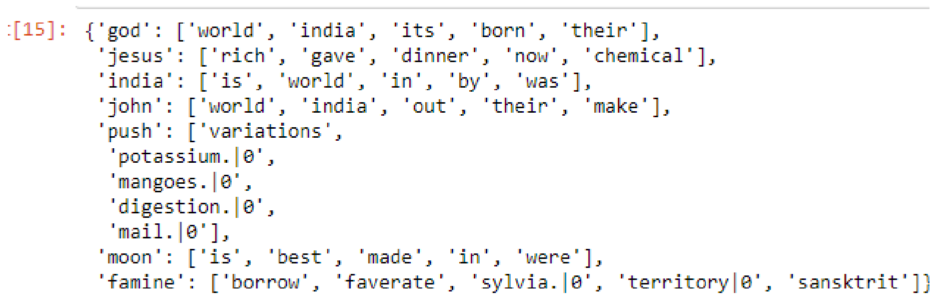
We used white space tokenizer followed by porter stemmer for processing the data and then vectorizer that to obtain numerical feature vectors. We performed punctuation symbols’ removal, frequent words removal, stemming and lemmatization as part of processing the data and hence improving the data in order to obtain better results after vectorization.

We had used word embeddings as way for converting our text data into numerical features according to their semantic meanings and similarities between words and phrases. The word embeddings were implemented using keras library of the python. The text samples were tokenized, then encoded and padded. The class labels were separately encoded using label encoder and one-hot encoder. The keras model was built using a sequential layer, an embedding layer followed by a flatten layer and a dense layer that implemented the sigmoid function as the activation function. The keras model was then compiled using 'binary cross entropy' as the loss function and 'adam' as the optimizer. The results obtained by using word embeddings as features were really good compared to other features. We has also used the Gensim model for understanding how word embeddings work and the features that they offer by converting words into vectors depicting their meaning or context in a multidimensional space.



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_8 (Embedding)      (None, 200, 600)          4396200
_________________________________________________________________
flatten_8 (Flatten)          (None, 120000)            0
_________________________________________________________________
dense_8 (Dense)              (None, 1)                 120001
=================================================================
Total params: 4,516,201
Trainable params: 120,001
Non-trainable params: 4,396,200
_________________________________________________________________
None
```

Fig. 4.4 Structure of word embedding through keras.



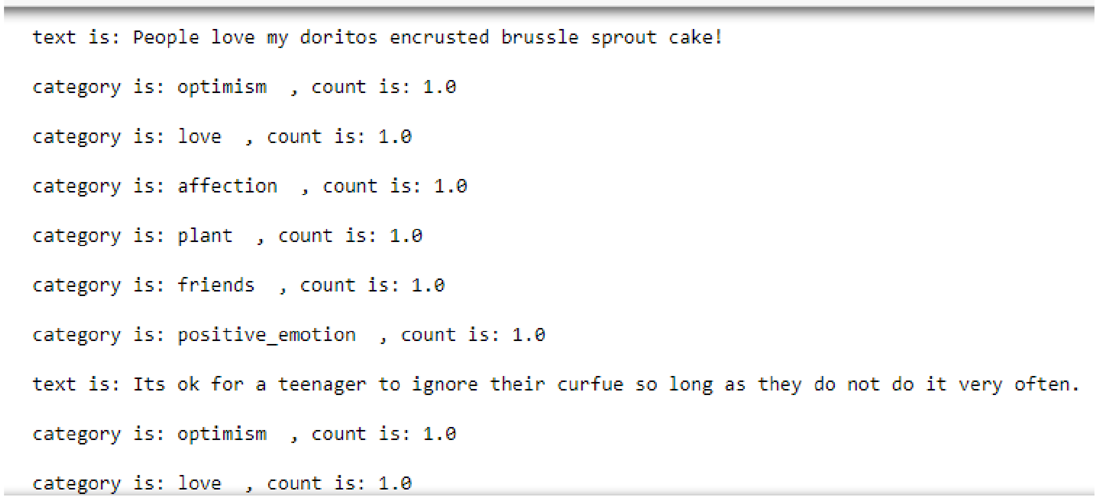
```
:[15]: {'god': ['world', 'india', 'its', 'born', 'their'],
         'jesus': ['rich', 'gave', 'dinner', 'now', 'chemical'],
         'india': ['is', 'world', 'in', 'by', 'was'],
         'john': ['world', 'india', 'out', 'their', 'make'],
         'push': ['variations',
          'potassium.|0',
          'mangoes.|0',
          'digestion.|0',
          'mail.|0'],
         'moon': ['is', 'best', 'made', 'in', 'were'],
         'famine': ['borrow', 'faverate', 'sylvia.|0', 'territory|0', 'sansktrit']}
```

Fig 4.5 Result of using Gensim model to calculate 'similar' words from text for given words.

We used the k-fold cross validation on the classification done using n-grams as features. We used voting ensemble of classifier for combining the best classifier models that we developed and this turned out to yield good results with 60.48% accuracy. We also used Ada boosting classifiers and gradient boosting classifiers as a form of ensemble of classifier and hence combining various classification results.

We had used POS tagged texts as features in order to include the sentence and language structure of the data samples as features since this is an efficient way to understand the linguistic style of deceptive communication and hence find cues for identifying it. The POS features were combined along with n-grams obtained and modifies using TF scores to obtain a single combined feature. We also combined these features along with the engineered features to obtain another hybrid feature space as a final feature column to be fed into the model.

We had used Empath's analyze function to generate semantic categories for words in the data sample. These categories and their count acted as features or numerical representation of the corresponding texts. We hand coded all these categories and their corresponding counts as feature vectors and then combined them to produce feature vectors for training and testing data separately. These features were used in a classifier model for classifying the data and obtained 54.63% accuracy when fed to Naive bayes classifier. We also combined these feature vectors along with the TF scores of the text to obtain final vectors for feeding into the classifier. We also had hardcoded some lie/deception categories by studying the common categories found in 'lie' samples and then used a threshold presence of these to classify a text as either lie or truth. This yielded accuracy of around 53.33%. We had used the formula (3.9) and (3.10) to calculate probability of occurrence of lie or truth and then classified the text sample as same and obtained accuracy of 57.96% accuracy.



```
text is: People love my doritos encrusted brussle sprout cake!

category is: optimism  , count is: 1.0

category is: love  , count is: 1.0

category is: affection  , count is: 1.0

category is: plant  , count is: 1.0

category is: friends  , count is: 1.0

category is: positive_emotion  , count is: 1.0

text is: Its ok for a teenager to ignore their curfue so long as they do not do it very often.

category is: optimism  , count is: 1.0

category is: love  , count is: 1.0
```

Fig. 4.6 Categories and their count generated through Empath.

Table 4.2 Results obtained with experimentation on dataset1.

Features	**Classifier**	**Results (accuracy)**
Count vectors	Naive bayes	52.63%
TF-IDF vectors	Naive bayes	56.63%
Hash vectors	Naive bayes	54.84%
Hash vectors	Passive aggressive	53.36%
Hash vectors	SGD	55.75%
Count+TFIDF +ngrams	SGD	54.24%
Text+country+ngrams	SGD	61.62%
Char count+word count+word density	SGD	53.45%
Text+word+char count+avgerage length	SVM	56.61%
Stemming	Naive bayes	58.88%
Preprocessing+lemmatization	Naive bayes	61.66%
Word embeddings	-	Trainingdata=84.34%;test data=58.46%
LSA	SGD	53.35%

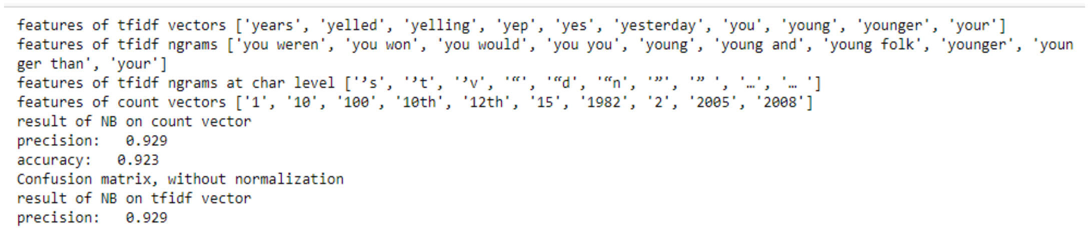
Table 4.2 (continued)

k-fold+count+ngrams	SVM	56.66%
POS+char+word count+TF	SGD	59.96%
TF vectors	Random forest	60.78%
TF vectors	Logistic regression	59.32%
TF vectors	Ada boost	59.12%
TF vectors	Gradient boost	59.71%
POS+TF+ngrams	SVM	57.168%
Empath+TF+ngrams	Naive bayes	62.61%

4.3. EXPERIMENTS ON REAL-LIFE TRIAL DATA

This dataset contained each of the 60 truthful testimonies and 61 false testimonies recorded from the courts, transcript into textual format for the purpose of deception detection through verbal cues. We collected all the truthful text and the false texts into a pandas data frame along with their correct label.

The data samples were modified into their numerical counterparts using count vectorizer, TF-IDF vectorizer and TF scores to generate feature vectors. These feature vectors were also combined with n-grams obtained on word level and character level of documents. The features obtained were fed into a number of classifier including multinomial naive bayes, SVM, random forest, KNN and stochastic gradient classifiers. The results obtained on these features were highly motivating. We obtained 84.61% accuracy when used TF-IDF vectors in SVM model while resulted in 69.2% accuracy with KNN classifier.



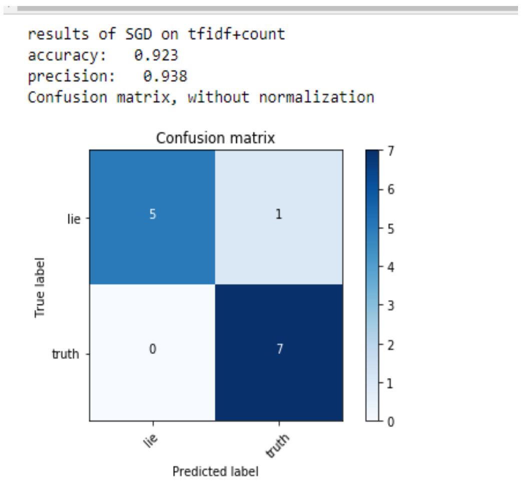
```
features of tfidf vectors ['years', 'yelled', 'yelling', 'yep', 'yes', 'yesterday', 'you', 'young', 'younger', 'your']
features of tfidf ngrams ['you weren', 'you won', 'you would', 'you you', 'young', 'young and', 'young folk', 'younger', 'youn
ger than', 'your']
features of tfidf ngrams at char level [''s', ''t', ''v', '“', '“d', '“n', '”', '” ', '…', '… ']
features of count vectors ['1', '10', '100', '10th', '12th', '15', '1982', '2', '2005', '2008']
result of NB on count vector
precision:   0.929
accuracy:   0.923
Confusion matrix, without normalization
result of NB on tfidf vector
precision:   0.929
```

Fig. 4.7 Features included in count and TF-IDF vectors.

We also used feature engineering to extract feature vectors using hidden numerical features in the linguistic model of the texts. The feature columns that were added were:

- Word count of each document
- Word density present in each document
- Punctuation count of each document
- Character count of each document

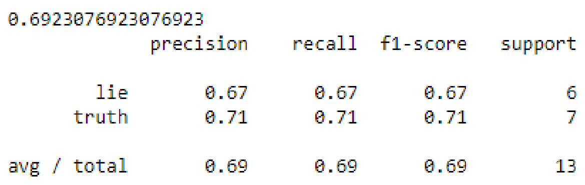
However the results obtained using these numerical features didn’t prove to be effective in classifying deception and yielded only 46.0% accuracy while the precision was 77.27%. These features were also combined with hash vector obtained on the text but didn’t help in increasing the results.



```
results of SGD on tfidf+count
accuracy:   0.923
precision:   0.938
Confusion matrix, without normalization
```

Fig. 4.8 Result of feature union (count and TF-IDF vectors) on SGD classifier.

We used POS tagged text samples as features to understand the linguistic style and semantic constructs of deceptive speeches in order to obtain cues that effectively identify deception. These features were combined along with n-grams and TF-IDF vectors of text to obtain a final single feature column which was input into the classifier algorithm and yielded 84.63% accuracy when fed to SVM classifier.

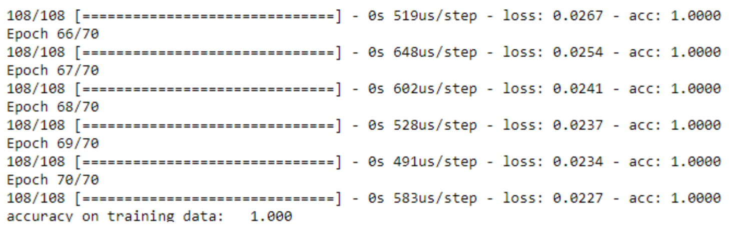


```
0.6923076923076923
             precision    recall  f1-score   support

        lie       0.67      0.67      0.67         6
      truth       0.71      0.71      0.71         7

avg / total       0.69      0.69      0.69        13
```

Fig 4.9 Result of POS features.

We used stemming, lemmatization, punctuation, common and rare words removal as part of the processing step done on the dataset to yield better results and then transformed those using TF scores in the TF-IDF vectorizer and obtained feature vectors to feed into the classifier whose parameters were fine tuned for better accuracy using the GridSearchCV function.

We used word embeddings that were learned using the dataset and then implemented using the keras library. The keras model included a sequential layer, an embedding layer and a flatten layer before using a final dense layer to build the classifier model. The embedding layer included the input size of the input vectors of the tokenized, then encoded and padded text samples, their vocabulary size and the output vectors' dimension for each word embedding that is learnt. The model is then compiled by specifying the activation function and loss function along with the optimizer that refines them. The model was evaluated on both the parts of data to examine how well it has learned the embeddings and how can it perform on unseen data.



```
108/108 [==============================] - 0s 519us/step - loss: 0.0267 - acc: 1.0000
Epoch 66/70
108/108 [==============================] - 0s 648us/step - loss: 0.0254 - acc: 1.0000
Epoch 67/70
108/108 [==============================] - 0s 602us/step - loss: 0.0241 - acc: 1.0000
Epoch 68/70
108/108 [==============================] - 0s 528us/step - loss: 0.0237 - acc: 1.0000
Epoch 69/70
108/108 [==============================] - 0s 491us/step - loss: 0.0234 - acc: 1.0000
Epoch 70/70
108/108 [==============================] - 0s 583us/step - loss: 0.0227 - acc: 1.0000
accuracy on training data:   1.000
```

Fig. 4.10 Result of word embeddings.

The k-fold cross validation technique was used along with n-grams and bag of word approach to get better accuracy. The results obtained on them yielded 66.15% accuracy when fed into SVM. We also used this validation technique on the voting ensemble of classifiers that combines predictions from various classifier models. We obtained 65.42% accuracy with this model. We used Ada boost and Gradient boosting classifier to use ensemble of classifier but obtained only 59.04% and 61.18% accuracy when fed with bag-of-words and n-grams as features, respectively. We also applied the shuffle split function of cross validation method.

We used truncated SVD as an implementation of the LSA or LSI (latent semantic indexing) for filtering out the features that hold more information for the classification process and using them as an input to the model. It takes TF-IDF features and selects the most informative out of these, from each vector and returns them as the final feature vector.

Table 4.3 Results obtained during experimentation on dataset2.

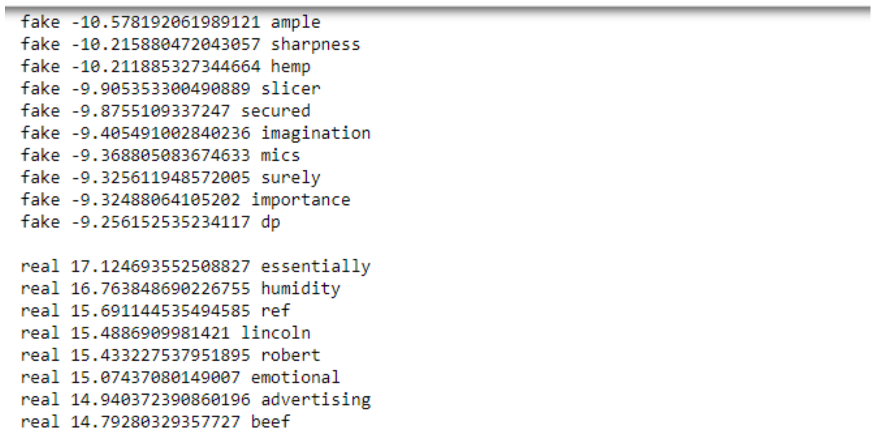
Features	**Classifier**	**Results (accuracy)**
Count vectors	Naive bayes	92.91%
TF-IDF vectors	Naive bayes	92.91%
TF-IDF+ ngrams	Naive bayes	87.55%
TF-IDF+char level ngrams	Naive bayes	77.32%
POS features	SVM	69.23%
Preprocessing+lemmatization	SVM	69.23%
Word embeddings	-	Trainingdata=100% ;test data= 54.8%
TF-IDF vectors	Random forest	76.91%
TF-IDF vectors	Logistic regression	92.32%
LSA	MLP	84.65%
Count+TFIDF+ngrams	SGD	92.34%
Stemming+count vector	Naive bayes	84.6%
LDA	SGD	54.8%

4.4.EXPERIMENTS ON PRODUCT REVIEWS DATA

We loaded all the 21000 reviews into the pandas data frame object from the pandas library. The LABEL column contained the annotated labels corresponding to the class to which the reviews belonged. We encoded them in readable form by changing them to their correct meaning i.e. '_label1_' was modified to 'FAKE' and '_label2_' was modified to 'REAL' and storing them in a separate column.

The review text column was first used to extract features by vectorizing them into numerical vectors with the help of count vectorizer, TF-IDF vectorizer, hash vectorizer and TF score values as feature vectors. These features were also combined with n-grams on character level and word level structure of the documents. The n-grams thus obtained were modified using these vectorizers to obtain final feature vectors that were fed into the classifier model. Some of the classifiers that were used throughout the experimentation to test different models were multinomial naive bayes, passive aggressive classifier, SVM, random forest classifier, logistic regression/linear classifier, K nearest neighbour classifier, stochastic gradient classifier etc. the result of TF-IDF vectors with passive aggressive and SGD classifier obtained accuracy of 59.7% and 65.8% , respectively. While count vectors with SVM classifier obtained 62.7% accuracy. Logistic regression classifier when fed TF vector yielded 65.6% accuracy.

We used stemming in combination with TF-IDF transformer and count vectorizer to generate features. We also implemented a pipeline of bag-of-words approach to generate features in combination with the n-gram model along with chi square function to select the best features from the features generated so that the most informative features are only used to train the classifier and hence improving the results. When fed into SVM model these features generated accuracy of 64.8%. We also displayed the features/terms with the most weightage in the TF-IDF scheme of vectorization. This helps in studying the terms or topics that are used most frequently by the reviewers/spammers to post fake reviews and hence identifying the hidden linguistic cues that signify deceptive behaviour.



```
fake -10.578192061989121 ample
fake -10.215880472043057 sharpness
fake -10.211885327344664 hemp
fake -9.905353300490889 slicer
fake -9.8755109337247 secured
fake -9.405491002840236 imagination
fake -9.368805083674633 mics
fake -9.325611948572005 surely
fake -9.32488064105202 importance
fake -9.256152535234117 dp

real 17.124693552508827 essentially
real 16.763848690226755 humidity
real 15.691144535494585 ref
real 15.4886909981421 lincoln
real 15.433227537951895 robert
real 15.07437080149007 emotional
real 14.940372390860196 advertising
real 14.79280329357727 beef
```

Fig. 4.11 Most informative features/words in binary classification.

K-fold cross validation method was used by us to demonstrate the learning accuracy of the models developed using n-grams and count vectors as features, yielding accuracy of 66.46% when fed into SVM. This validation technique was also employed to evaluate the results of the voting ensemble of the best performing models among the models developed for the purpose of identifying fake reviews. Ensemble of classifiers were also implemented using Gradient boosting and Ada boost classifiers that run various epochs of the same classifiers in order to minimize the loss and maximize the accuracies henceforth obtained. The function shuffle split cross validation was also implemented to evaluate the effect of learning on the data that doesn't hold any unexplained benefit of unarranged data and its position, rather than the model for which it is trained and should produce results for.

Numerical features were engineered from the available main text and metadata in the dataset. These features were obtained from lexical structure of the reviews and its related information. The features extracted and included in new separate columns were:

- Number of punctuation characters in a review
- Number of capital letters count in review
- Number of total words present in a review

- Average word length of a review
- Average length of a review itself
- Sentiment present in the reviews

These features were extracted to find out hidden inferences and deception cues that are not directly visible or present in the dataset itself. These features were used separately as standalone features for the input to the model or combined among themselves and review text of the data samples. The n-grams on reviews and corresponding bag-of-word features were combined with the extracted features to include as many as relevant information in the final feature set as possible for the purpose of increasing the accuracy of the classifier.

The other information present in the dataset apart from the main review text was used to extract features like, product category, verified purchase, ratings etc. these information columns were combined along with the review text to add more information to the features, other than just the review to be classified. These feature contained information representing the reviewer behaviour and background information apart from just the content of the reviews and hence are more informative for the classifier model.

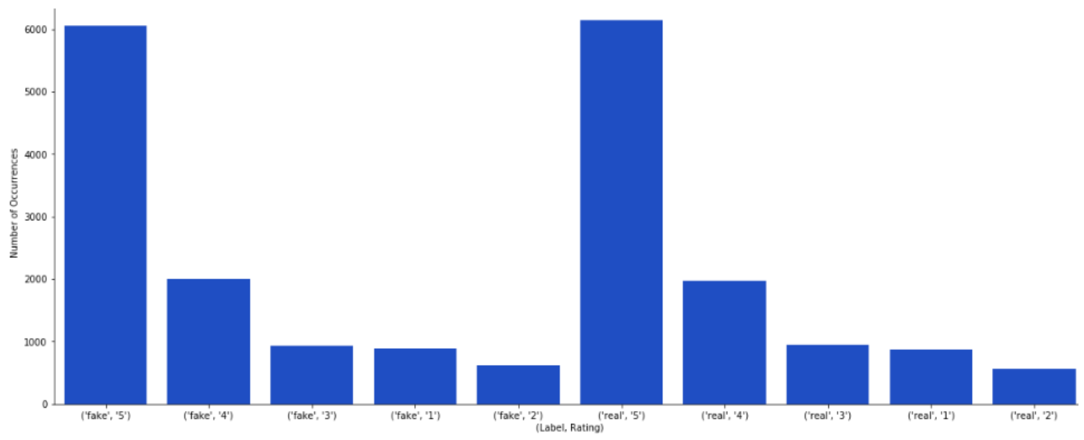


Fig. 4.12 Distribution of ratings in fake and real categories.

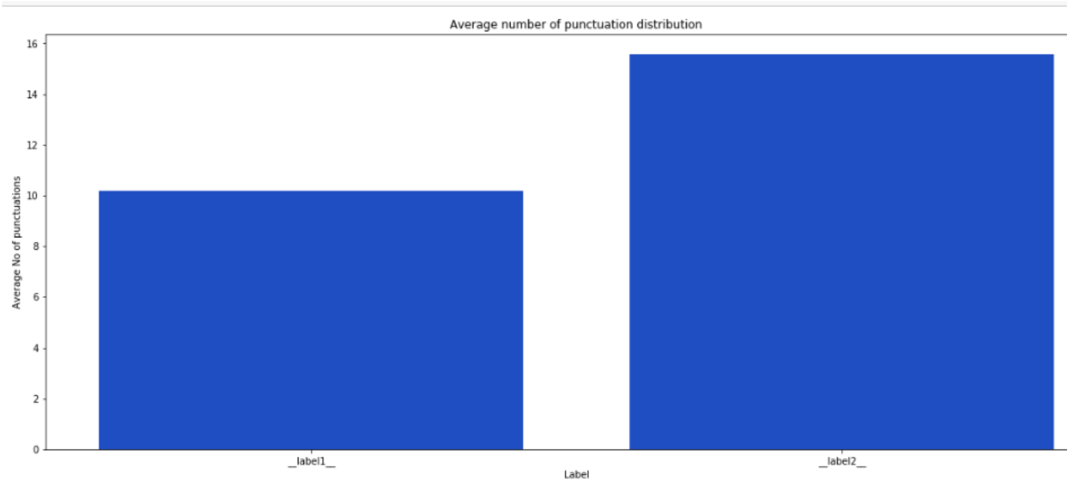


Fig. 4.13 Average number of punctuation symbols in fake and real class.

POS tagged reviews were used to examine the semantic and linguistic structure of the documents. The features extracted for it was also used in combination with the TF scores and n-grams features of the text, resulting i accuracy of 62.4% when fed into SVM classifier. These features were used separately as input to the classifier model to study the effect of each in classifying the reviews correctly and identifying their corresponding labels. We also used LDA to generate topics present in the reviews and hence classifying them using those features. These topic modelling features were also used along with the bag-of-words and n-grams as features to obtain a single combined feature as input to the model. SVD was also used to extract the most crucial features from the TF-IDF feature vectors and generate a better set of features to be fed into the model.

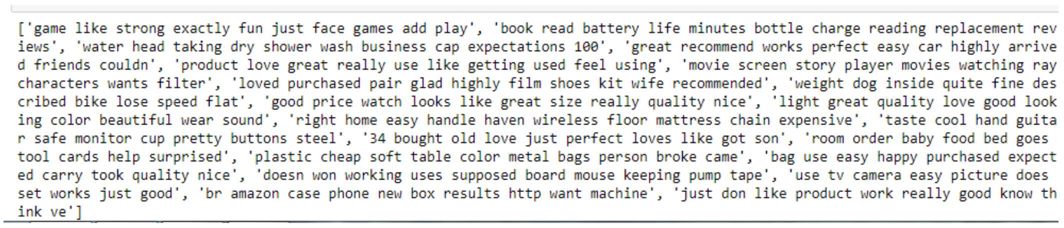

```
['game like strong exactly fun just face games add play', 'book read battery life minutes bottle charge reading replacement rev
iews', 'water head taking dry shower wash business cap expectations 100', 'great recommend works perfect easy car highly arrive
d friends couldn', 'product love great really use like getting used feel using', 'movie screen story player movies watching ray
characters wants filter', 'loved purchased pair glad highly film shoes kit wife recommended', 'weight dog inside quite fine des
cribed bike lose speed flat', 'good price watch looks like great size really quality nice', 'light great quality love good look
ing color beautiful wear sound', 'right home easy handle haven wireless floor mattress chain expensive', 'taste cool hand guita
r safe monitor cup pretty buttons steel', '34 bought old love just perfect loves like got son', 'room order baby food bed goes
tool cards help surprised', 'plastic cheap soft table color metal bags person broke came', 'bag use easy happy purchased expect
ed carry took quality nice', 'doesn won working uses supposed board mouse keeping pump tape', 'use tv camera easy picture does
set works just good', 'br amazon case phone new box results http want machine', 'just don like product work really good know th
ink ve']
```

Fig 4.14 Top words in topics generated by LDA.

The reviews and related data present in the dataset were also processed to improve the tokens generated and maintaining the context while removing the not so important information present in the features. The common and rare words were removed, punctuation characters were removed, and stemming and lemmatization were performed to study the effect of these procedures on the process and results of the classification.

Word embeddings were used to learn the numerical representation of the text using the keras model. These embeddings were used as a feature in the classification process. The sequential keras model for learning the word embeddings was made up of the embedding layer and the dense layer, which included a flatten layer in between for the right conversion of the vector dimensions between the input and output vectors. The embedding layer contained all the required parameters for the input and output. The model was later compiled and then evaluated on the training and testing data to see its performance in learning and implementing the word embeddings.

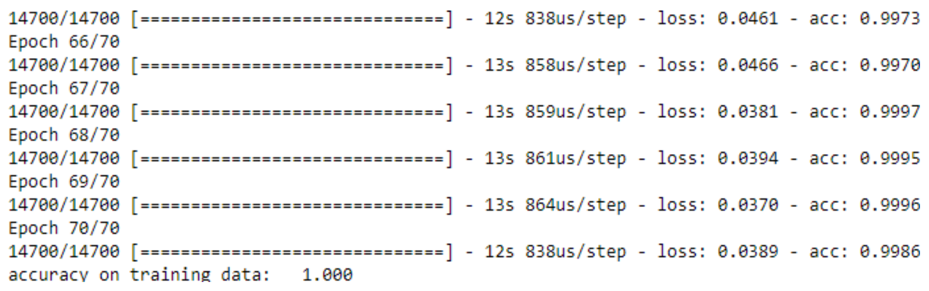

```
14700/14700 [==============================] - 12s 838us/step - loss: 0.0461 - acc: 0.9973
Epoch 66/70
14700/14700 [==============================] - 13s 858us/step - loss: 0.0466 - acc: 0.9970
Epoch 67/70
14700/14700 [==============================] - 13s 859us/step - loss: 0.0381 - acc: 0.9997
Epoch 68/70
14700/14700 [==============================] - 13s 861us/step - loss: 0.0394 - acc: 0.9995
Epoch 69/70
14700/14700 [==============================] - 13s 864us/step - loss: 0.0370 - acc: 0.9996
Epoch 70/70
14700/14700 [==============================] - 12s 838us/step - loss: 0.0389 - acc: 0.9986
accuracy on training data:   1.000
```

Fig 4.15 Result of word embeddings through keras on product reviews.

Empath was used to extract the hidden semantic features present in the language style of the reviews and its contents. The categories and their occurrence counts returned for text samples were combined to generate hardcoded numerical feature vectors for training and testing data separately. These feature vectors were input to the classifier model to obtain classification results. When used as a standalone feature fed into SVM they obtained 54.92% accuracy. We also combined these feature matrices with n-grams of reviews and TF scores on them to generate final feature vectors for the classification process.

Table 4.4 Results obtained with experiments on dataset 3.

Features	**Classifier**	**Results (accuracy)**
Count vector	Naive bayes	64.25 %
TF-IDF vector	Naive bayes	65.97%
Hash vector	Passive aggressive	61.7%
Hash vector	Naive bayes	65.3%
Hash vector	SGD	64.6%
Count+ TFIDF +ngram	SGD	62.4%
Stemmed count +TFIDF	Naive bayes	66.8%
Punctuation + capital char count+ average length	Naive bayes	54.8%
POS features	SVM	66.26%
Processing+ lemmatization	Naive bayes	66.25%
Word embeddings	-	Train size=100% Test size=59.8%
Text+ratings+ngrams	SVM	64.4%
Text+ product category+ngram	SGD	63.8%
Text +verified purchase+ TFIDF	SVM	82.1%
LSA	SGD	65.3%
LDA+ TF vector	NB	58.7%
TF vector	Random forest	63.4%
TF vector	KNN	56.9%
Empath + ngram+ TF vector	SVM	66.57%
Text +sentiment + ngrams	SVM	84.4%
Text+ product category+ Ratings+ verified purchase	SVM	82.3%
TF vector	Gradient boost	60.1%
TF vector	Ada boost	59.8%
Combination of above models	Voting classifier	67.8%

4.5.SUMMARY

We had used all the 3 datasets for the purpose of deception detection and studying its techniques when applied to different scenarios and quality and type of data. First we parsed the datasets using the pandas library. The general NLP techniques applied on the datasets included conversion of data into feature vectors using count vectroizer, TF-IDF vectorizer and hash vectorizer and also TF scores as values in some of the vectors. These features were also combined with n-grams of the text. Those resulting feature vectors were then fed into classifiers such as multinomial naïve bayes, SVM, stochastic gradient descent, passive aggressive classifier, logistic regression, k nearest neighbour, random forest classifiers, ensemble of classifiers that included voting ensemble, gradient boosting and ada boost classifier etc. these models also employed the k-fold cross validation technique for obtaining better results on the data by increasing its learning rate.

Feature extraction and feature selection modules were used for the purpose of combining features together or selecting some of the best features among them which also helps in dimensionality reduction. These were also done using SVD and LSA which in a way extract the main/important points from the text and generate features from them. Numerical or hidden features from the text samples were extracted as part of feature engineering. Some of these were: the word count, character count, word density etc. We had also used the main text along with other information present in the dataset as features that include additional relevant information that is helpful in the classification process. We also performed processing on the data to obtain better and important tokens as features such as, frequent words removal, stemming, lemmatization etc.

Word embeddings were used to convert text into features based on their language structure and similarities between words/sentences. These were implemented using keras library of the python. The text samples were tokenized and encoded and fed into the embedding layer as input vectors. The sequential model of keras also included a flatten layer and a dense layer that contains the activation function for producing the output vectors i.e. the learned embeddings. The model was then compiled and evaluated on the data. The classification results obtained by using word embeddings as features were really promising and better than other features.

POS tags corresponding to the tokens in the texts were used as features to take into account the linguistic style of deceptive texts. These features were combined along with bag-of-words and n-grams model to generate feature vectors.

Empath library and its functions were used to generate semantic categories present in the text of the data. These categories and their count were converted into hand-coded feature matrices combining each of the function results. These features were input to the model for classifying the text. We also used bag-of-words approach along with n-grams to combine with these hardcoded features to generate single final feature vectors for classification.

We analyzed that the open-domain deception data mostly didn't yield good results for classification because of the major reason that it was freely collected without any specific topic or target and hence lacked the structure that should be present for learning classification correctly. Since it contained lies and truths from across all domains and depth of imagination because they were crowd sourced and hence could be considered as pseudo lies instead of deceptive language that would be found in real situations, it made an already tough job of deception detection, even tougher.

The product reviews dataset and the trial transcripts dataset were favourable for the purpose of deception detection because they contained actual deceptive behaviour of people in real situations. Hence they contained in a sense true deceptive cues hidden in them. They contained supporting information along with the main text that further helped in providing additional information for the purpose of classification. Due to all these factors the accuracies obtained by classification models on these dataset were quite good and promising for further research scope in these topics.

CHAPTER 5

CONCLUSIONS AND FUTURE SCOPE

5.1.CONCLUSION

In recent years, fake and deceptive contents have become increasingly difficult to segregate and identify in the scenario of easy and fast creation and sharing of such misinformation. Fake reviews are affecting businesses and potential customers both. Deceptive contents are present in day to day communications and online platforms alike. Hence it has become the need of the hour to make use of technology to defeat the purpose of malicious users who create and disseminate such false contents for deception of the unsuspecting people. As indicated in [8], deception detection is a very tough task especially without the presence of adequate information in relevant formats in the form of datasets.

We have applied NLP techniques for this purpose such as bag-of-words and also made use of n-grams as features. The n-grams extract the contextual meaning of the texts and hence act as better features to obtain deceptive cues hidden in the language structure. The n-grams that we extracted from the main text were also used in combination with the other feature sets that resulted in performance improvement, such as, numerical features engineered from the dataset, POS etc.

We also used semantic features for identifying the semantic structure of the deceptive language used by the users/people. The features extracted for it were: word embeddings to represent textual data into corresponding numerical feature vectors based on their semantic similarity and representation in the multidimensional vector space. The categories to which words belong based on their semantic and lexical meanings were generated by using empath. It returned categories and their counts present in a text piece in the form of a python dictionary, which we converted into matrix form and generated feature vectors corresponding to each data samples and hence trained the classifier on them. These features performed well because they took into account the language style in deceptive texts.

Linguistic cues associated with deception were identified using lexical and syntactic features that depict the same. The features extracted for this included numerical and binary features that were developed from the information present in the dataset like-Word count of documents, word density of documents etc. POS tagged documents were also used as features for examining the efficiency of grammatical structure of texts as features in the classification process.

Various techniques for improving the classifier model were also employed such as LSA, LDA, k-fold cross validation, shuffle split CV, grid search CV etc. Different classifiers were also combined together to increase the efficiency or by combining the features extracted from different schemes. Dimensionality reduction and best k features were also selected for improving the results along with preprocessing the data and tuning the parameters.

However there is still scope in improving the classification model by improving the quantity and quality of the data that is more inclusive of the relevant information for the purpose of deception detection. The features extracted from it would then include better deception cues which would result in increased accuracy of classification model.

5.2. FUTURE WORK

The classification model that we developed for deception detection made use of everything from preprocessing methods to different types of feature extraction methods that carve out the linguistic, semantic features and so on. We employed word embeddings and semantic and lexical category generator along with examining the syntactic structure of documents. Feature engineering methods were used to dig out other hidden features in the dataset and feature union, feature selection and feature extraction methods were also used to improve and hybrid already known features for gaining some accuracy on the classifiers.

Since real world data that have all the necessary information and correct labels are hard to find, hence unsupervised and semi-supervised methods for classifying deceptive content are also suitable for this task. However they generally do not match the results of supervised learning but they can definitely be given a try and combined with supervised classification models to obtain a hybrid approach that takes the best of both worlds. The dataset might be improved by including other types of data than just text, such as images, videos etc. however these require a lot of computational effort and resources.

Other features that might turn out to be useful can be features that use data analytics and return statistical numbers/information or features that explore the reviewer behavior and language style of users more precisely like use of strong language, writing patterns with more usage of some category of words etc. other than these, models that make use of neural networks like RNN, CNN etc. can also be used by taking into account higher dimensional processing and methods of deep learning. Some kind of a fact checking system based on knowledge sources can also be used in cases where there is clear distinction between facts and imaginative writing.

REFERENCES

[1].V. P. Rosas, B. Kleinberg, A. Lefevre and R. Mihalcea, “Automatic Detection of Fake News”, Proceedings of the 27th International Conference on Computational Linguistics, August 2018.

[2].Á. Figueiraa, L. Oliveirab “The current state of fake news: challenges and opportunities”, CENTERIS International Conference on Project MANagement / HCist, 2017, Spain.

[3].BBC News (2019), “Samsung probed in Taiwan over 'fake web reviews'”, April 16, 2013 [BBC News]. Available at: https://www.bbc.com/news/technology-22166606

[4].The New York Times (2019), “For $2 a Star, an Online Retailer Gets 5-Star Product Reviews” Jan 26, 2012 [The New York Times]. Available at: https://www.nytimes.com/2012/01/27/technology/for-2-a-star-a-retailer-gets-5-star-reviews.html?_r=2&ref=business

[5].7NEWS. (2019). “Woman Paid To Post Five-Star Google Feedback”, September 15, 2012 [ABC7 News]. Available at: http://www.thedenverchannel.com/news/woman-paid-to-post-five-star-google-feedback

[6].M. Anderson (2019). “88% of Consumers Trust Online Reviews As Much As Personal Recommendations”, July 7, 2014. Available at: http://searchengineland.com/88-consumers-trust-online-reviews-much-personal-recommendations-195803

[7].N. Jindal, & B. Liu, “Opinion Spam and Analysis”. In Proceedings of the 2008 International Conference on Web Search and Data Mining (pp. 219–230). February, 2008. New York, NY, USA: ACM.

[8].M. Ott, Y. Choi, C. Cardie, & J.T. Hancock, “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). June, 2011. Stroudsburg, PA, USA: Association for Computational Linguistics.

[9].A. Mukherjee, V. Venkataraman, B. Liu, & N. Glance, “Fake Review Detection: Classification and Analysis of Real and Pseudo Reviews”. UIC-CS-03-2013. Technical Report.

[10].A. Mukherjee, V. Venkataraman, B. Liu, & N. Glance, “What yelp fake review filter might be doing”. In Proceedings of the International Conference on Weblogs and Social Media. January, 2013.

[11].R. Y. Lau, S. Y. Liao, R. C. W. Kwok, K. Xu, Y. Xia, Y. Li, “Text mining and probabilistic language modeling for online review spam detecting”. ACM Trans Manage Inf Syst 2(4) (pp: 1–30). March, 2012.

[12].S. Shojaee, M. A. A. Murad, A. B. Azman, N. M. Sharef, & S. Nadali, "Detecting deceptive reviews using lexical and syntactic features". In Intelligent Systems Design and Applications (ISDA), 2013 13th International Conference on (pp. 53-58). December 2013 IEEE.

[13].A. Mukherjee, A. Kumar, B. Liu, J. Wang, M. Hsu, M. Castellanos, & R. Ghosh, " Spotting opinion spammers using behavioral footprints." In Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 632-640). August, 2013. ACM.

[14].E. P. Lim, V. A. Nguyen, N. Jindal, B. Liu, & H. W. Lauw , "Detecting product review spammers using rating behaviors". In Proceedings of the 19th ACM international conference on Information and knowledge management (pp. 939-948). October, 2010. ACM.

[15].S. Feng, L. Xing, A. Gogar, & Y. Choi , "Distributional Footprints of Deceptive Product Reviews". ICWSM, 12, 98-105. January, 2012.

[16]. G. Fei, A.Mukherjee, B. Liu, M. Hsu, M. Castellanos, & R. Ghosh(2013). "Exploiting Burstiness in Reviews for Review Spammer Detection". ICWSM, 13, 175-184. January, 2013.

[17]. S. Xie, G. Wang, S. Lin, & P. S. Yu, "Review spam detection via time series pattern discovery". In Proceedings of the 21st International Conference on World Wide Web (pp. 635-636). April, 2012. ACM.

[18].M. Granik and V. Mesyura, "Fake News Detection Using Naive Bayes Classifier", 2017 IEEE First Ukraine Conference on Electrical and Computer Engineering (UKRCON)

[19].S. Gilda, "Evaluating machine learning algorithms for fake news detection", 2017 IEEE 15th Student Conference on Research and Development (SCOReD). December, 2017.

[20].S. Singhania, N. Fernandez, and S. Rao, "3HAN: A Deep Neural Network for Fake News Detection", International Conference on Neural Information Processing, November 2017.

[21]. Y.Yang, L.Zheng, J.Zhang, Q.Cui, X.Zhang, Z.Li, and P. S. Yu, "Convolutional Neural Networks for Fake News Detection", Published in ArXiv 2018. June, 2018.

[22].V.P´.Rosas and R.Mihalcea,"Experiments in Open Domain Deception Detection", Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, September 2015, (p.1120–112), Association for Computational Linguistics. Dataset available at: http://web.eecs.umich.edu/~mihalcea/downloads.html

[23].N. J. Conroy, V. L. Rubin, and Y.Chen, "Automatic Deception Detection: Methods for Finding Fake News", Proceedings of the Association for Information Science and Technology banner, February 2016, Association for Information Science and Technology.

[24]. Á. Almela, R. V. García, P.Cantos,"Seeing through deception: A computational approach to deceit detection in written communication", Proceedings of the

Workshop on Computational Approaches to Deception Detection, April 2012, (pp.15-22), Association for Computational Linguistics.

[25].I.Fette, N.Sadeh and A.Tomasic, “Learning to Detect Phishing Emails”, Proceedings of the 16th international conference on World Wide Web, (pp.649-656), May, 2007.ACM.

[26].B. d. Ruiter, G. Kachergis, “The Mafiascum Dataset: A Large Text Corpus for Deception Detection”, Published in ArXiv, December, 2018, Computation and Language, arXiv:1811.07851v2

[27].G. Krishnamurthy, N. Majumder, S. Poria, and E. Cambria, “A Deep Learning Approach for Multimodal Deception Detection”, 19th International Conference on Computational Linguistics and Intelligent Text Processing (CICLing), March, 2018.

[28].M.Jaiswal ; S.Tabibu ; R.Bajpai, “The Truth and Nothing But the Truth: Multimodal Analysis for Deception Detection”, 2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW), December, 2016.

[29].V. P. Rosas, M. Abouelenien, R. Mihalcea and M. Burzo, “Deception Detection using Real-life Trial Data”, Proceedings of the 2015 ACM on International Conference on Multimodal Interaction, (pp.59-66), November, 2015. Dataset available at: http://web.eecs.umich.edu/~mihalcea/downloads.html

[30].E.Fast, B.Chen, M.S.Bernstein,“Empath: Understanding Topic Signals in Large-Scale Text”,Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems, (pp. 4647-4657), May, 2016.

[31].Stanford (2019), “Empath”. Available at: http://empath.stanford.edu/

[32].Ethan Fast (2019) “Empath-client”, April 22, 2017. Available at: https://github.com/Ejhfast/empath-client

[33].Kaggle (2019) “Amazon reviews”, 29 January, 2019. Available at: https://www.kaggle.com/lievgarcia/amazon-reviews/metadata