Natural Language Understanding

AT70.23

Asian Institute of Technology

School of Engineering and Technology

Project report on

Text similarity in fake news classification in COVID-19 domain

Submitted to:
Prof. Phan Minh Dung

Submitted by:
Abhishek Koirala (st120199)
Suprava Sahoo(st120984)

Submitted Date: 21st July 2020

CHAPTER 1
INTRODUCTION

There was one news spreading all over the internet, **"Sanitizers Don't Help Against Coronavirus".** This news was shared on social media with a reasoning that because coronavirus is a virus and anti-bacterial hand sanitizer won't do anything. According to FactCheck.org, the rumor started on Twitter when a self-identified "scientist" posted their theory on March 1, 2020. However, according to the CDC, using a hand sanitizer that contains at least 60 percent alcohol is a suitable substitute if there is no soap and water available.

There was another story going around, **"The federal government has mandated that workplaces close for two weeks to avoid the spread of coronavirus".** However, no such mandate was made. According to FactCheck.org, the posts being shared were intended as a joke with a punchline awaiting users when they click the link to "read more." Other posts also switched out the "government" reference to make the claim specific to Michigan, Florida, Alabama and other states. States of emergency had been declared in some parts of the country, but it did not extend to forcing workplaces to close.

These two are representative stories. We see these kinds of stories on our social media platform everyday. And because these stories create sensationalism, people tend to agree to it and start sharing it. One thing they don't realize is that these stories are fake.

There has been no universal definition for fake news, even in journalism. (Zhou & Zafarani,2018) attempts to define fake news by two approaches. With the broader approach, fake news is defined as false news where news broadly include claims, statements, speeches, posts, among other types of information related to public figures and organizations. This definition supports most existing fake-news-related studies and dataset, as provided by the existing fact-checking websites. With the narrower approach, fake news is defined as a story intentionally and verifiably false published by a news outlet. His narrow definition addresses the public's perception of fake news, especially following the 2016 U.S. presidential election. Note that deceptive news (i.e., maliciously false news) is more harmful and less distinguishable than incautiously false news, as the former pretends to be truth to mislead the public.

Fake news can be created to create influence on political views or can be a viral post based on fictitious accounts made to look like real news reports. They are fabricated information that mimics a real news media content. While a real news report goes through the media's editorial process, the fake news lacks such a process and also lacks accuracy and credibility of information. Fake news often overlaps with misinformation(false or misleading information) and

disinformation(false information purposely spread to mislead people (Granik and Mesyura (2017).

Many perspectives on who creates fake news, how and why it is created, how it propagates, and how it can be detected motivate the need for an in-depth analysis.This survey aims to develop a systematic framework for the comprehensive study of fake news. As fake news is not clearly defined and current studies of fake news are limited, we extend our study to related fields that can serve as a foundation for fake news research. We hope this survey can facilitate fake news studies by inspiring researchers vertically, to extend current fake news studies in-depth, and horizontally, to enrich and improve fake news studies by interdisciplinary research.

People fall for fake news despite having easy access to real facts and fact checkers. Part of the reason that fake news is such a problem is that people do fall for these stories, and being shown the facts doesn't help correct this problem. This is due to a feature of human thinking called cognitive biases. A cognitive bias is a gap in reasoning, remembering, or evaluating something that can lead to mistaken conclusions. They're universal. Everyone has them. Cognitive biases affect the way we use information. Four types of cognitive biases are especially relevant in relation to fake news and its influence on society: First, we tend to act on the basis of headlines and tags without reading the article they're associated with. Second, social media convey signals that affect our sense of the popularity of information, which leads us to greater acceptance. Third, fake news takes advantage of the most common political mental shortcut: partisanship. And fourth, there's a weird tendency for false information to stick around, even after it's corrected.

CHAPTER 2:
LITERATURE REVIEW

N. Shibata et. al., (2010) performed a comparative study to measure the semantic similarity between academic papers and patents, using three methods like the Jaccard coefficient, cosine similarity of tf-idf vector, and cosine similarity of log-tf-idf vectors. All of these methods are corpus-based methods and they also performed a case-study for further analysis. S. Zhang et. al, (2015) explored the different applications of semantic similarity related to the field of Social Network Analysis. H. Pu et. al., (2017) incorporated a semantic similarity calculation algorithm that used the large corpus to compare and analyze several words. This method used the Word2Vec model to calculate semantic information of the corpus and used Li similarity measure to compute similarity. Wael H. Gomaa, (2013) also discussed three text similarity approaches: string-based, corpus-based and knowledge-based similarities and proposed different algorithms based on it. The above works showed that different methods and metrics have already been carried out for this research.

In this experiment, we have also introduced three different semantic similarity measures based on both corpus-based and knowledge based methods. The basic concept was that the small features from text words are not enough, and the inclusion of word semantic information in the process can improve the method. The tf-idf vectors were used to gain information about the corpus or document in case of a corpus-based method ie. cosine similarity using tf-idf vectors. The knowledge-based method included computation of word embedding for each text, and cosine and soft-cosine similarity metrics were used for similarity calculation between these word embedding vectors.

A. Cosine Similarity using TF-IDF Vectors

The pre-processed documents were converted into tf-idf vectors by using a vectorized tf-idf model. The obtained vectors were a sparse matrix containing tf-idf weights for each word of each document having the size of [number of documents * number of features(unique words)]. Now, these tf-idf weights from the matrix were used as a feature for each document, and similarity between documents are computed using cosine similarity. The inbuilt cosine similarity module from sklearn was used to compute the similarity.

B. Cosine Similarity using Word2Vec Vectors

In this method, the pre-trained word2vec model was loaded using gensim. This word2vec model was used to compute the vector values or word embeddings of each word of all the preprocessed documents. The word2vec vectors for the words that do not exist in the vocabulary of the pre-trained model were set to zero. An average of all the word vectors of a document was computed and the resultant vector was used as a feature word2vec vector for that document. So,

all the documents were vectorized from n-gram vectors into word2vec vectors. These word2vec vectors were then used as feature vectors to compute the cosine similarity within the documents.

C. Soft Cosine Similarity using Word2Vec Vectors

This method also used the word embeddings from the same pre-trained word2vec model. These word2vec vectors were used to construct a term similarity index which computes cosine similarities between these word2vec vectors from the pretrained model. The pre-processed documents were converted into a dictionary or a bag of words model and the formed dictionary was converted into a corpus (a sparse vector containing unique word ids and its number of occurrences). Since soft cosine measure uses a similarity matrix between pairs of features, the dictionary (feature of the dataset) and term similarity index (features of word2vec model) were used to compute a term similarity matrix. Finally, soft cosine similarity between the documents was computed. The modules provided by gensim [8] were used to compute this similarity measure.

In 2016, Alcott. et al., (2017) conducted a quantitative study to identify how fake news on social media influenced the Presidential election in the United States. They carried out a controlled survey to measure the recall and believability to find out the users' exposure to fake news on social media. Jin et. al., (2017) conducted a study by examining a corpus of over 8 million tweets collected from the followers of the two Presidential Candidates and matched text in tweets against a collection of debunked rumors from snopes.com, to identify rumors spread by tweets.
Sam Biggins et al., (2012) have researched on two approaches that can be used to achieve the semantic text similarity. One approach is a supervised machine learning method and it is based on comparing sentences based on the overlap of n-grams they contain. The second approach is an unsupervised method. It has used an extended form of vector space model by calculating the similarity between word senses and then finding the distances between vectors constructed using these distances.

Unsupervised Method-Vector Space Model

This model is mostly used to compare texts in Information Retrieval and Natural Language Processing. When creating vectors, at first, each sentence is tokenized, stop words removed and the remaining words are lemmatized using NLTK (Natural Language Toolkit). Then, for each sentence, binary vectors are created. Then cosine metric will be used to compare two vectors to compute the similarity between the sentences. According to research, it does not consider words with similar meanings such as “the dog eats foods' ' and “the hound eats foods”. WordNet based similarity measures can be used to take these similarities into account.

Similarity between two sentences will be calculated using cosine metrics. To calculate word similarity, two methods of Word Sense Disambiguation Namely, Most Frequent Sense (MFS) and Lesk are used.

Supervised Method-Vector Space Model

According to this method, sentences are represented as sets of n-grams (i.e. contiguous sequence of n items from a given sequence of texts or speech). When generating n-grams, preprocessing is carried out using NLTK. Then each sentence is tokenized and lemmatized. Finally, a set of n-grams are extracted from each sentence. Sentences will be denoted by S and sets of n-grams are denoted by S0. In this approach for each n-gram in S0 a list of alternative n-grams are generated using WordNet. These sets of expanded n-grams are denoted by Sa. When comparing sentences (S1 and S2) n-grams extracted from original sentences (S10 and S20) will be compared with the modified version (S1a and S2a) created using WordNet.

Aly A. Fahmy, (2013) have presented an approach that combines corpus-based semantic relatedness measures over the whole sentence along with the knowledge-based semantic similarity scores that were obtained for the words falling under the same syntactic roles in both sentences. All the scores as features were fed to machine learning models, like linear regression, and bagging models to obtain a single score giving the degree of similarity between sentences. This approach showed a significant improvement in calculating the semantic similarity between sentences by combining the knowledge-based similarity measure and the corpus-based relatedness measure against the corpus based measure taken alone. A promising correlation between manual and automatic similarity results were achieved by combining two modules. The first module calculates the similarity between sentences using N-gram based similarity, and the second module calculates the similarity between concepts in the two sentences using a concept similarity measure and WordNet.

Fake news studies, e.g., Lazer et al. [2018], have emphasized the importance of business models adopted by social media sites to address fake news intervention, which suggests shifting the emphasis from maximizing user engagement to that on increasing information quality, e.g., using self- or government regulations. Technically, a fake news intervention strategy can be based on network structure, which has been discussed in [Shu et al. 2018], or based on users, which we will discuss here. From a user perspective, fake news intervention relies on specific roles users play in fake news activities. One such role is being an (i) influential user (i.e., opinion leader). When blocking a certain fake news in a social network, handling these influential spreaders first, leads to a more efficient intervention compared to handling users that may have a negligible

social influence on others. Another beneficial role is being a (ii) corrector, users on social networks who take an active role in mitigating the spread of fake news by attaching links that debunk the fake news in their posts or comments [Vo and Lee 2018]. Furthermore, the intervention strategy for (iii) malicious users and (iv) naïve users should be different, while they both spread fake news; malicious users should be removed or penalized, while naïve users should be [actively or passively] assisted to improve their ability to distinguish fake news. To enhance a user's ability to differentiate fake news from true news, for example, personal recommendation of true news articles and/or related links for users can be helpful. The recommendation should not only cater to the topics that users want to read, but should also capture topics and events that users are most gullible to due to their political biases or preexisting knowledge.

This research aims at providing a domain independent approach for PI and STS analysis of Arabic news tweets using a supervised machine learning approach. Although, tweets are considered noisy, informal and personal (Xu, Callison-Burch, & Dolan, 2015), collected tweets are posted by well-known Arabic news agencies such as Al-Jazeera1 and Al-Arabiya2 where they post their breaking news using the Modern Standard Arabic (MSA). The proposed approach builds on lexical, syntactic and semantic computed measures and extracted features adapted from literature. The widely used social media have flooded their users with news talking about similar events. Therefore, detecting paraphrases and semantic similarity analysis have become a need to avoid receiving the same news post several times. Moreover, the approach uses word alignment features to detect the level of similarity between tweets pairs.

Chandrathlake, (2018) They proposed in this paper to indicate the accuracy level of a news which is posted by a social media user as a status update in the news feed. This accuracy level will be calculated by considering a comparison done with the well reputed online news providing sources such as online news sites, online newspapers and gossip sites. When the process is concerned, the first social media news update will be extracted from social media sites such as Twitter, Facebook. In this project, we consider only the news updates posted by users as status updates (as texts written in English) on their walls and do not consider the news posts which are posted by a page and shared by the users. Then the semantic of each news update will be identified and news will be categorized into different categories such as, political, weather, sports, economical and other. To do this categorizing, it is needed to fuse a word corpus which is related to different categories. Finally, the system will generate a score which will explain the accuracy level of a news post based on the similarity measure calculated between social media news posts and the content of the online news article.

CHAPTER 3
METHODOLOGY

The project mainly focuses on analyzing similarity scores in text data. The project is divided into 5 major parts as shown in figure 1.1. The first part is focused on analyzing related work in this domain. We already saw in the previous chapter that there has been application of similarity metrics calculation for appropriate clustering or classification of documents or text data. This metric does not include any form of machine learning pattern, which makes them efficient to calculate even with less amount of data. After discussion with our advisor, we decided to work on a small number of data. The second part focuses on the collection of such real news articles. We had collected seven news articles of such kind. In the third stage of the research, we worked on analyzing the collected data and tried to understand the context of it. Since these articles were collected from various sources, all of these articles followed different styles of writing.

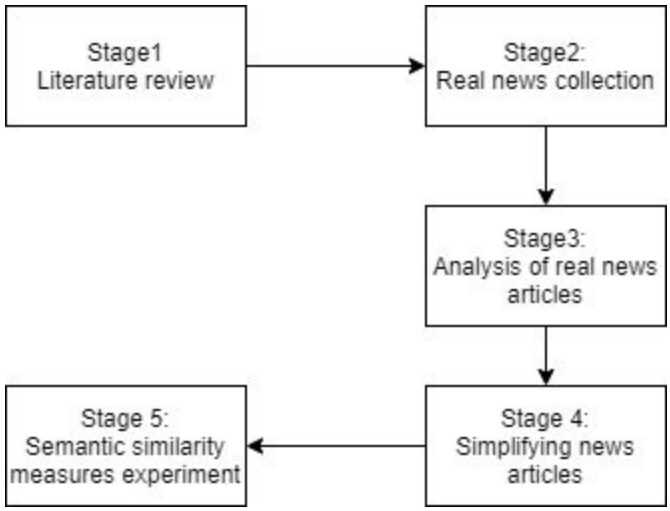

Fig 1.1 : Figure showing various stages of the project

One of the major problems in natural language understanding is complexity of language in a text. Because these articles all had different approaches to news writing, we concluded that applying the similarity metrics and generalizing results in such a diverse language representation would not be a good idea. Hence we decided to simplify the language representation as much as possible. The fourth stage of the project was focused on simplifying the real news article. In this stage, we also tried to generate fake news from those real news by changing some words in the text data. In the first three articles, we simply inverted a positive word to a negative word to convert a real news into fake news. For example, the third article states "*One of the primary issues that the U.S. federal and state governments were wrestling with during the COVID-19 coronavirus disease pandemic in May 2020 was the trade-off of keeping social distancing and*

business closure restrictions in place to protect lives, versus the trade-off of ongoing (and possibly permanent) economic harm to individuals, businesses, and the country as a whole." This story was converted to a false story by simply changing the underlined words to a "*were not wrestling* ".

Similarly, in the last four articles, we tried changing the structure of the article and converting it into a false story but by preserving the context of the article. For example, the sixth article talked about a $3.7 million dollar grant to the Wuhan Institute of Virology by the President Obama administration, from where the origin of coronavirus is speculated. We added our own additional information about President Obama giving direct instruction to release the virus to sabotage Donald Trump presidency which is false. The converted false story is as follows.
"*In April 2020, there were reports that the National Institute of Health(NIH) provided a grant of 3.7 million dollars to Wuhan Institute of Virology under direct order of President Barack Obama while he was in office. The reports also state that President Obama gave clear instructions to leak a deadly virus to cause problems for the upcoming president Donald Trump. These reports support rumors that the novel coronavirus has escaped from this virology lab.*"

The fifth stage of this research was the actual experiment on application of semantic similarity measures on these data. We used python programming language and google colab as our workspace. We mainly focused on three similarity metrics: *Jaccardian similarity, Cosine similarity and Soft-cosine similarity.* Let's discuss them in detail first.

Jaccard's Similarity:

Jaccard Similarity (coefficient), a term coined by Paul Jaccard, measures similarities between sets. Jaccardian similarity can be calculated for two representations: *set based representation* and *vector based representation.* It is defined as the size of the intersection divided by the size of the union of two data. It compares members for two datasets to see which members are shared and which are distinct. It's a measure of similarity for the two sets of data, with a range from 0% to 100%. The higher the percentage, the more similar the two populations. Although it's easy to interpret, it is extremely sensitive to small sample sizes and may give erroneous results, especially with very small samples or data sets with missing observations.

Let's see an example of Jaccard similarity in set based representation. Consider following two sentences:

A= {Harry is a student}
B= {He is a school going student}

The mathematical representation of Jaccard Similarity of above two set is:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$

Now, $|A \cap B| = |\{ is, a, student \}| = 3$ and $|A \cup B| = |\{ Harry, is, a, student, He, school, going \}| = 7$.

Thus, $J(A,B) = |A \cap B| / |A \cup B| = 3/7 = 0.43$

Let's also analyze the same example in terms of vector representation

Consider two sets x={x1, x2, …., xn} and y={y1, y2,....., yn} in a vector space representation. Then the Jaccard coefficient between them is:

$$J_W(\mathbf{x}, \mathbf{y}) = \frac{\sum_i \min(x_i, y_i)}{\sum_i \max(x_i, y_i)}$$

Let us consider the above two sentences A and B. In vector based representation, we represent these two sentences in a Bag of Words(BoW) model. In the bag of words approach, all the words in complete data are represented in the columns and the sentences or documents are represented in the rows. Now, we begin to count the occurrence of each word in all of the document/dataset and fill those values. The following table shows the bag of words representation of our two example sentences.

A= { Harry is a student } , B= { He is a school going student}

	Harry	is	a	school	going	student	he
A	1	1	1	0	0	1	0
B	0	1	1	1	1	1	1

From the above formula, the numerator is calculated by the sum of minimum value in each column. This results in the numerator = (0+1+0+0+1+0) = 3. Similarly the denominator is calculated by the sum of maximum value in each column. This results to denominator = (1+1+1+1+1+1+1+ = 7

Thus *Jaccard Similarity = numerator/denominator = 3/7 = 0.43*

Jaccard similarity is useful with only a small volume of data. Consider a dataset with thousands of documents. Representing a bag of words model for such a large volume of data might not be practical. Also, Jaccard similarity fails when we have a similar sentence but with a different set of words. Consider following sentences:

A = { COVID-19 vaccine was successfully tested by Russia }
B = { Human trial of coronavirus antidote is now completed }

These two sentences infer similar meaning although the information in sentence B might not be a complete one. But if we analyze similarity among these sentences, the Jaccardian similarity would result to 0 because there is no common words between sentence A and B.

$| A \cap B | = | \{ \ \} | = 0$

Thus Jaccard's Coefficient (A, B) = 0.

Cosine similarity

Cosine similarity is a metric used to determine how similar the documents are irrespective of their size. Mathematically, it measures the cosine of the angle between two vectors projected in a multi-dimensional space. In this context, the two vectors I am talking about are arrays containing the word counts of two documents. As a similarity metric, how does cosine similarity differ from the number of common words? When plotted on a multi-dimensional space, where each dimension corresponds to a word in the document, the cosine similarity captures the orientation (the angle) of the documents and not the magnitude. If you want the magnitude, compute the Euclidean distance instead. The cosine similarity is advantageous because even if the two similar documents are far apart by the Euclidean distance because of the size they could still have a smaller angle between them. Smaller the angle, higher the similarity.

Let us consider two sets x={x1, x2, …., xn} and y={y1, y2,....., yn} in a vector space representation. The cosine similarity between X and Y is:

$$\text{cosine}(X,Y)=\frac{X.Y}{\|X\|\,\|Y\|}$$

Where $X.Y=\sum_{k=1}^{n}(X_k \times Y_k)$ and L^p norm: $|X|_p=\left(\sum_{i=1}^{n}|X_i|^p\right)^{1/p}$

Let us see how cosine similarity works. Consider 3 news articles with 5 extracted features (f1-f5). Let us again represent these articles and features in a bag of words models. Let's say we achieve following bag of words model as shown in the table below:

	Feature 1	Feature 2	Feature 3	Feature 4	Feature 5
X	10	4	6	3	5
Y	3	7	5	2	9
Z	4	2	3	8	2

Suppose we want to analyze which document among Y and Z are most common to X. First we calculate cosine distance between X and Y. From the formula above, the numerator is calculated as a dot product of X and Y with each feature.

*X . Y = 10*3 + 4*7 + 6*5 + 3*2 + 5*9 = 30+28+30+6+45 = 139*

In case of cosine similarity, we take L2 normalization. At first the sum of squares of each feature count is taken. Later, the square root of that sum is calculated and the value is achieved. We calculate the L2 norm for both X and Y.

|| X || = [(10^2)+(4^2)+(6^2)+(3^2)+(5^2)]^(½) = (100+16+36+9+25)^(½) = 13.63

|| Y || =[(3^2)+(7^2)+(5^2)+(2^2)+(9^2)]^(½) = (9+49+25+4+81)^(½) = 12.96

Finally, the cosine distance between X and Y is calculated using the above values and following formula.

*Cosine distance(X,Y) = (X.Y)/(||X|| ||Y|) = 139/(13.63*12.96) = 0.81*

In a similar way, we also calculate cosine distance between X and Z.

*X . Z = 10*4 + 4*2 + 6*3 + 3*8 + 5*2 = 40+8+18+24+10 = 100*

|| X || = [(10^2)+(4^2)+(6^2)+(3^2)+(5^2)]^(½) = (100+16+36+9+25)^(½) = 13.63

|| Z || =[(4^2)+(2^2)+(3^2)+(8^2)+(2^2)]^(½) = (16+4+9+64+4)^(½) = 9.84

*Cosine distance(X,Y) = (X.Y)/(||X|| ||Y|) = 100/(13.63*9.84) = 0.74*

Hence we obtain cosine distance between X and Y to be 0.81 and cosine distance between X and Z to be 0.74. We can see that the cosine angle of separation between X and Y is smaller than that of X and Z. Hence document X is more similar to document Y than document Z.

SoftCosine similarity:

A soft cosine or ("soft" similarity) between two vectors considers similarities between pairs of features. The traditional cosine similarity considers the vector space model (VSM) features as independent or completely different, while the soft cosine measure proposes considering the similarity of features in VSM, which help generalize the concept of cosine (and soft cosine) as well as the idea of (soft) similarity.

For example, in the field of natural language processing (NLP) the similarity among features is quite intuitive. Features such as words, *n*-grams, or syntactic *n*-grams can be quite similar, though formally they are considered as different features in the VSM. For example, words “play” and “game” are different words and thus mapped to different points in VSM; yet they are semantically related. In case of *n*-grams or syntactic *n*-grams, Levenshtein distance can be applied (in fact, Levenshtein distance can be applied to words as well).

For calculating soft cosine, the matrix **s** is used to indicate similarity between features. It can be calculated through Levenshtein distance, WordNet similarity, or other similarity measures. Then we just multiply by this matrix.

Given two *N*-dimension vectors *a* and *b,* the soft cosine similarity is calculated as follows:

$$\text{soft\_cosine}_1(a, b) = \frac{\sum_{i,j}^{N} s_{ij} a_i b_j}{\sqrt{\sum_{i,j}^{N} s_{ij} a_i a_j} \sqrt{\sum_{i,j}^{N} s_{ij} b_i b_j}};$$

where, $s_{ij} = \text{similarity}(\text{feature}_i, \text{feature}_j)$. If there is no similarity between features, $(s_{ii} = 1, s_{ij} = 0 \text{ for } i \neq j)$, The above equation is equivalent to the conventional cosine similarity formula.

This measure is useful in clustering related topics in a dataset. Suppose there were 3 different documents with features such as “COVID”, “Coronavirus” or “novel coronavirus”. Initially cosine similarity, considered each of these features as independent features, but with soft cosine we have some measure of similarity between these features. The documents consisting these features might be now similar to one another rather than with documents consisting features like “flu”, “cancer” or some other disease.

Above mentioned similarity measures were implemented in our actual calculation.

CHAPTER 4
EXPERIMENTS AND RESULTS

We implemented all the similarity index measures discussed in the previous chapter with our dataset. The complete set of real and fake news data are given at the end of this report. Let us look at the experiments and result one by one

Jaccard's similarity:

A real news article was considered as a string along with the generated fake news articles related to the real news.

document_1_0 = "Circulating on social networks a video that shows an excerpt from a Spanish television show, supposedly issued December 24, 2019, in which it appears a woman (who claims to be psychic) to make predictions. In this video, the woman describes a set of events that have been interpreted as a detailed forecast of Covid-19 pandemic that has hit the world. It is, however, a fake video, at least as regards the date of issue. The video has been being disseminated on the Internet with a date and not tampered with the real."
document_1_1 = "Circulating on social networks a video that shows an excerpt from a Spanish television show, supposedly issued December 24, 2019, in which it is not appearing a woman (who claims to be psychic) to make predictions. In this video, the woman does not describe a set of events that have not been interpreted as a detailed forecast of Covid-19 pandemic that has not hit the world record. It is, however, a fake video, at least as regards the date of issue. The video has not been disseminated on the Internet with a date and not tampered with the real news."

A tokenizer was defined using lambda function as follows:

```
tokenize = lambda doc: doc.lower().split(" ")
```

The tokenizer split each word in a string identified by a blank space and also converted all uppercase letters to lowercase so as to maintain consistency in features. This tokenizer was applied to both sets of string defined in the program.

```
tokenized_documents_1_0 = [tokenize(document_1_0) for d in all_documents]
tokenized_documents_1_1 = [tokenize(document_1_1) for d in all_documents]
```

A function for calculating jaccard's similarity was then defined as per the formula mentioned earlier:

```
def jaccard_similarity(query, document):
    intersection = set(query).intersection(set(document))
    union = set(query).union(set(document))
    return len(intersection)/len(union)
```

The similarity index was then achieved by passing two tokenized documents. Following table represents the result of Jaccardian distance between each set of real and fake news

Set	1	2	3	4	5	6	7
Similarity	0.81	0.90	0.97	0.35	0.27	0.39	0.34

We see that similarity between news articles in set 1, 2 and 3 are very high. This is because news articles in these sets consist of high common features as the words in these articles are not changed and are simply inverted. But in case of article 4, 5, 6, 7, the words are changed along with the meaning of the sentences preserving the context. So we have less number of common features which lead to higher lower value of similarity measures among them.

Let us also analyze Jaccard's distance between real news of set 1 and real news of set 2 to see if this is a good measure for classification.

```
all_documents = [document_1_0, document_2_0]

tokenized_documents = [tokenize(d) for d in all_documents] # tokenized docs
all_tokens_set = set([item for sublist in tokenized_documents for item in sublist])

jaccard_similarity(tokenized_documents[0],tokenized_documents[1])

>>>>>0.09345794392523364
```

We obtain Jaccardian distance value 0.093. We can see that similarity between the same class of news is very low with regard to similarity between different classes of news. This Jaccardian similarity is not a good measure for text classification.

Cosine similarity

Here CountVectorizer is used to create document term matrices along with stopword removal. Stopwords can also be referred to as high frequency n-grams which have no significant contribution to feature engineering.

```
from sklearn.feature_extraction.text import CountVectorizer
import pandas as pd

# Create the Document Term Matrix
count_vectorizer = CountVectorizer(stop_words='english')
```

Now the represented set of real and fake news is combined to form a list and a bag of words model is created by passing the list through Countvectorizer.

```
documents = [document_1_0, document_1_1]
sparse_matrix = count_vectorizer.fit_transform(documents)
```

Now based on the sparse matrix and the data frame created from *documents*, cosine similarity is calculated.

```
df = pd.DataFrame(doc_term_matrix,
                  columns=count_vectorizer.get_feature_names(),
                  index=['true', 'false'])
# Compute Cosine Similarity
from sklearn.metrics.pairwise import cosine_similarity
print(cosine_similarity(df, df))
```

The result of cosine similarity obtained from various set is given in the table below:

Set	1	2	3	4	5	6	7

Similarity	0.91	0.86	0.98	0.79	0.71	0.73	0.62

We see a similar trend of results in the first three news sets, because the features in them are very common. If we see the result in the last four articles, even though the features are not so common as in the first three articles but the machine can sense some similarity in the context of the articles due to which, the similarity measure using cosine similarity is higher than that of Jaccardian distance.

Let's also calculate similarity between real news of set 4, 5, 6 and 7

```
documents = [document_4_0, document_5_0, document_6_0, document_7_0]
# Create the Document Term Matrix
count_vectorizer = CountVectorizer(stop_words='english')
count_vectorizer = CountVectorizer()
sparse_matrix = count_vectorizer.fit_transform(documents)

# OPTIONAL: Convert Sparse Matrix to Pandas Dataframe if you want to see the word frequencies.
doc_term_matrix = sparse_matrix.todense()
df = pd.DataFrame(doc_term_matrix,
                  columns=count_vectorizer.get_feature_names(),
                  index=['Set 4', 'Set 5', 'Set 6','Set 7'])
# Compute Cosine Similarity
from sklearn.metrics.pairwise import cosine_similarity
print(cosine_similarity(df, df))
```

```
>>>>>
[[1.         0.55025685 0.43467788 0.35961785]
 [0.55025685 1.         0.38752486 0.31582201]
 [0.43467788 0.38752486 1.         0.38250284]
 [0.35961785 0.31582201 0.38250284 1.        ]]
```

The cosine similarity between real news of set 4 and 5 is 0.55 , set 4 and 6 is 0.43, set 4 and 7 is 0.35. Similarly, the cosine similarity between real news of set 5 and 6 is 0

38 and set 5 and 7 is 0.38. Finally the similarity between real news of set 6 and 7 is 0.38

Let us also calculate cosine similarity between real news of set 4 with fake news of set 5, 6 and 7.

```
documents = [document_4_0, document_5_1, document_6_1, document_7_1]
# Create the Document Term Matrix
count_vectorizer = CountVectorizer(stop_words='english')
count_vectorizer = CountVectorizer()
sparse_matrix = count_vectorizer.fit_transform(documents)

# OPTIONAL: Convert Sparse Matrix to Pandas Dataframe if you want to see the word frequencies.
doc_term_matrix = sparse_matrix.todense()
df = pd.DataFrame(doc_term_matrix,
                  columns=count_vectorizer.get_feature_names(),
                  index=['Set 4', 'Set 5', 'Set 6','Set 7'])
# Compute Cosine Similarity
from sklearn.metrics.pairwise import cosine_similarity
print(cosine_similarity(df, df))
>>>>>>
[[1.         0.61857347 0.45259309 0.55249671]
 [0.61857347 1.         0.44449298 0.53149684]
 [0.45259309 0.44449298 1.         0.43816624]
 [0.55249671 0.53149684 0.43816624 1.        ]]
```

The cosine similarity between real news of set 4 and fake news of set 5 is 0.61, real news of set 4 with fake news set 6 is 0.45 and real news of set 4 with fake news of set 7 is 0.55.

This result is far better than the one with Jaccardian distance, which means with cosine similarity we are partially able to cluster our real news. But there are still some high similarities between real news and fake news of certain sets.

Soft Cosine similarity

Gensim library was used during the project for preprocessing documents, building word embeddings and also for sofcosine calculation using its *matutils* tool.

```
import gensim
documents = [document_1_0, document_1_1]
from gensim.matutils import softcossim
from gensim import corpora
import gensim.downloader as api
from gensim.utils import simple_preprocess
```

To get word vectors, we needed a word embedding model. We downloaded the FastText model using the gensim downloader api.

```
# Download the FastText model
fasttext_model300 = api.load('fasttext-wiki-news-subwords-300')
```

We needed a dictionary (a map of words to unique id), the corpus (word counts) for each document and similarity matrix.

```
# Prepare a dictionary and a corpus.
dictionary = corpora.Dictionary([simple_preprocess(doc) for doc in documents])

# Prepare the similarity matrix
similarity_matrix = fasttext_model300.similarity_matrix(dictionary, tfidf=None, threshold=0.0, exponent=2.0, nonzero_limit=100)

# Convert the sentences into bag-of-words vectors.
sent_1 = dictionary.doc2bow(simple_preprocess(document_1_0))
sent_2 = dictionary.doc2bow(simple_preprocess(document_1_1))
```

The similarity matrix consisted of a similarity index among each feature. If the total number of features was n(f1…...fn), the similarity matrix consisted of values in the following format.

(f1,f1) 1
(f1,f2) 0.6

(f1,f3) 0.02
.
.
.
.
(f1,fn) 0.32
(f2,f1) 0.4
.
.
.
(fn,fn) 1

This was the main difference between cosine and soft cosine. The soft cosine similarity also included similarity between features. If the similarity between all the features were 1, the soft cosine would behave as the cosine similarity approach.

Let us look at the soft cosine similarity values between real and fake news of different sets in the following table.

Set	1	2	3	4	5	6	7
Similarity	0.99	0.98	0.99	0.89	0.94	0.96	0.95

Clearly, soft cosine approach cannot directly be applied for classification. But what it provides is a similarity score between the extracted features. Now these extracted features can be extended with clustering algorithms to group particular features within a particular class and can be used as one of the criteria for classification.

Let's say we have a corpora with features f1…..fn. Let clustering algorithms cluster these features into 2 distinct sets: real and fake. Consider a test document with x features obtained during feature extraction. Now based on the highest probability and scoring of these extracted features, we can classify if our news article lies in a real news set or fake news set.

CHAPTER 5
CONCLUSION

In the literature review in Chapter 2, we saw researchers applying machine learning models for classification even with small volumes of text data. Text data collected from different sources may not resemble any pattern or the amount of data might not be sufficient to generalize vast amounts of patterns in any data. This would result in machine learning models guessing the result of classification. In such cases, machine learning models for classification clearly fail. General approach to data preprocessing is first required while working with a small set of data. Like mentioned in the previous chapters, if the dataset is first generalized with a common writing style, machine learning models could focus on learning parameters rather than struggling to understand writing patterns. Also, linguistic approaches such as semantic modelling and clustering could be one additional solution for text classification. Better preprocessing, proper understanding of linguistics, proper feature extraction could be a huge step forward for text classification rather than direct application of machine learning models.

REFERENCES

Zhou, X., & Zafarani, R. (2018). Fake news: A survey of research, detection methods, and opportunities. arXiv preprint arXiv:1812.00315.

Granik, M., & Mesyura, V. (2017). Fake news detection using naive bayes classifier. In 2017 ieee first ukraine conference on electrical and computer engineering (ukrcon) (pp. 900-903).

Gomaa, Wael & Fahmy, Aly. (2013). A Survey of Text Similarity Approaches. international journal of Computer Applications. 68. 10.5120/11638-7118.

AL-Smadi, Mohammad & Jaradat, Zain & Al-Ayyoub, Mahmoud & Jararweh, Yaser. (2017). Paraphrase identification and semantic text similarity analysis in Arabic news tweets using lexical, syntactic, and semantic features. Information Processing & Management. 53. 640-652. 10.1016/j.ipm.2017.01.002.

R. Chandrathlake, L. Ranathunga, S. Wijethunge, P. Wijerathne and D. Ishara, "A Semantic Similarity Measure Based News Posts Validation on Social Media," 2018 3rd International Conference on Information Technology Research (ICITR), Moratuwa, Sri Lanka, 2018, pp. 1-6, doi: 10.1109/ICITR.2018.8736136.

N. Shibata, Y. Kajikawa, I. Sakata, "How to measure the semantic similarities between scientific papers and patents in order to discover uncommercialized research fonts: A case study of solar cells", in Proceedings of PICMET technology management for global economic growth, Phuket, pp. 1-6, 2010

S. Zhang, X. Zheng, C. Hu, "A Survey of Semantic Similarity and its Application to Social Network Analysis", IEEE International Conference on Big Data (Big Data), pp. 2362-2367, 2015.

W.H. Gomaa and A. A. Fahmy, "A Survey of Text Similarity Approaches", International Journal of Computer Applications, Volume68- No.13, 2013.

Allcott, H., & Gentzkow, M. (2017). Social media and fake news in the 2016 election. Journal of economic perspectives, 31(2), 211-36.

Jin, Z., Cao, J., Guo, H., Zhang, Y., Wang, Y., & Luo, J. (2017, July). Detection and analysis of 2016 us presidential election related rumors on twitter. In International conference on social computing, behavioral-cultural modeling and prediction and behavior representation in modeling and simulation (pp. 14-24). Springer, Cham.

David MJ Lazer, Matthew A Baum, Yochai Benkler, Adam J Berinsky, Kelly M Greenhill, Filippo Menczer, Miriam J Metzger, Brendan Nyhan, Gordon Pennycook, David Rothschild, et al. 2018. The science of fake news. Science 359, 6380 (2018), 1094–1096.

Kai Shu, H Russell Bernard, and Huan Liu. 2018. Studying Fake News via Network Analysis: Detection and Mitigation. arXiv preprint arXiv:1804.10233 (2018).

Nguyen Vo and Kyumin Lee. 2018. The Rise of Guardians: Fact-checking URL Recommendation to Combat Fake News. arXiv preprint arXiv:1806.07516 (2018)

W. Xu, C. Callison-Burch, B. Dolan Semeval-2015 task 1: Paraphrase and semantic similarity in twitter (pit)Proceedings of the 9th international workshop on semantic evaluation (semeval 2015), Association for Computational Linguistics, Denver, Colorado (2015), pp. 1-11

*Biggins, S., Mohammed, S., Oakley, S., Stringer, L., Stevenson, M., & Preiss, J. (2012). University_of_Sheffield: two approaches to semantic text similarity. In * SEM 2012: The First Joint Conference on Lexical and Computational Semantics–Volume 1: Proceedings of the main conference and the shared task, and*

Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012) (pp. 655-661).