**Chandigarh College of Engineering and Technology**

**(Degree Wing)**

**Logo

Description automatically generated with medium confidence**

**Compiler Design (CS – 605C)**

**Final Project File**

**Submitted By Submitted To**

Dipesh Singla Dr. Varun Gupta

Roll: - CO19322 Professor CSE, HOD (A.S)

C.S.E 6th Semester

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| **S.No** | **Topics** | **Page No.** |
|  | Problem Introduction | **3** |
|  | **Project methodology** | **3** |
|  | **Flow of Work**  **3.1 ETL(Exploration, Transformation and Loading)**  **3.2 Feature Engineering**  **3.3 Model Definition and Training (Machine Learning Models)**  **3.4 Model Definition and Training (Deep Learning Algorithms)**  **3.5 Model Evaluations** | **4**  **4**  **4**  **4**  **4** |
|  | **Architectural Decisions/ Tools Used**  **4.1 Data Source**  **4.2 Enterprise Data**  **4.3 Data Repository**  **4.4 Discovery and Exploration**  **4.5 Actionable Insights**  **4.6 Applications / Data Products**  **4.7 Security, Information Governance and Systems Management** | **4**  **5**  **5**  **5**  **5**  **5**  **5**  **6** |
|  | **Final Results**  **5.1 Based on the Accuracy of the chosen model**  **5.2 Based on the F1 scores of the chosen model**  **5.3 Table of Comparison** |  |
|  | **Conclusion** | **9** |
|  | **References** | **9** |
|  | **Source Code** | **10-30** |

# Problem Introduction

News is being circulated in an ever-increasing social context, whether on social media platforms or the internet and that too in bulk. With so much news arriving from everywhere, it becomes difficult to check the source and validity of the news, that is, to distinguish between false and accurate news. This project provides an overview of news detection, which uses existing Machine Learning Models and Deep Learning Algorithms to determine whether the news is correct, i.e. authentic, and trustworthy.



# Why fake news is a problem?

Fake news is defined as misinformation, disinformation, or malicious information conveyed by word of mouth and conventional media, as well as, more recently, digital modes of communication such as manipulated videos, memes, unconfirmed adverts, and social media promoted rumours. Fake news on social media has become a severe concern, with the potential for it to lead to mob violence, suicides, and other negative outcomes as a result of disinformation spreading on social media.



# Keywords

True and Fake News, Data Mining, Twitter, Google News Vector, word2vec gensim model

# Project Methodology

Clement Bisaillon at [1] has donated the dataset used. It comprises two CSV files, one for bogus news called Fake.csv and the other for factual news called True.csv.

First, we displayed each of the data columns (such as Title, Subject, and Date) and examined how they are connected and what sort of problem we can deduct from that situation. And concluded that it is a binary classification problem in which a news item can be either phony or truthful. Using these representations, we vectorized the textual data in several ways and predicted it using the Logistic Regression Model, Decision Tree Classifier, and Artificial Neural Network. The Decision Tree Classifier surpasses other models in identifying false and real with an accuracy of 99.63 percent in the final model tests. As of now, a comprehensive solution for the use case is also attached.

# Flow of Work

The flow of work is as shown:

## ETL (Exploration, Transformation, and Loading)

* + 1. Analysis of Dates
    2. Analysis of Subjects
    3. Analysis of Title

Based on the length of the title

Based on the word count of the title

Visualizing the most used words (100) (for this we used df\_data\_1 and df\_data\_2)

* + 1. Analysis of Text
* Based on the length of text
* Based on word count of column text
* Visualizing the most used words (100) (for this we used df\_data\_1 and df\_data\_2)

## Feature Engineering

* + 1. Data Cleaning
    2. Pre-Processing
    3. Feature Creation
    4. Visualizations

## Model Definition and Training (Machine Learning Models)

* + 1. Using Logistic Regression (from scikit-learn) with 96D vector features provided by spaCy
    2. Using Decision Tree Classifier (from sci-kit-learn) with 96D vector features provided by spaCy
    3. Iteration/Approach 1: Using the above models mentioned with spaCy vectors as input features for predictions.
    4. Iteration/Approach 2: Using the above models mentioned by adding CountVectorizer and TFIDF Transformer from sklearn.feature\_extraction

## Model Definition and Training (Deep Learning Algorithms)

* + 1. Using Artificial Neural Network from a Sequential Model
    2. Iteration 2 uses more features with accuracy and F1-score as the primary metrics of evaluation.

## Model Evaluations

* + 1. Based on test accuracy
    2. Based on F1 scores

# Architectural Decisions/ Tools Used

## Brief description of the dataset

This dataset consists of about 40000 articles consisting of fake as well as real news. We aim to train our model so that it can correctly predict whether a given piece of news is real or fake. The fake and real news data are given in two separate datasets with each dataset consisting of around 20000 articles.

## Data Source

### **Technology Choice**:

### The data source makes data available in CSV format as Fake.csv and True.csv [1].

### **Justification:**

### It is simple to access, gather and explore such datasets, and they also include research on a variety of people, which provides us with the foundational approach for working on machine learning or deep learning challenges.

## Enterprise Data

### **Technology Choice:** Not required initially. We can use Lift for the enterprise data if required.

### **Justification:** The lift enables us to efficiently shift on-premises data to cloud databases. In our situation, we used a preset data set and verified that it worked in any similar real-world scenario.

## Data Repository

### **Technology Choice**: Cloud Storage (IBM)

### **Justification**: Cloud object storage allows you to store almost unlimited amounts of data. It's commonly utilized for scalable and long-term data archiving and backup, online and mobile apps, and analytics storage.

## Discovery and Exploration

### **Technology Choice:** Python, Jupyter, sci-kit-learn, pandas, Matplotlib, Seaborn

### **Justification**: Open source and supported in IBM Cloud are Jupyter, Python, sci-kit-learn, pandas, Matplotlib, and Seaborn. Some of these components have overlapping characteristics, while others have complementing characteristics.

## Actionable Insights

### **Technology Choice:** Python, pandas, nltk, and sci-kit-learn for Machine Learning Models and Keras and Tensorflow for Deep Learning Algorithm

### **Justification**: Python, pandas, nltk, and sci-kit-learn are great alternatives to R/R-Studio for data research. Python is also a cleaner and simpler programming language to learn than R. Pandas is Python's equivalent of R data frames, giving access to relational data. Nltk is a natural language processing toolkit that comes in handy when dealing with vast quantities of text. Finally, scikit-learn conveniently organizes all of the machine learning methods necessary. It is also available on IBM Cloud via IBM Watson Studio.

### Keras and TensorFlow are the most popular technologies for deep learning algorithms for framing neural networks. One of the most commonly used deep learning frameworks is TensorFlow. It is a linear algebra library that supports automated differentiation at its core. TensorFlow's Python-based syntax is rather complicated. As a result, Keras adds an abstraction layer to TensorFlow. Watson Studio and Watson Machine Learning on IBM Cloud provide seamless support for both frameworks.

## Applications / Data Products

### **Technology Choice**: Export of an already run, static Jupyter Notebook

### **Justification**: IBM Watson Studio's Jupyter Notebook is one of the simplest, fastest, and most efficient methods to provide a data product based on what we have worked on. It not only provides a fantastic platform for doing everything at once, but it also keeps everyone on the same page, whether it's data visualization, cleaning, pre-processing, training the model, or delivering the entire project.

## Security, Information Governance and Systems Management

### **Technology Choice**: Cloud Object Storage

### **Justification:** When it comes to contemporary, cost-effective cloud storage, Object Storage is the gold standard.

### **Proposed approach/ Methodology and Model**:

### In this work, in Phase-1 first of all we’ve cleaned the data, then a list of indices where the publisher is not mentioned, then Separating Publication info, the from the actual, text and The listing text column with new text there were 630 rows in fake news with empty text, then we created world cloud for both true and fake news and both were pretty different. In pre-processing text we first combined the title and text for real and fake news. Then in Phase-2 we used vectorization in Word2Vec, Word2Vec is one of the most popular technique to learn word embeddings using shallow neural network. Then we created Word2Vec model with genism. Then we feeded US presidents, etc to form some vectors. Further we tokenized the text and Checking the first 10 words of first news # every word has been represented with a number, made Histogram for no of words in news shows that most news article are under 700 words, and created a weight matrix. Then embedding vectors from word2vec and usings it as weights of non-trainable keras embedding layer and defined Neural Network. Then finally predict probability of news being real so converting into classes.

### Lastly we used pre-trained Word2Vec explored them and got the required accuracies and F-1 Scores.

### **Block Diagrams**:

### Block Diagram for fake news detection in Previous research works.

### **Framework Block-Diagram of Fake News detection with NLP and ML**.

### Diagram Description automatically generated

### **Figure 1:** Framework Block-Diagram of Fake News detection with NLP and ML [6].

### **Fake News detection using supervised Learning Algorithms**

### Fake news detection within online social media using supervised artificial intelligence algorithms - ScienceDirect

### **Figure 2:** Fake News detection using supervised Learning Algorithms [7].

### **MBi-LSTM method in the Fake news detection method**

### The block diagram of the proposed MBi-LSTM method in the Fake news detection method

### **Figure 3**: Block diagram of MBi-LSTM method in the Fake news detection method [8].

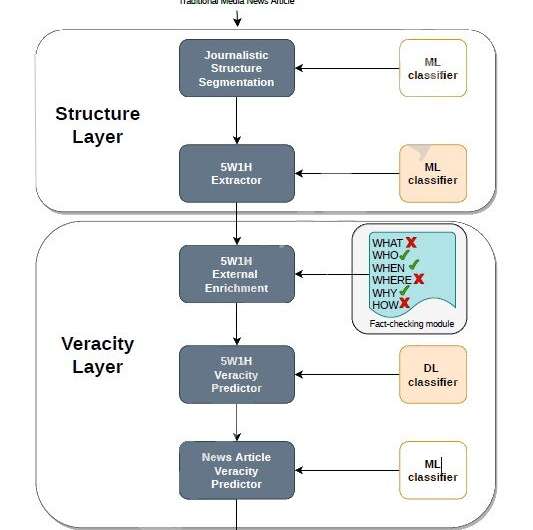
1. **Fake news detection approaches:**

**Diagram

Description automatically generated**

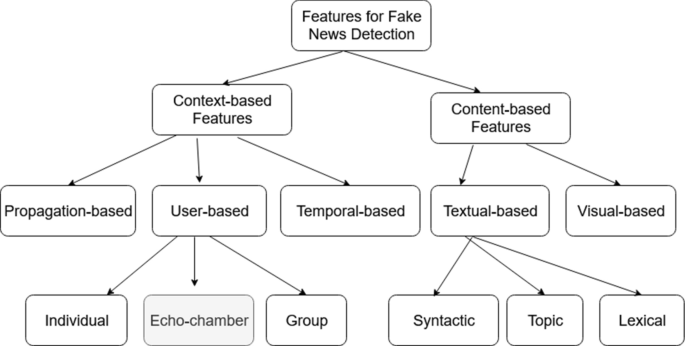
**Figure 4:** Fake news detection approaches [9]

1. **Development of a smart system for fake news detection**



**Figure 5:** Development of a smart system for fake news detection [10]

1. **EchoFakeD: improving fake news detection in social media with an efficient deep neural network**



**Figure 6:** Features of Fake news detection [11]

# Final Results

Along with a comparison from other top models Based on the chosen model's accuracy.

1. By selecting blogs as the solver and increasing the maximum number of iterations to 200, we can get an accuracy of 90.77 percent on the testing data for Logistic Regression with spaCy vectors as input features.
2. With spaCy vectors as input features, the decision tree classifier achieves an accuracy of 85.51 percent.
3. Logistic Regression with textual data as an input feature and Count Vectorizer and TFIDF Transformer in pipeline achieves 98.93% accuracy.
4. A Decision Tree Classifier using textual data as an input feature and Count Vectorizer and TFIDF Transformer in the pipeline achieves 99.63 percent accuracy.
5. Using 1000 input features, an artificial neural network constructed with Keras and TensorFlow backend with the Sequential Model and dense layers achieves an accuracy of 99.28 percent.
6. An Artificial Neural Network built using Keras and Tensorflow backends with the Sequential Model and dense layers achieves an accuracy of 99.21 percent on testing data with 10000 input features. Based on the F1 scores of the selected model, we can calculate the number of false positives and false negatives. Overall, we get an accuracy of 98.93%.
7. By selecting blogs as the solver and increasing the maximum number of iterations to 200, we can obtain a f1-score of 90.42 percent on the testing data for Logistic Regression with spaCy vectors as input features.
8. A decision tree classifier with spaCy vectors as input features achieves an f1-score of

84.49 percent.

1. Logistic Regression with textual data as an input feature and Count Vectorizer and TFIDF Transformer in the pipeline yields an f1 of 98.81%.
2. A Decision Tree Classifier using textual data as an input feature and Count Vectorizer and TFIDF Transformer in the pipeline produces an f1-score of 99.61 percent.
3. Keras with Tensorflow backend Artificial Neural Network with Sequential Model and Dense Layers supply us with an f1-score of 99.25 percent on the testing data with 1000 input features.
4. An artificial neural network built using Keras and Tensorflow backends with the Sequential Model and dense layers achieve a f1-score of 99.17 percent on testing data with 10000 input features. Also, an F1 score comparable to that of accuracy demonstrates how accurate our model is in forecasting

## Future Scope:

1. Gathering specific news-related tweets.
2. Using the same Decision Tree Classifier and pipeline, we will forecast whether or not this is false news.
3. Another suggestion is to utilize the same for the credibility score assignment of a certain tweet, which is slightly different from this.

## Table of Comparison

## Table of comparison of fake news detection and clean is shown in Table 1.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Model/ Classifier** | **Epochs/ Iterations** | **Batch Size** | **Input Dimensions** | **Test Set Accuracy** | **Test Set F1-Score** | **Other Parameters** |
| **Logistic Regression** | 200 |  | spaCy Vectorized Features | 90.77 | 90.42 | solver = lfbgs |
| **Logistic Regression** | 200 |  | Features from Count Vectorizer | 98.93 | 98.88 | solver = lfbgs |
| **Decision Tree Classifier** | - |  | spaCy Vectorized Features | 85.51 | 84.49 | criterion=entropy max\_depth = 10 splitter=best, random\_state=2020 |
| **Decision Tree Classifier** | - |  | Features from Count Vectorizer | **99.63** | 99.20 | criterion=entropy max\_depth =10 splitter=best random\_state=2020 |
| **Neural Network 1** | 7 | 512 | 1000 Features from Count Vectorizer | 99.19 | 99.16 | optimizer=adam loss=binary\_crosse ntropy |
| **Neural Network 2** | 7 | 512 | 10000 Features from Count Vectorizer | 99.24 | 99.12 | optimizer=adam loss=binary\_crosse ntropy |

# Conclusion

### Using Pre-Trained Word2Vec Vectors in a pipeline with we've achieved maximum accuracy, recall, and F1 score of about 99%, but there is always room for improvement. Various things, such as tweaking the hyperparameters, can be done in the future to improve the accuracy and f1 score based on another dataset. The use of tensors may enhance the overall results. Since the beginning of the project, the authors were enthralled while working, facing different challenges: may it be in terms of preprocessing the textual data, model training, or evaluation. Finally, the maximum accuracy and F1 score as per the models trained are achieved by using the Decision Tree Classifier in a pipeline with the Count Vectorizer and TFIDF Transformer, while with other models the team believes that there is always a scope of improvement accommodating various changes. Pretrained 300-D spacy model (en\_core\_web\_lg()) can be used which can take the models having a lower accuracy to the next level, on the contrary, it will require a huge amount of system resources to be compromised. Nonetheless, more hyperparameter tuning can be done in other prospects depending on another dataset using, this project will help us in our future research inin which collection of certain news-based tweets can be performed using twtwinsnd then using the same Decision Tree Classifier with the said pipeline, it can predict whether that is a piece of fake news or not. Apart from this, another idea is to use the same for the credibility score assignment of a particular tweet. As a whole, the project will help the contributors build models or research.

1. **References**
2. Dataset Link: https://www.kaggle.com/clmentbisaillon/fake-and-real-news-dataset
3. Ahmed H, Traore I, Saad S. “Detecting opinion spams and fake news using text classification”, Journal of Security and Privacy, Volume 1, Issue 1, Wiley, January/February 2018.
4. Ahmed H, Traore I, Saad S. “Detection of Online Fake News Using N-Gram Analysis and Machine Learning Techniques. In: Traore I., Woungang I., Awad A. (eds) Intelligent, Secure, and Dependable Systems in Distributed and Cloud Environments. ISDDC 2017. Lecture Notes in Computer Science, vol 10618. Springer, Cham (pp. 127-138).
5. Fake News Vs Real News Dataset
6. The lightweight IBM Cloud Garage Method for data science
7. Dataset Link: https://www.kaggle.com/clmentbisaillon/fake-and-real-news-dataset
8. Ahmed H, Traore I, Saad S. “Detecting opinion spams and fake news using text classification”, Journal of Security and Privacy, Volume 1, Issue 1, Wiley, January/February 2018.
9. Ahmed H, Traore I, Saad S. “Detection of Online Fake News Using N-Gram Analysis and Machine Learning Techniques. In: Traore I., Woungang I., Awad A. (eds) Intelligent, Secure, and Dependable Systems in Distributed and Cloud Environments. ISDDC 2017. Lecture Notes in Computer Science, vol 10618. Springer, Cham (pp. 127-138).
10. Fake News Vs Real News Dataset
11. The lightweight IBM Cloud Garage Method for data science
12. Ahmed H, Traore I, Saad S. “Detecting opinion spams and fake news using text classification”, Journal of Security and Privacy, Volume 1, Issue 1, Wiley, January/February 2018.
13. Ahmed H, Traore I, Saad S. (2017) “Detection of Online Fake News Using N-Gram Analysis and Machine Learning Techniques. In: Traore I., Woungang I., Awad A. (eds) Intelligent, Secure, and Dependable Systems in Distributed and Cloud Environments. ISDDC 2017. Lecture Notes in Computer Science, vol 10618. Springer, Cham (pp. 127-138).
14. Ahmed H, Traore I, Saad S. “Detecting opinion spams and fake news using text classification”, Journal of Security and Privacy, Volume 1, Issue 1, Wiley, January/February 2018.
15. Ahmed H, Traore I, Saad S. (2017) “Detection of Online Fake News Using N-Gram Analysis and Machine Learning Techniques. In: Traore I., Woungang I., Awad A. (eds) Intelligent, Secure, and Dependable Systems in Distributed and Cloud Environments. ISDDC 2017. Lecture Notes in Computer Science, vol 10618. Springer, Cham (pp. 127-138).

# Appendix:

**Source Code**

**Data Mining Lab:2019-2023 Batch**

**Semester - 6: Final Project: True vs Fake and Cleaning News Detection Name: Dipesh Singla Roll: CO19322**

**Keywords: True and Fake News, Data Mining, Twitter, Google News Vector, word2vec gensim model**

# WHY FAKE NEWS IS A PROBLEM?

## Fake news is defined as misinformation, disinformation, or mal-information that spreads by word of mouth, conventional media, and, more recently, digital modes of communication such as manipulated films, memes, unconfirmed adverts, and social media disseminated rumors. Fake news on social media has become a severe concern, with the potential for it to lead to mob violence, suicides, and other negative outcomes as, a result of disinformation propagated on social media.

**Motivation**

In this ever-increasing social world, we see lots of news circulating may it be on social media platforms or the internet across the globe. So there is a need to classify how can one be sure whether the news is true i.e. is genuine and can be trusted up by existing Machine Learning Models and Deep Learning Algorithms. We have also worked on research on how various social networking sites like Twitter make their algorithms work to get to know the facts about particular news which helped us carry this idea forward.

**Use Case:** To visualize, classify, predict and compare various models/ preprocessing algorithms to check whether the news is true or fake based on a given dataset.

## Data Sets

1. <https://www.kaggle.com/datasets/clmentbisaillon/fake-and-real-news-dataset>
2. <https://www.kaggle.com/datasets/sandreds/googlenewsvectorsnegative300>

# Library Imports

import warnings warnings.filterwarnings('ignore')

import numpy as np import pandas as pd

import matplotlib.pyplot as plt import seaborn as sns

import nltk import re

from wordcloud import WordCloudfrom tensorflow.keras.preprocessing.text import Tokenizer

from tensorflow.keras.preprocessing.sequence import pad\_sequences from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, Embedding, LSTM, Conv1D,

MaxPool1D

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import classification\_report, accuracy\_score

# Exploring Fake News

fake =

pd.read\_csv("/kaggle/input/fake-and-real-news-dataset/Fake.csv")

fake.head()

Table

Description automatically generated

# Counting by Subjects

**for** key,count **in** fake.subject.value\_counts().iteritems(): print(f"{key}:\t{count}")

*#Getting Total Rows*

print(f"Total Records:\t{fake.shape[0]}")

Text

Description automatically generated

plt.figure(figsize=(8,5)) sns.countplot("subject", data=fake) plt.show()

Chart, bar chart

Description automatically generated

# Word Cloud

text = ''

**for** news **in** fake.text.values: text += f" {news}"

wordcloud = WordCloud( width = 3000,

height = 2000, background\_color = 'black', stopwords =

set(nltk.corpus.stopwords.words("english"))).generate(text) fig = plt.figure(

figsize = (40, 30), facecolor = 'k', edgecolor = 'k')

plt.imshow(wordcloud, interpolation = 'bilinear') plt.axis('off')

plt.tight\_layout(pad=0) plt.show()

**del** text

A picture containing text, newspaper

Description automatically generated

# Exploring Real/ True News

real = pd.read\_csv("/kaggle/input/fake-and-real-news-dataset/True.csv")

real.head()

Table

Description automatically generated

**Difference in Text**

Real news seems to have source of publication which is not present in fake news set Looking at the data:

* most of text contains reuters information such as "**WASHINGTON (Reuters)**".
* Some text are tweets from Twitter
* Few text do not contain any publication info

# Cleaning Data

Removing Reuters or Twitter Tweet information from the text

* Text can be splitted only once at " - " which is always present after mentioning source of publication, this gives us publication part and text part
* If we do not get text part, this means publication details was't given for that record
* The Twitter tweets always have same source, a long text of max 259 characters

# Creating list of index that do not have publication part

unknown\_publishers = []

**for** index,row **in** enumerate(real.text.values):

**try**:

record = row.split(" -", maxsplit=1)

*#if no text part is present, following will give error*

record[1]

*#if len of piblication part is greater than 260*

*#following will give error, ensuring no text having "-" in between is counted*

**assert**(len(record[0]) < 260)

**except**:

unknown\_publishers.append(index)

# List of indices where publisher is not mentioned

real.iloc[unknown\_publishers].text

*#true, they do not have text like "WASHINGTON (Reuters)"*

Text

Description automatically generated

While looking at texts that do not contain publication info such as which reuter, we noticed one thing.

## Text at index 8970 is empty

real.iloc[8970]

Graphical user interface, text

Description automatically generated

# Seperating Publication info, from actual text

publisher = [] tmp\_text = []

**for** index,row **in** enumerate(real.text.values):

**if** index **in** unknown\_publishers:

*#Add unknown of publisher not mentioned*

tmp\_text.append(row)

publisher.append("Unknown")

**continue**

record = row.split(" -", maxsplit=1) publisher.append(record[0]) tmp\_text.append(record[1])

# Replace existing text column with new text

Add seperate column for publication info

real["publisher"] = publisher real["text"] = tmp\_text

**del** publisher, tmp\_text, record, unknown\_publishers

real.head()

Table

Description automatically generated with medium confidence

**Table

Description automatically generated with medium confidence**

**New column called "Publisher" has been added.**

# Checking for rows with empty text like row:8970

[index **for** index,text **in** enumerate(real.text.values) **if**

str(text).strip() == '']

*#dropping this record*

real = real.drop(8970, axis=0)

# Checking for the same in fake news

empty\_fake\_index = [index **for** index,text **in** enumerate(fake.text.values) **if** str(text).strip() == ''] print(f"No of empty rows: {len(empty\_fake\_index)}") fake.iloc[empty\_fake\_index].tail()

Table

Description automatically generated

## 630 Rows in Fake news with empty text

I've also observed a lot of CPATIAL-CASES in bogus news. Could keep letter cases, however we'll be utilising Google's pretrained word2vec vectors later on, which have well-formed lower case words. We will try to use lower case.

The text for these rows appears to be included in the title. Let's combine the title and text to solve these problems.

# Getting Total Rows

print(f"Total Records:\t{real.shape[0]}")

Graphical user interface, text, application, Word

Description automatically generated

***# Counting by Subjects***

**for** key,count **in** real.subject.value\_counts().iteritems(): print(f"{key}:\t{count}")

sns.countplot(x="subject", data=real) plt.show()

Chart, bar chart

Description automatically generated

# WordCloud For Real News

text = ''

**for** news **in** real.text.values: text += f" {news}"

wordcloud = WordCloud( width = 3000,

height = 2000, background\_color = 'black', stopwords =

set(nltk.corpus.stopwords.words("english"))).generate(str(text)) fig = plt.figure(

figsize = (40, 30), facecolor = 'k', edgecolor = 'k')

plt.imshow(wordcloud, interpolation = 'bilinear') plt.axis('off')

plt.tight\_layout(pad=0) plt.show()

**del** text

Text

Description automatically generated

**Preprocessing Text** **Adding class Information**

real["class"] = 1

fake["class"] = 0

# Combining Title and Text

real["text"] = real["title"] + " " + real["text"]

fake["text"] = fake["title"] + " " + fake["text"]

# Subject is Diffrent for real and fake => dropping

real = real.drop(["subject", "date","title", "publisher"], axis=1) fake = fake.drop(["subject", "date", "title"], axis=1)

# Combining both into new dataframe

data = real.append(fake, ignore\_index=True)

**del** real, fake

Removing StopWords, Punctuations and single-character words

# Converting X to format acceptable by gensim, removing annd punctuation stopwords in the process

y = data["class"].values X = []

stop\_words = set(nltk.corpus.stopwords.words("english")) tokenizer = nltk.tokenize.RegexpTokenizer(r'\w+')

**for** par **in** data["text"].values: tmp = []

sentences = nltk.sent\_tokenize(par)

**for** sent **in** sentences: sent = sent.lower()

tokens = tokenizer.tokenize(sent)

filtered\_words = [w.strip() **for** w **in** tokens **if** w **not in**

stop\_words **and** len(w) > 1]

tmp.extend(filtered\_words) X.append(tmp)

**del** data

**Vectorization -- Word2Vec**

Word2Vec is one of the most popular technique to learn word embeddings using shallow neural network. It was developed by Tomas Mikolov in 2013 at Google.

Word embedding is the most popular representation of document vocabulary. It is capable of capturing context of a word in a document, semantic and syntactic similarity, relation with other words, etc.

*Let's create and check our own Word2Vec model with* ***gensim***

import gensim

# Dimension of vectors we are generating

EMBEDDING\_DIM = 100

*#Creating Word Vectors by Word2Vec Method*

w2v\_model = gensim.models.Word2Vec(sentences=X, size=EMBEDDING\_DIM, window=5, min\_count=1)

# Vocab Size

len(w2v\_model.wv.vocab)

*# Represented each of 122248 words by a 100dim vector.*

**Exploring Vectors**

# See a sample vector for random word, say Corona

w2v\_model["corona"]

Table

Description automatically generated

w2v\_model.wv.most\_similar("iran")

Text

Description automatically generated

w2v\_model.wv.most\_similar("fbi")

Text, letter

Description automatically generated

w2v\_model.wv.most\_similar("facebook")

Text, letter

Description automatically generated

w2v\_model.wv.most\_similar("computer")

A screenshot of a computer

Description automatically generated with low confidence

# Feeding US Presidents

w2v\_model.wv.most\_similar(positive=["trump","obama", "clinton"])

*#First was Bush*

*Text, letter

Description automatically generated*

## Looking at the similar words, vectors are well formed for these words :)

These Vectors will be passed to LSTM/GRU instead of words. 1D-CNN can further be used to extract features from the vectors.

Keras has implementation called "**Embedding Layer**" which would create word embeddings(vectors). Since we did that with gensim's word2vec, we will load these vectors into embedding layer and make the layer non-trainable.

Tokenizer can represent each word by number. Since we cannot pass string words to embedding layer, thus need some way to represent each words by numbers.

**Tokenizing Text -> Repsesenting each word by a number**

**Mapping of orginal word to number is preserved in word\_index property of tokenizer**

**Tokenized applies basic processing like changing it yo lower case, explicitely setting that as False**

tokenizer = Tokenizer() tokenizer.fit\_on\_texts(X)

X = tokenizer.texts\_to\_sequences(X)

# Checking the first 10 words of first news # every word has been represented with a number

X[0][:10]

# Mapping is preserved in dictionary -> word\_index property of instance

word\_index = tokenizer.word\_index

**for** word, num **in** word\_index.items(): print(f"{word} -> {num}")

**if** num == 10:

**break**

**Text, letter

Description automatically generated**

## Notice it starts with 1

We can pass numerical representation of words into neural network.

We can use Many-To-One (Sequence-To-Word) Model of RNN, as we have many words in news as input and one output ie Probability of being Real.

For Many-To-One model, lets use a fixed size input.

# Histogram for no of words in news shows that most news article are under 700 words.

**Keeping each news small and truncate all news to 700 while tokenizing**

plt.hist([len(x) **for** x **in** X], bins=500)

plt.show()

*# Its heavily skewed.*

*# Truncate these outliers*

*Chart, histogram

Description automatically generated*

nos = np.array([len(x) **for** x **in** X]) len(nos[nos < 700])

*# Out of 48k news, 44k have less than 700 words*

# Keep all news to 700, add padding to news with less than 700 words and truncating long ones

maxlen = 700

*#Making all news of size maxlen defined above*

X = pad\_sequences(X, maxlen=maxlen)

#all news has 700 words (in numerical form now). If they had less words, they have been padded with 0

*# 0 is not associated to any word, as mapping of words started from 1 # 0 will also be used later, if unknows word is encountered in test set*

len(X[0])

*# Adding 1 because of reserved 0 index*

*# Embedding Layer creates one more vector for "UNKNOWN" words, or padded words (0s). This Vector is filled with zeros.*

*# Thus our vocab size inceeases by 1*

vocab\_size = len(tokenizer.word\_index) + 1

# Function to create weight matrix from word2vec gensim model

**def** get\_weight\_matrix(model, vocab):

*# total vocabulary size plus 0 for unknown words*

vocab\_size = len(vocab) + 1

*# define weight matrix dimensions with all 0*

weight\_matrix = np.zeros((vocab\_size, EMBEDDING\_DIM))

*# step vocab, store vectors using the Tokenizer's integer mapping*

**for** word, i **in** vocab.items(): weight\_matrix[i] = model[word]

**return** weight\_matrix

We Create a matrix of mapping between word-index and vectors. We use this as weights in embedding layer

Embedding layer accepts numecical-token of word and outputs corresponding vercor to inner layer.

It sends vector of zeros to next layer for unknown words which would be tokenized to 0.

Input length of Embedding Layer is the length of each news (700 now due to padding and truncating)

# Getting embedding vectors from word2vec and usings it as weights of non-trainable keras embedding layer

embedding\_vectors = get\_weight\_matrix(w2v\_model, word\_index)

# Defining Neural Network

model = Sequential()

*#Non-trainable embeddidng layer*

model.add(Embedding(vocab\_size, output\_dim=EMBEDDING\_DIM, weights=[embedding\_vectors], input\_length=maxlen, trainable=False)) *#LSTM*

model.add(LSTM(units=128)) model.add(Dense(1, activation='sigmoid'))

model.compile(optimizer='adam', loss='binary\_crossentropy', metrics=['acc'])

**del** embedding\_vectors model.summary() #Train

Table

Description automatically generated

test split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y)

model.fit(X\_train, y\_train, validation\_split=0.3, epochs=6)

Graphical user interface

Description automatically generated with low confidence

# Prediction is in probability of news being real, so converting into classes

*# Class 0 (Fake) if predicted prob < 0.5, else class 1 (Real)*

y\_pred = (model.predict(X\_test) >= 0.5).astype("int")

accuracy\_score(y\_test, y\_pred)



print(classification\_report(y\_test, y\_pred))

Table

Description automatically generated

**del** model

**Using Pre-Trained Word2Vec Vectors**

**Requirements RAM: 12GB and HardDisk Space: 4GB**

# Invoke garbage collector to free ram

import gc gc.collect()

from gensim.models.keyedvectors import KeyedVectors

*# Requires RAM*

word\_vectors = KeyedVectors.load\_word2vec\_format('../input/googlenewsvectorsnegative3 00/GoogleNews-vectors-negative300.bin', binary=True) EMBEDDING\_DIM=300

**Exploring these trained Vectors**

embedding\_matrix = np.zeros((vocab\_size, EMBEDDING\_DIM))

**for** word, i **in** word\_index.items():

**try**:

*# embedding\_vector = word\_vectors[word]*

embedding\_matrix[i] = embedding\_vector

**except** KeyError: embedding\_matrix[i]=np.random.normal(0,np.sqrt(0.25),EMBEDDING\_DIM)

*# del word\_vectors*

model = Sequential()

model.add(Embedding(vocab\_size, output\_dim=EMBEDDING\_DIM, weights=[embedding\_matrix], input\_length=maxlen, trainable=False)) model.add(Conv1D(activation='relu', filters=4, kernel\_size=4)) model.add(MaxPool1D())

model.add(LSTM(units=128)) model.add(Dense(1, activation='sigmoid'))

model.compile(optimizer='adam', loss='binary\_crossentropy', metrics=['acc'])

**del** embedding\_matrix model.summary()

Table

Description automatically generated

model.fit(X\_train, y\_train, validation\_split=0.3, epochs=12)

Table

Description automatically generated with low confidence

y\_pred = (model.predict(X\_test) > 0.5).astype("int")

accuracy\_score(y\_test, y\_pred)



print(classification\_report(y\_test, y\_pred))

Table

Description automatically generated with medium confidence

**Front- End Using Flask**:

**App.py**

from flask import Flask, render\_template, request, jsonify

import nltk

import pickle

from nltk.corpus import stopwords

import re

from nltk.stem.porter import PorterStemmer

app = Flask(\_\_name\_\_)

ps = PorterStemmer()

model = pickle.load(open('model2.pkl', 'rb'))

tfidfvect = pickle.load(open('tfidfvect2.pkl', 'rb'))

@app.route('/', methods=['GET'])

def home():

return render\_template('index.html')

def predict(text):

review = re.sub('[^a-zA-Z]', ' ', text)

review = review.lower()

review = review.split()

review = [ps.stem(word) for word in review if not word in stopwords.words('english')]

review = ' '.join(review)

review\_vect = tfidfvect.transform([review]).toarray()

prediction = 'FAKE' if model.predict(review\_vect) == 0 else 'REAL'

return prediction

@app.route('/', methods=['POST'])

def webapp():

text = request.form['text']

prediction = predict(text)

return render\_template('index.html', text=text, result=prediction)

@app.route('/predict/', methods=['GET','POST'])

def api():

text = request.args.get("text")

prediction = predict(text)

return jsonify(prediction=prediction)

if \_\_name\_\_ == "\_\_main\_\_":

app.run()