UNIVERSITY OF KWAZULU-NATAL ™

INYUVESI YAKWAZULU-NATALI

FAKE NEWS DETECTION

COMP316 – Natural Language Processing

Project Report – 2023

Dylan Govender – 221040222

Ayush Harrishun – 220002773

CONTENTS

INTRODUCTION

In recent years, the spread of fake news has raised some significant questions about the reliability of online information. With most people having access to the internet, this rapid dissemination of fake news is turbulent. The ability to automatically detect and classify fake news articles has become essential to facilitate informed decision-making and reduce the negative impact of misinformation. Combining Natural Language Processing (NLP) techniques with advanced machine learning models offers a promising solution to this problem. This project report uses a fine-tuned pre-trained BERT-BASED-UNCASED model, in conjunction, with a Convolutional Neural Network (CNN), to detect fake news, accurately and reliably.

AREA OF APPLICATION

Fake news detection using NLP techniques has a broad use in many fields of application. It can be deployed in social media platforms and news outlets to detect and curb the spread of misinformation, empowering users to make informed decisions based on trusted information. Governments and regulators can also use NLP-based fake news detection to monitor and stop the spread of disinformation during important public health events, elections, or crises. In addition, educational institutions and researchers can use these technologies to develop tools and resources that promote media literacy, and critical thinking, and help individuals distinguish between real and fake news.

PROBLEM

Fake news is the deliberate act of making fake or fabricated news look like real news. The problem with fake news is that it appears too real, and uses trickery and manipulation techniques, to make it look legit or genuine. Identifying fake news from real news is tasking, very time-consuming, and requires advanced techniques.

In an era of information overload and rapid proliferation of online content, the spread of fake news has become an urgent issue. Fake news can have dire consequences, leading to misinformation, and public distrust, and even affecting important decision-making processes. Fake news undermines the integrity of sources, undermines public trust, and undermines democratic processes. Additionally, the sheer volume of online content makes manual reviews impractical. Therefore, there is an urgent need for automated solutions that can effectively and efficiently detect fake news, combat the spread of misinformation, and mitigate its negative impact.

SOLUTION AND GOALS

We aim to accurately identify and classify misleading information from real news articles. We want to detect fake news and messages using NLP techniques, which have proven to be a promising approach, with intelligent applications that can automatically detect and flag misleading or fake messages, text, or news.

Applying NLP techniques to detect fake news provides a solution to the problem at hand. Leveraging machine learning algorithms and linguistic analysis, NLP models can effectively analyse and interpret text data to extract features and patterns that distinguish fake news from genuine articles. These models can learn from tagged datasets to identify linguistic cues such as misleading headlines, biased wording, and inconsistencies in the text. Through this approach, NLP-based systems can help automatically identify and flag potential fake news, helping users make more informed decisions and building a more trustworthy information ecosystem. We want our system to be efficient and least time-consuming. We decided to use a transfer learning model, fine-tuned BERT-BASE-UNCASED, and a deep learning model, Convolutional Neural Network (CNN), to tackle fake news.

Our main goal is to explore and analyse the fake or real news dataset and to build a classifier that can distinguish fake news with as much accuracy as possible.

RESOURCES

PROGRAMMING LANGUAGE

We decided to use Python for coding our project. It offers a rich ecosystem of libraries and tools designed specifically for NLP, including NLTK (Natural Language Toolkit), spaCy, and TensorFlow. Python's simplicity, readability, and extensive community support made it an excellent choice for both beginners, such as us, and experienced developers. Python was also introduced to us at the beginning of the Natural Language Processing (NLP) course and hence was easy to use and follow.

The second programming language we considered was Java. We are familiar with Java as it has been taught to us from the first year of our academic learning. Java is a bit more complex to use than Python, though Java is a versatile and widely used language with many libraries and frameworks for NLP such as Apache OpenNLP, Stanford NLP, and LingPipe. It is known for its performance, scalability, and platform independence.

INTEGRATED DEVELOPMENT ENVIRONMENT (IDE)

We decided to use Jupyter Notebooks with Google Collaboratory (Google Colab). Jupyter Notebook is a web-based interactive computing environment that allowed us to create and share documents containing code, visualisations, and explanatory text. It is commonly used for exploratory NLP work and data analysis. This IDE was easy to interact with and perform analysis on data, using any graphs and structures that were required. It also helped to save and download necessary files, such as our datasets and models. Importing all the necessary APIs, libraries and dependencies was hassle-free. Upgrading or downloading required dependencies was simple, and fast. These are just some of the advantages of working with Jupyter Notebooks. Using Jupyter Notebooks with Google Colab also abled us to switch between various run-time types such as GPU, CPU, or TPU which vastly improved the computation time of various tasks, such as training our machine learning models.

Other IDEs we considered were PyCharm, Eclipse and Visual Studio Code because we were familiar with those from the first year of our academic learning. We did not use those, because we found Jupyter Notebooks with Google Colab more user-interactive, easy to use, understandable, and easy to plan and manage code.

NLP TECHNIQUES

NLP TECHNIQUES USED

We decided to use **fine-tuned BERT-BASE-UNCASED** and **CNN**.

Fine-tuned BERT-BASE-UNCASED refers to a specific instance of a BERT (Transformers Bidirectional Encoder Representations) model that has been pre-trained on copious amounts of text data and tuned for a specific task such as fake news detection. BERT is a transformer-based model that uses a self-supervised learning approach to learn contextual representations of words within a given text. In BERT-BASED-UNCASED, "BASE" refers to the BERT's basic architecture, consisting of twelve transformers layers. "UNCASED" indicates that the model was trained to lowercase. That is, it is case insensitive. After pre-training on a large text corpus, BERT models are typically tuned for specific tasks through supervised learning. Fine-tuning trains the model on task-specific datasets containing labelled examples, such as datasets labelled with fake and real news articles. During fine-tuning, the model learns to adapt its pre-trained representation to the specific characteristics of the task, resulting in better performance on specific tasks.

A **CNN** (**Convolutional Neural Network**) for fake news detection is another type of model that uses convolutional layers commonly used in computer vision tasks. As part of fake news detection, CNN can be applied to text data by treating the text as a one-dimensional sequence of words or characters. CNN models apply convolutional operations to capture local patterns and features in the text. Filters that slide over the input text and perform convolutions are used to extract relevant features at different spatial locations. The resulting feature map is fed into fully connected layers for classification. The CNN fake news detection model is effective at capturing local patterns and identifying relevant features in the text. However, compared to models like BERT, CNNs may not be able to capture long-term dependencies or effectively capture contextual information. CNNs are more computationally efficient and can be trained on smaller datasets, making them suitable for certain scenarios.

NLP TECHNIQUES TRIED AND CONSIDERED

Recurrent Neural Network (RNN): RNNs such as Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) can be used for sequential data processing. Previously, we did try LSTM, but we decided to remove it as the results provided by the LSTM model were undesirable for such a copious amount of data. LSTM and RNN are particularly effective for analysing textual data because they can capture contextual information and inter-word dependencies within a sentence.

GPT (generative pre-trained transformer): GPT models such as GPT-2 and GPT-3 are based on the Transformer architecture and achieve outstanding performance in natural language processing. We can tune a pre-trained GPT model using the fake news detection dataset to improve its ability to detect misleading or inaccurate information. We did not use these, because of the complexity of understanding and incorporating such a model. Also, we had insufficient resources, such as the necessary GPU and CPU to train such a model.

Random Forests and Decision Trees: These are traditional machine-learning models that can be used for feature-based classification. By developing relevant functions from text, metadata, or other sources, we can train decision tree-based models like random forests to classify news articles as real or fake.

Support Vector Machine (SVM): SVMs are another class of traditional machine learning models that are effective at detecting fake news. By representing news articles as numerical feature vectors, we can train an SVM to learn decision boundaries between real and fake news examples.

Hybrid approach: We also considered a hybrid approach that combines different techniques, such as combining deep learning models with traditional machine learning models and rule-based systems. We previously tried Bi-LSTM, and CNN with a Bi-LSTM layer, called a CNN-Bi-LSTM model, these methods proved to be too complex, given that they provided the same desired results that we received from less complex models such as a CNN and LSTM. Also, our CNN-Bi-LSTM model was providing inaccurate results, factoring all these, we decided to rule out CNN-Bi-LSTM and use a normal CNN. These approaches allowed us to leverage the strengths of each method to create a more robust fake news detection system.

IMPLEMENTATION

APPLICATION PROGRAMMING INTERFACES (APIs)

Transformers: From the transformers API, we used the functions, BertTokenizer and BertTokenizerFast, which provided tokenization for our BERT model. We needed the BertForSequenceClassification module, which provided a pre-trained BERT model. The AdamW function implemented the AdamW optimizer used for fine-tuning our pre-trained BERT model. The get_linear_schedule_with_warmup function generated a learning rate schedule with a warm-up for fine-tuning and configuring our pre-trained model.

Sklearn (Sci-Kit-Learn): We used the Sklearn API for evaluating our models and using the metrics module, within this API, to provide the functionality for computing evaluation metrics when testing the models. The confusion_matrix function showed the true positive, false positive, true negative, and false negative values for our model's predictions. The classification_report function generated a comprehensive classification report, which showed the precision, recall, F1 score, and support metrics of our models.

PyTorch: It is a deep-learning framework. The utils.data function, within this framework, was used to provide the utilities for creating and working with data loaders and datasets. The data_loaders module, within this API, was used to fine-tune our pre-trained BERT model. We also used the Dataset module, which represented a dataset for fine-tuning and testing our BERT model. We used the pad_sequences function to pad sequences from our data frame or dataset to a specified length. We also used the DataLoader function to create an iterator for the efficient loading of data, during the fine-tuning and testing stages, of our pre-trained model.

NumPy: We used NumPy for numerical analysis and computing. It also provided support for large, multi-dimensional arrays and mathematical operations.

Pandas: We used pandas for data manipulation and analysis. It was also used when we were working with large collections of data, manipulating data from our datasets and data frames.

Seaborn: We used the Seaborn for data analysis and visualising data patterns. It provided a high-level interface for creating informative and visually appealing statistical graphics.

Matplotlib: We used Matplotlib for our bivariate analysis of data. Used to plot graphs and analytical structures.

Gensim: We used Gensim for text pre-processing, including tokenization, stopword removal, stemming, and lemmatization. Which provided the necessary cleaning of our dataset.

NLTK: Used for processing and pre-processing our data. We used stopwords from NLTK to remove common or unnecessary words from our datasets. We used Wordnet from NLTK to provide lexical and semantic data processing. We used Punkt from NLTK for tokenizing, padding, and splitting our data into sequences, sentences, or words.

TensorFlow: We used TensorFlow for providing the deep learning framework needed to configure and train our models. From TensorFlow we mostly used Keras, which runs on top of TensorFlow, to provide an interface for building, training, or fine-tuning our models. From Keras, we constructed our model using the various layers this API provides, such as Dense, Dropout, Conv1D, MaxPooling1D, and more.

INITIALISING THE DATASETS

We obtained our datasets from a Google website: ISOT Fake News Dataset.
https://onlineacademiccommunity.uvic.ca/isot/2022/11/27/fake-news-detection-datasets/
The datasets from this website contain two dataset files True.csv and Fake.csv. Which contains real news and fake news, separately. We downloaded the datasets and stored them in our Google Drive. We then created a direct download link for each dataset and imported them using the gdown API into the Google Colab directory.

Real News Dataset:

	title	text	subject	date
0	As U.S. budget fight looms, Republicans flip t...	WASHINGTON (Reuters) - The head of a conservat...	politicsNews	December 31, 2017
1	U.S. military to accept transgender recruits o...	WASHINGTON (Reuters) - Transgender people will...	politicsNews	December 29, 2017
2	Senior U.S. Republican senator: 'Let Mr. Muell...	WASHINGTON (Reuters) - The special counsel inv...	politicsNews	December 31, 2017
3	FBI Russia probe helped by Australian diplomat...	WASHINGTON (Reuters) - Trump campaign adviser ...	politicsNews	December 30, 2017
4	Trump wants Postal Service to charge 'much mor...	SEATTLE/WASHINGTON (Reuters) - President Donal...	politicsNews	December 29, 2017
...	...	...	...	...
21412	'Fully committed' NATO backs new U.S. approach...	BRUSSELS (Reuters) - NATO allies on Tuesday we...	worldnews	August 22, 2017
21413	LexisNexis withdrew two products from Chinese ...	LONDON (Reuters) - LexisNexis, a provider of l...	worldnews	August 22, 2017
21414	Minsk cultural hub becomes haven from authorities	MINSK (Reuters) - In the shadow of disused Sov...	worldnews	August 22, 2017
21415	Vatican upbeat on possibility of Pope Francis ...	MOSCOW (Reuters) - Vatican Secretary of State ...	worldnews	August 22, 2017
21416	Indonesia to buy $1.14 billion worth of Russia...	JAKARTA (Reuters) - Indonesia will buy 11 Sukh...	worldnews	August 22, 2017

21417 rows × 4 columns

Fake News Dataset:

	title	text	subject	date
0	Donald Trump Sends Out Embarrassing New Year'...	Donald Trump just couldn t wish all Americans ...	News	December 31, 2017
1	Drunk Bragging Trump Staffer Started Russian ...	House Intelligence Committee Chairman Devin Nu...	News	December 31, 2017
2	Sheriff David Clarke Becomes An Internet Joke...	On Friday, it was revealed that former Milwauk...	News	December 30, 2017
3	Trump Is So Obsessed He Even Has Obama's Name...	On Christmas day, Donald Trump announced that ...	News	December 29, 2017
4	Pope Francis Just Called Out Donald Trump Dur...	Pope Francis used his annual Christmas Day mes...	News	December 25, 2017
...	...	...	...	...
23476	McPain: John McCain Furious That Iran Treated ...	21st Century Wire says As 21WIRE reported earl...	Middle-east	January 16, 2016
23477	JUSTICE? Yahoo Settles E-mail Privacy Class-ac...	21st Century Wire says It s a familiar theme. ...	Middle-east	January 16, 2016
23478	Sunnistan: US and Allied 'Safe Zone' Plan to T...	Patrick Henningsen 21st Century WireRemember ...	Middle-east	January 15, 2016
23479	How to Blow $700 Million: Al Jazeera America F...	21st Century Wire says Al Jazeera America will...	Middle-east	January 14, 2016
23480	10 U.S. Navy Sailors Held by Iranian Military ...	21st Century Wire says As 21WIRE predicted in ...	Middle-east	January 12, 2016

23481 rows × 4 columns

DATASET PREPARATION

We added a column to each dataset, which is the label column. The label column will store the value one (1) if it is real news and zero (0) if it is fake news. Secondly, we combined both datasets, using the Pandas library, into one main dataset and then shuffled the dataset, using the shuffle function provided by the sklearn API, which we now call the df (data frame).

	title	text	subject	date	label
0	Trump considering Dr. Scott Gottlieb to head FDA	(Reuters) - Dr. Scott Gottlieb, a partner at o...	politicsNews	December 12, 2016	1
1	HOLLYWOOD 'HAS BEEN' HYPOCRITE Danny DeVito Te...	If you can t get an acting role in Hollywood, ...	left-news	Jan 23, 2016	0
2	WE GUESS SHE SHOWED 'EM! Lefty Woman Rips Off ...	We can t imagine congressmen thinking we d be...	left-news	Dec 19, 2017	0
3	U.S. welcomes Hariri's return to Lebanon: Stat...	WASHINGTON (Reuters) - The United States welco...	worldnews	November 22, 2017	1
4	HILLARY CLINTON CRASHING IN POLLS: Moves To Ob...	So, the working people of America are basicall...	politics	Aug 10, 2015	0
...	...	...	...	...	...
44893	Trevor Noah Tears Ben Carson Apart For Attack...	It s hard to believe that Ben Carson is a brai...	News	February 25, 2016	0
44894	NOT SO FUNNY GUY, WILL FERRELL Throws His Supp...	Supporting Hillary is about as funny as an ebo...	left-news	Feb 21, 2016	0
44895	No word Tuesday on Supreme Court nomination: W...	WASHINGTON (Reuters) - The Obama administratio...	politicsNews	March 15, 2016	1
44896	Trump Freaks Out, Talks 'World War Three' (VI...	Donald Trump, increasingly desperate as the el...	News	October 25, 2016	0
44897	Polls Show Millennials Hate Trump, Would Over...	The notion that millennials are stupid, spoile...	News	April 25, 2016	0

44898 rows × 5 columns

The main dataset attribute information consists of:

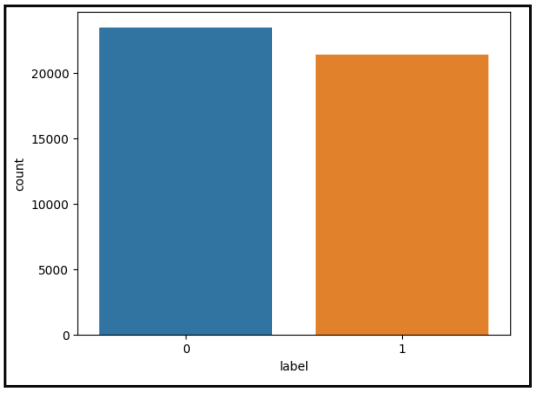
```
Shape of dataset: (44898, 5)
Dataset columns Index(['title', 'text', 'subject', 'date', 'label'], dtype='object')
```

We will remove text, subject, and date, as we do not need these columns. We will only train our models using the title and label attributes of the dataset.

DATA VISUALISATION PART I

FAKE AND REAL NEWS DISTRIBUTION:

We have slightly more fake news in the data frame. Our data frame consists of 52% fake news and 48% real news. This is not an entirely balanced dataset but can be considered as such.




```
Distribution of labels:
Fake = 0
Real = 1

Count each labels:
0    23481
1    21417
Name: label, dtype: int64

0    52.0
1    48.0
Name: label, dtype: float64
```

TITLE STATISTICS:

```
1 title_length = df.title.str.split().str.len()
2 title_length.describe()
```

```
count    44898.000000
mean        12.453472
std          4.111476
min          1.000000
25%         10.000000
50%         11.000000
75%         14.000000
max         42.000000
Name: title, dtype: float64
```

TEXT STATISTICS:

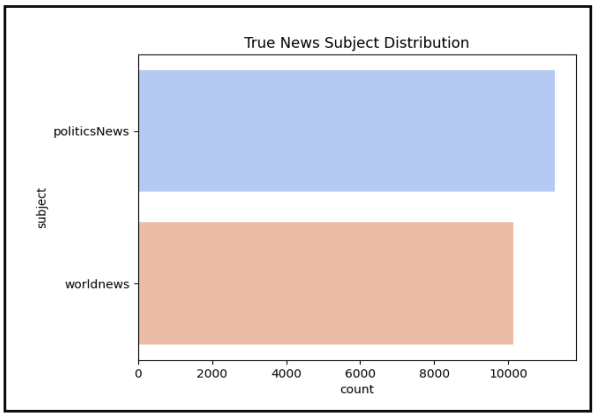
```
1 text_length = df.text.str.split().str.len()
2 text_length.describe()
```

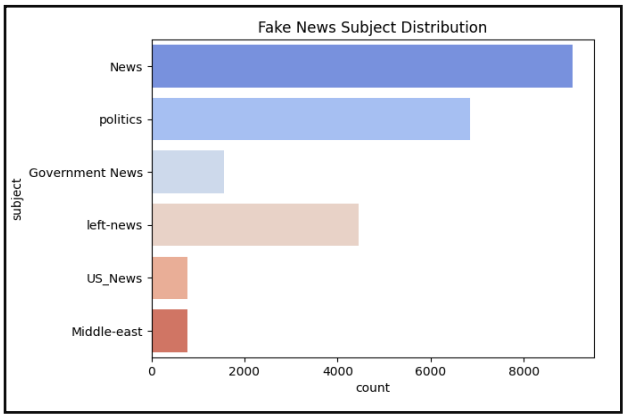
```
count    44898.000000
mean       405.282284
std        351.265595
min          0.000000
25%        203.000000
50%        362.000000
75%        513.000000
max       8135.000000
Name: text, dtype: float64
```

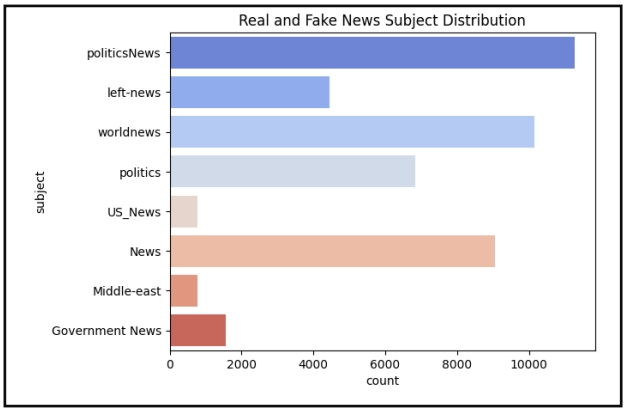
The title column is a short statement with an average of 12.45 words, and 75% of the entries are around fourteen (14) words. Hence, we will be using the title column to train our models, since it will reduce the computation time and training time of our models. The text column has a higher word count with an average of 405.28 words and 75% of the entries have more than 513 words.

SUBJECT DISTRIBUTION:

These subject distribution graphs show the distribution of the subjects within the dataset:







DATA CLEANING AND PREPROCESSING

We needed to download two dependencies that will help us to clean the data from the dataset and enable us to pre-process the data, which are, the NLTK stopwords, which contain a corpus of unnecessary or common words, and NLTK wordnet, used for lemmatization.

We used PorterStemmer to strip words from their morphological affixes. We also initialized a WordNetLemmatizer() which is better than stemming words, which evaluates a language's whole lexicon to apply morphological analysis to a word, to return the word's lemma. We also extended the list of stopwords and added a few more unnecessary words that will be irrelevant to our project results.

We first replaced any empty entries or null entries in the title column with "None". We used regex to remove unnecessary words, punctuation, numbers, and text. We also applied lowercasing to every word. We used the gensim API to perform pre-processing, which allowed us to remove stopwords, and remove words with two (2) or fewer characters. The gensim pre-processing was applied to a new column, called "clean" in the data frame, which returns the title column entries cleaned and in the form of a list. This pre-processing function is the main cleaning function that will affect our results in the training, testing, and evaluation of our models.

The second form of pre-processing we needed to accomplish is the NLTK pre-processing. This pre-processing makes the column entries clean but does not change their form into a list. This NLTK pre-processing function is unnecessary and only used for data analysis and visualisation. Since we only need to use the title and label attributes of the dataset to train our models, we removed the unnecessary columns from the dataset, such as the text, subject, and date columns.

Our dataset after pre-processing and cleaning:

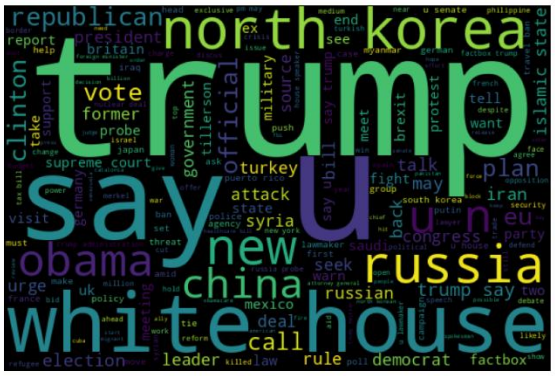
	title	label	clean
0	two item kill people year u	0	[item, kill, people, year]
1	senate leader mcconnell say wait replacing scalia	1	[senate, leader, mcconnell, wait, replacing, s...
2	australia tighten airport security foiled attack	1	[australia, tighten, airport, security, foiled...
3	trump say puerto rico trouble hurricane debt m...	1	[trump, puerto, rico, trouble, hurricane, debt...
4	india punjab haley relative cheer appointment ...	1	[india, punjab, haley, relative, cheer, appoin...
...	...	...	...
44893	make america mexico movement real organized	0	[america, mexico, movement, real, organized]
44894	brics name pakistan based militant group regio...	1	[brics, pakistan, based, militant, group, regi...
44895	ron paul blame obama stock market crash happen...	0	[paul, blame, obama, stock, market, crash, hap...
44896	russian custody tatar leader vow return crimea	1	[russian, custody, tatar, leader, return, crimea]
44897	clinton lead trump 2 point fox news poll	1	[clinton, lead, trump, point, news, poll]

44898 rows × 3 columns

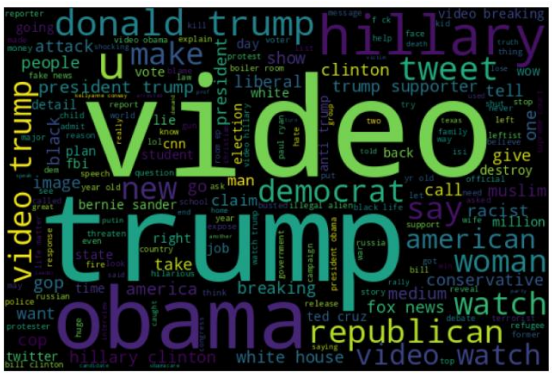
DATA VISUALISATION PART II

UNIVARIATE ANALYSIS: The statistical analysis of our data in the title field. We used wordclouds for the univariate analysis of our data. The bigger the word in the wordcloud, the more common or frequent it appears in the title entries.

REAL NEWS WORDCLOUD:

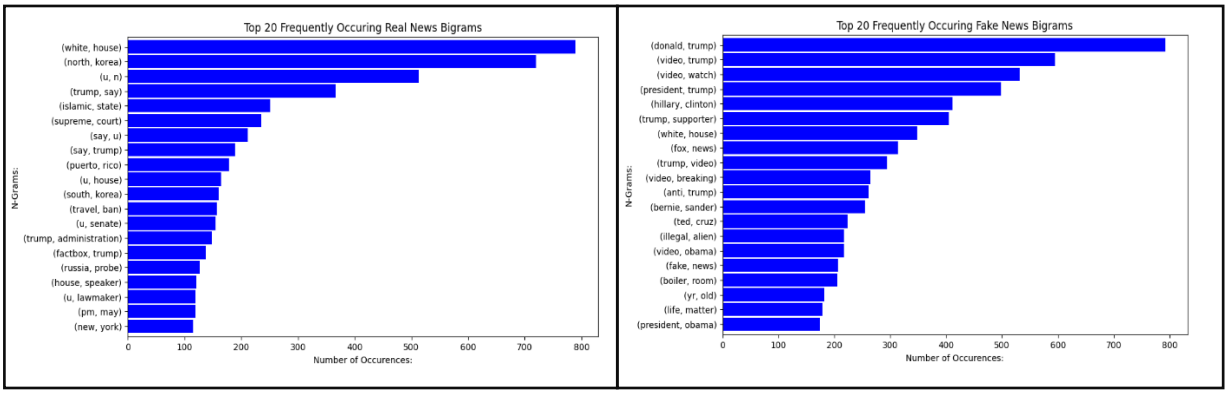


FAKE NEWS WORDCLOUD:

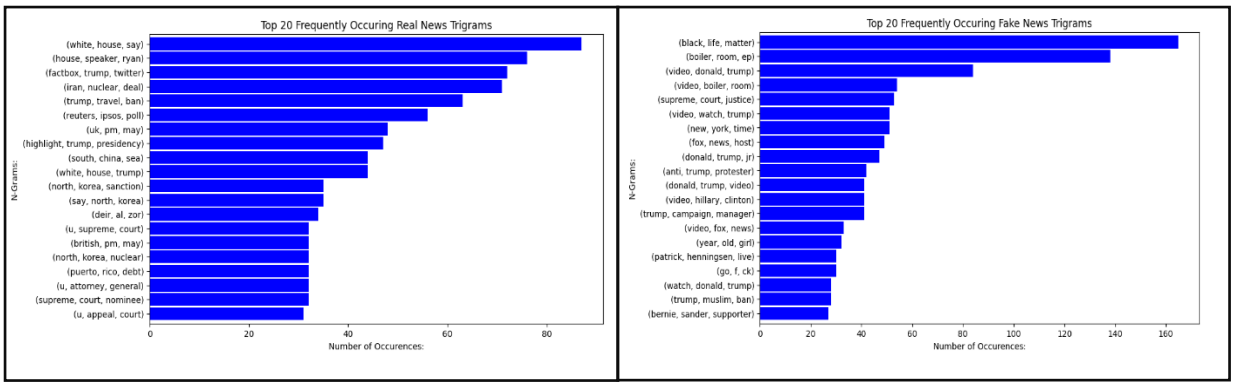


BIVARIATE ANALYSIS: We used n-grams to show the relationship between words. To analyse which sequence of words appeared the most.

BIGRAMS:



TRIGRAMS:



DATA PREPARATION

Due to the limited computational resources, we were working with, we had to split our data into three (3) sets: train set, valid set, and test set. By splitting the dataset into these three sets, we can train the model on the training set, fine-tune it on the validation set, and finally, assess its performance on the test set. This division allows us to have an accurate estimation of the model's performance on unseen data and helps prevent **overfitting**, where the model performs well on the training data but poorly on unseen data. This also lowers the amount of hardware or computational resources needed to accomplish our project. The train set was the largest dataset, because we wanted our models to be trained well, and be used on unseen data. The three dataset sizes and shapes were recorded:

```
Train set size: (28734, 3)
Validation set size: (7184, 3)
Test set size: (8980, 3)
```

We also needed to get the total number of tokens or words from the cleaned title column. We used the total number of words for padding and tokenization, so our data is in the right format to train our CNN model. We used the Punkt module provided by the NLTK API to achieve this.

CNN CONFIGURATION AND TRAINING

CNN CONFIGURATION

We first imported the required models or building blocks, used to construct our neural network model, from the Keras API. We initialised the dimensions (embedding_vector_features which is a hyper-parameter) of the embedding layer to forty (40), which determined the size of the word embeddings used in our CNN model. The other **hyper-parameters** we have are:

batch_size: Defines the number of times the model will iterate over the entire training dataset during the training process.

epochs: Specifies the number of times the model will iterate over the entire training dataset during the training process.

patience: Used in the EarlyStopping call-back function. Represents the number of epochs to wait before stopping the training if there is no improvement in the validation loss.

We initialised a Sequential model or an instance of the Sequential class. Using this feed-forward method allowed us to stack layers sequentially, one after the other. The output of one layer serves as the input for the next layer, forming a sequence of transformations. In this case, our sequential model will extract meaningful features that will classify fake and real news. The **layers** we have in our sequential model are:

Embedding layer: Converts the input text into a dense vector representation. Captures semantic relationships and contextual information of words in the text.

Dropout layer: Drops a fraction of the input units to 30% during training. Prevents overfitting by adding regularization. Improves generalization.

Convolutional Layer (Conv1D): Extracts local patterns and detects key features from the text. Analyses a one-dimensional sequence of text data. Uses an activation function ReLU.

Max Pooling Layer (MaxPool1D): Down-samples the output of the Conv1D layer. Reduces the dimensionality of the features whilst retaining the most prominent or salient features of the data.

Flatten Layer: Converts the output of the MaxPool1D layer into a flat, one-dimensional vector. Reshapes the output of the convolutional layers to make it compatible with the incoming dense layer.

Dense Layer: Connects every neuron from the previous layer to the current layer. Learns complex patterns and relationships between the features extracted from the previous layers. Uses an activation function, Sigmoid, which produces a probability value between 0 and 1.

CNN MODEL CONFIGURATION:

```
Model: "CNN"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 embedding_1 (Embedding)      (None, 35, 40)            660320

 dropout_2 (Dropout)          (None, 35, 40)            0

 conv1d_2 (Conv1D)            (None, 31, 32)            6432

 max_pooling1d_2 (MaxPooling  (None, 15, 32)            0
 1D)

 conv1d_3 (Conv1D)            (None, 11, 32)            5152

 max_pooling1d_3 (MaxPooling  (None, 5, 32)             0
 1D)

 flatten_1 (Flatten)          (None, 160)               0

 dropout_3 (Dropout)          (None, 160)               0

 dense_1 (Dense)              (None, 1)                 161

=================================================================
Total params: 672,065
Trainable params: 672,065
Non-trainable params: 0
_________________________________________________________________
None
```

CNN TRAINING

We imported the EarlyStopping call-back function from the TensorFlow Keras library. This function allows us to stop the training of our model when there is no improvement in the validation loss. It reduces computation time and is used to prevent overfitting. We first converted our train set and validation set of sequences to NumPy arrays. We shuffled the order of sequences in both arrays to prevent any sequence biases. We finally trained the model using the fit() function.

BERT CONFIGURATION AND TRAINING

BERT CONFIGURATION

Loading the tokenizer for BERT:
We first needed to instantiate the BertTokenizer, there are 2 types of BERT tokenizers that we can use, the normal BertTokenizer and the improved BertTokenizerFast, we decided to use the BertTokenizerFast to efficiently manage the little resources we had.

Pre-processing, tokenizing, and formatting:
We had to define a custom FakeNewsDataset class which is used for loading and pre-processing the split datasets into a suitable format for training and fine-tuning the BERT model. This class loads the datasets: train_set, valid_set, and test_set. Each dataset title column is then tokenized using the BERT tokenizer. It also converted the data into tensors making the data suitable for training the BERT model for fake news detection.

Padding, extracting tensors, and creating data loaders:
We also had to define a function create_mini_batch that is used for collating individual samples into a batch when creating data loaders for training or evaluating the model. This function takes a list of samples, individual data points in a dataset, and collates them into a batch by padding the sequences of tokens and segment tensors to have the same length. It also creates a mask tensor to identify the valid tokens in each sequence. This function is used when creating the data loaders to efficiently process and feed data in batches during the training or evaluation stage.

We have the following **hyper-parameters**:
BATCH_SIZE: Determines the number of samples in each mini-batch during training.
EPOCHS: Specifies the number of times the entire train_set is passed through during the training.
LEARNING_RATE: Controls the step size at each iteration of the optimization algorithm.
WARMUP_STEPS: Determines the number of warm-up steps needed for the learning rate scheduler.
EPSILON: Small value used for numerical stability in the AdamW optimizer. Prevents an indeterminate.
GRADIENT_ACCUMULATION_STEPS: Defines the number of batches for which gradients are accumulated before performing a parameter update. Reduces CPU usage.

Since we are fine-tuning a BERT model, we did not have enough resources to simulate higher-valued hyper-parameters. We used gradient accumulation to reduce our CPU and RAM usage, during training. We also reduced the BATCH_SIZE to the bare minimum, ideally, it should have been as high as possible.

```
Epoch 1/3 | Average Loss: 0.0526
Epoch 2/3 | Average Loss: 0.0374
Epoch 3/3 | Average Loss: 0.0366
```

We trained our model with EPOCHS = 3 and BATCH_SIZE = 8. We also trained our model with EPOCHS = 3 and BATCH_SIZE = 12. Link to the various models we trained, each model's accuracy varies by 0.1%, which is not a significant difference:

https://drive.google.com/drive/folders/1Xya2yZ0St6jbCqf4MMIDY0y9Yx7-mBgP?usp=sharing

MODEL COMPARISONS AND EVALUATIONS

Overall, the fine-tuned BERT provided the most accurate results than the CNN model. It had a higher accuracy than the CNN model, though the difference was barely significant.

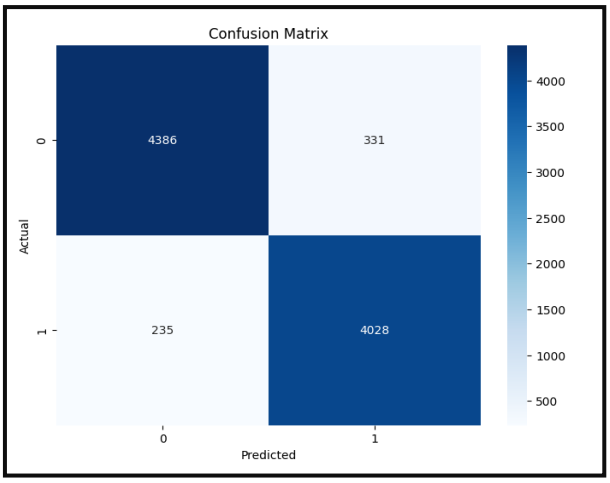
CNN CLASSIFICATION REPORT:

```
Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.93      0.94      4717
           1       0.92      0.94      0.93      4263

    accuracy                           0.94      8980
   macro avg       0.94      0.94      0.94      8980
weighted avg       0.94      0.94      0.94      8980

Training Time: 49.49 seconds
```

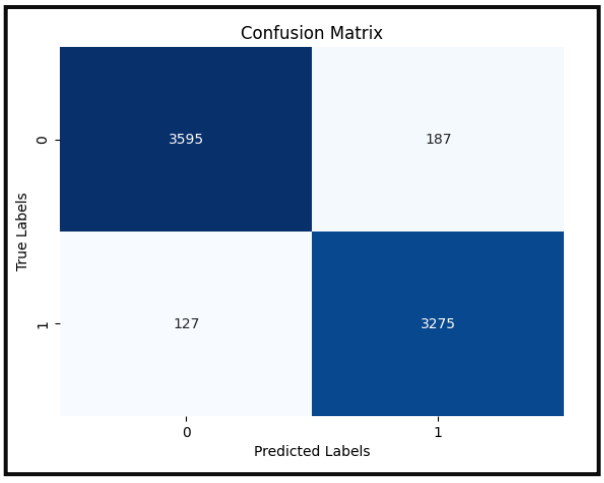
BERT CLASSIFICATION REPORT:

```
              precision    recall  f1-score   support

           0       0.95      0.96      0.96      3827
           1       0.95      0.95      0.95      3357

    accuracy                           0.95      7184
   macro avg       0.95      0.95      0.95      7184
weighted avg       0.95      0.95      0.95      7184

Training Time: 7462.60 seconds
```

CNN CONFUSION MATRIX:



BERT CONFUSION MATRIX:



CNN TESTING:

```
281/281 [==============================] - 1s 4ms/step
```

	title	label	pred_label
0	democratic senator oppose trump u supreme cour...	1	1
1	privileged trump supporter cry discrimination ...	0	0
2	video carly take view hack day saying demented...	0	0
3	crazy u pay import mentally ill illegal alien ...	0	0
4	trump owes lender least 315 million disclosure...	1	1
...	...	...	...
8975	new jersey hold governor primary shadow unpopu...	1	1
8976	bulgaria parliament pass anti corruption law e...	1	1
8977	trump retweets hilarious video hitting golf ba...	0	0
8978	obama confident u move right direction climate	1	1
8979	criminal detail emerge washington fast furious...	0	0

8980 rows × 3 columns

```
Accuracy: 0.9409799554565702
Precision: 0.9463825801919925
Recall: 0.9306930693069307
F1-score: 0.9384722544694684
```

BERT TESTING:

	title	label	pred_label
0	watch football great hershel walker give nfl k...	0	0
1	afghan civilian casualty air strike rise 50 pe...	1	1
2	vote hillary 2016 look may headed supreme court	0	0
3	detroit suburb elect first muslim majority cit...	0	0
4	conservative cry meryl streep destroys trump g...	0	0
...	...	...	...
7179	self funded longer trump need huge string atta...	0	0
7180	macedonian national arrested greece wiretap sc...	1	1
7181	trump host lunch republican senator obamacare ...	1	1
7182	hillary clinton crashing poll move obama strat...	0	0
7183	evan mcmullin issue dire warning american trum...	0	0

[7184 rows x 3 columns]

```
Accuracy: 0.9522550111358574
Precision: 0.9506578947368421
Recall: 0.9469764670837056
F1-score: 0.9488136099089689
```

The CNN model although provided a lower accuracy, did great. The difference between the training time is too significant to ignore. The CNN model had the lowest training time and almost provided the same results as the BERT model. The CNN model, whilst may not perform well on unseen data, will work accurately enough. The testing of each model is recorded in the new label column, pred_label.

We used the split test_set to test each model. We did test the CNN thrice, and in some instances, we did visibly see some incorrect predictions. We tested the BERT model a few times and failed to see any incorrect predictions as of yet, although we know that there are some inconsistencies.

The confusion matrix and classification report were based upon:

True Positive (TP): when predicted fake news is actually fake news.
True Negative (TN): when predicted true news is actually true news.
False Negative (FN): when predicted true news is actually fake news.
False Positive (FP): when predicted fake news is actually true news.

STRENGTHS, SHORTCOMINGS, AND LIMITATIONS

STRENGTHS: The CNN model was excellent at extracting features within the text and detecting patterns and features regardless of their position in the input. The CNN model needed the least amount of data cleaning and pre-processing. It was easier to build and understand the CNN model than the fine-tuned BERT-BASED-UNCASED. BERT was able to capture contextual relationships and meaning in text, providing higher accuracies than CNN. The data cleaning and pre-processing aided the training and predictive ability of each model immensely, allowing us to attain better results than expected. The CNN model took less than a minute to train.

SHORTCOMINGS: CNN requires fixed-size inputs. We had to use the title text instead of the news text to train the CNN model because of the limited size of the input that could process the data. We had to ensure that overfitting does not occur and did our best to prevent each model from getting too familiar with the trained data. BERT required a large number of computational resources that we did not have at hand. So, we limited the hyper-parameters to their bare minimum. We needed copious amounts of data to train BERT because BERT usually is inapt at generalising on small sets of data. BERT required immense data cleaning and pre-processing and also needed to transform the datasets into the right format, which is computationally expensive and complex. BERT took a lengthy amount of time to train.

LIMITATIONS: We were limited by the low resources we had. Finding the right dataset posed to be challenging. Both CNN and BERT were challenging to interpret and understand the reasons behind why they made such predictions. We were unsure about the biases the data contained and hence our results may lack diversity. We also noted that it is easy to evade the detection of each model, by providing manipulated data that is deceptive. To overcome these limitations, we could use additional techniques such as data augmentation, ensemble learning, and model interpretability methods. We can also get better resources and re-train with larger more unbiased data.

CONCLUSION

In summary, using a convolutional neural network (CNN) and fine-tuning the BERT-BASE-UNCASED model to detect fake news offers several advantages and benefits. CNNs excel at feature extraction, translation invariance, and learning hierarchical representations, making them effective at capturing local and global patterns in text and images. BERT models, on the other hand, offer contextual understanding, fine-tuning capabilities, and high-quality representation learning, enabling them to grasp complex semantic relationships. By leveraging their strengths and employing appropriate strategies, we can enhance the accuracy and robustness of fake news detection systems, contributing to the fight against misinformation and promoting a more reliable information ecosystem.

BIBLIOGRAPHY

Fake News Classification using transformer-based enhanced LSTM and BERT:
https://www.sciencedirect.com/science/article/pii/S2666307422000092

Fake news detection using parallel BERT deep neural networks:
https://arxiv.org/ftp/arxiv/papers/2204/2204.04793.pdf

Fake News Classification with BERT:
https://towardsdatascience.com/fake-news-classification-with-bert-afbeee601f41

Detecting Fake News — with a BERT Model:
https://medium.com/@skillcate/detecting-fake-news-with-a-bert-model-9c666e3cdd9b

Detection of fake news using deep learning CNN–RNN based methods:
https://www.sciencedirect.com/science/article/pii/S2405959521001375

New explainable method for BERT-based model in fake news detection:
https://www.nature.com/articles/s41598-021-03100-6

Classification of Fake News by Fine-tuning BERT Language Model:
https://publications.eai.eu/index.php/sis/article/view/2113

Fake news detection using CNN:
https://www.kaggle.com/code/davidegalelli/fake-news-detection-using-cnn

Fake News Classification Using Deep Learning:
https://www.analyticsvidhya.com/blog/2022/03/fake-news-classification-using-deep-learning/

Book on Using Transformers with Python, Tony Snake:
https://zlibrary.to/download/natural-language-processing-practical-using-transformers-with-python

Fake News Detection using a BERT-based Deep Learning Approach:
https://github.com/wutonytt/Fake-News-Detection/tree/main

Fake BERT: Fake news detection in social media with a BERT-based deep learning approach:
https://link.springer.com/article/10.1007/s11042-020-10183-2

Fake News Detection Using RNNs and Distributed Representations with Portuguese:
https://sbic.org.br/wp-content/uploads/2021/09/pdf/CBIC_2021_paper_163.pdf

---

Link to all our resources, models, and datasets:
https://drive.google.com/drive/folders/1Xya2yZ0St6jbCqf4MMIDY0y9Yx7-mBgP?usp=sharing