SC1015 Mini-Project

Fake News Detector

Team 10

Magnus Lim Runjie
Brian Tey Zhi Fong
Lim Shan Shan

Background

Motivation

Problem Statement

Motivation

Our motivation for embarking on this project stems from the importance of recognising fake news to improve the reliability of information online and reduce the harmful effects of misinformation.

Problem Statement

How might we leverage the ability of BERT to effectively detect fake news articles and combat the spread of misinformation in digital media platforms?

Exploratory Data Analysis

Dataset Overview
Dataset Cleaning and Preparation

Dataset Overview

01 WELFake

72,134 news articles

02 Source of data in WELFAKE

kaggle

University of Virginia
McIntire School of Commerce

REUTERS®

Dataset Overview

03

Preliminary Exploration

4 columns of data

```
data=pd.read_csv("WELFake_Dataset.csv")
print(data.shape)
print(data)
```

```
       Unnamed: 0                                              title  \
0               0  LAW ENFORCEMENT ON HIGH ALERT Following Threat...
1               1                                                NaN
2               2  UNBELIEVABLE! OBAMA’S ATTORNEY GENERAL SAYS MO...
3               3  Bobby Jindal, raised Hindu, uses story of Chri...
4               4  SATAN 2: Russia unvelis an image of its terrif...
...           ...                                                ...
72129       72129  Russians steal research on Trump in hack of U....
72130       72130   WATCH: Giuliani Demands That Democrats Apolog...
72131       72131  Migrants Refuse To Leave Train At Refugee Camp...
72132       72132  Trump tussle gives unpopular Mexican leader mu...
72133       72133  Goldman Sachs Endorses Hillary Clinton For Pre...

                                                    text  label
0      No comment is expected from Barack Obama Membe...      1
1         Did they post their votes for Hillary already?      1
2       Now, most of the demonstrators gathered last ...      1
3      A dozen politically active pastors came here f...      0
4      The RS-28 Sarmat missile, dubbed Satan 2, will...      1
...                                                  ...    ...
72129  WASHINGTON (Reuters) - Hackers believed to be ...      0
72130  You know, because in fantasyland Republicans n...      1
72131  Migrants Refuse To Leave Train At Refugee Camp...      0
72132  MEXICO CITY (Reuters) - Donald Trump’s combati...      0
72133  Goldman Sachs Endorses Hillary Clinton For Pre...      1
```

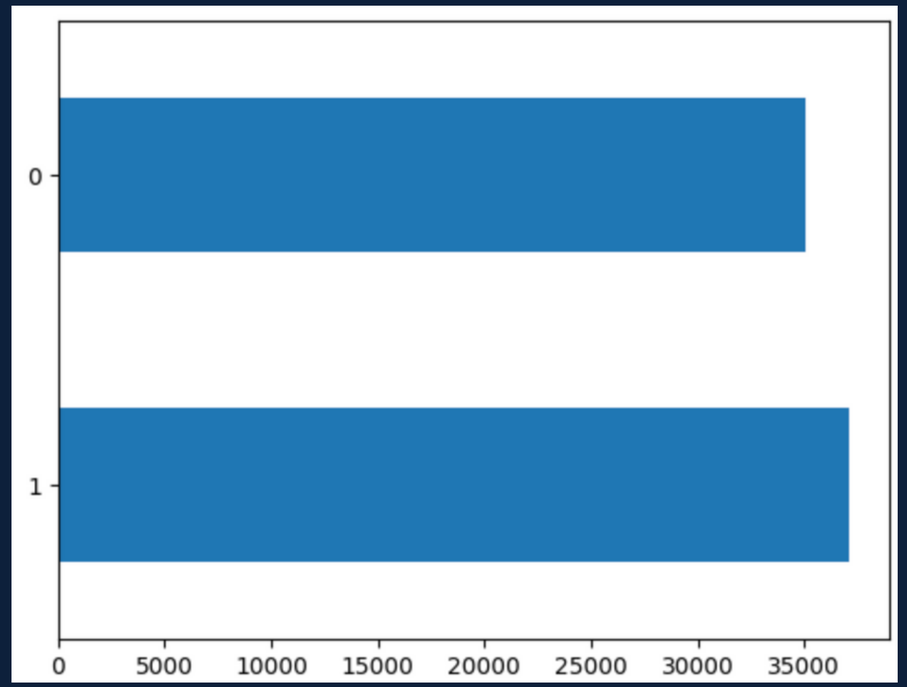
Dataset Overview

03

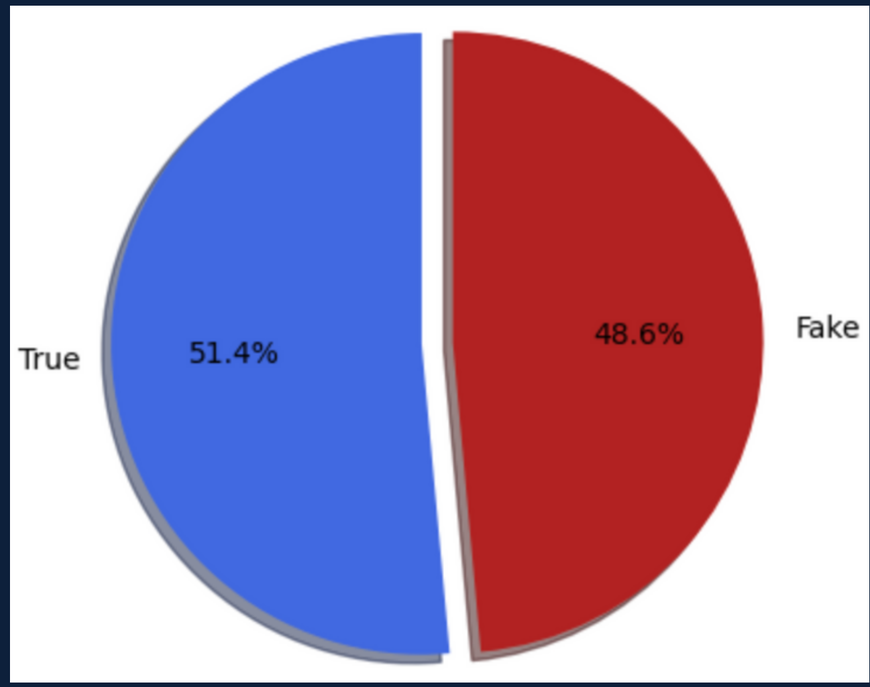
Preliminary Exploration

Distribution of news

```
data['label'].value_counts().plot(kind='barh')
```




```
# Checking if our data is well balanced
label_sizes = merged_data['Target'].value_counts()
plt.pie(label_sizes, explode=[0.1, 0], colors=['RoyalBlue', 'firebrick'], startangle=90, shadow=True,
        labels=['True', 'Fake'], autopct='%1.1f%%')
```

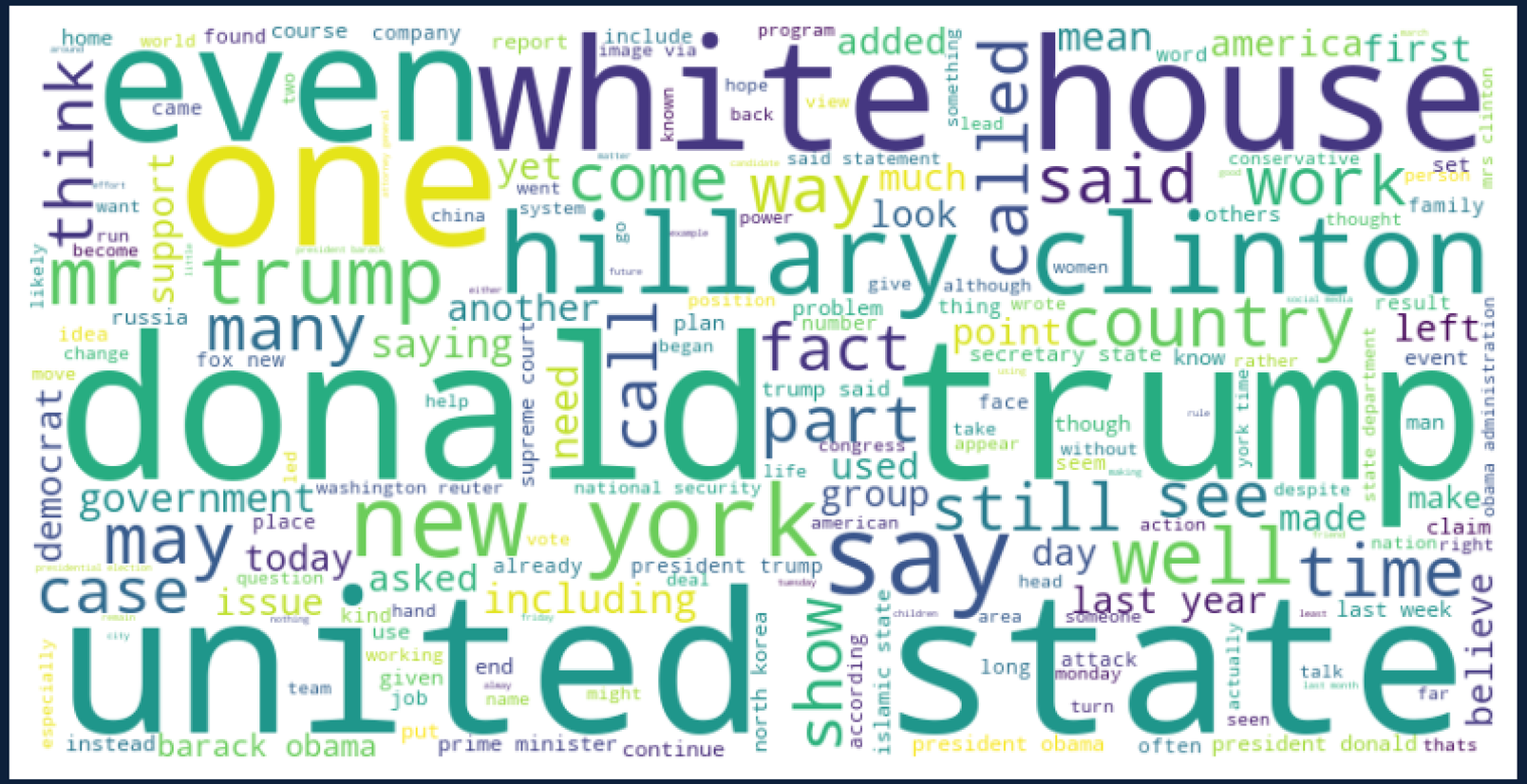


Dataset Overview

03

Preliminary Exploration

Wordcloud



Dataset Cleaning

01 Nature of Data

02 Content Authenticity

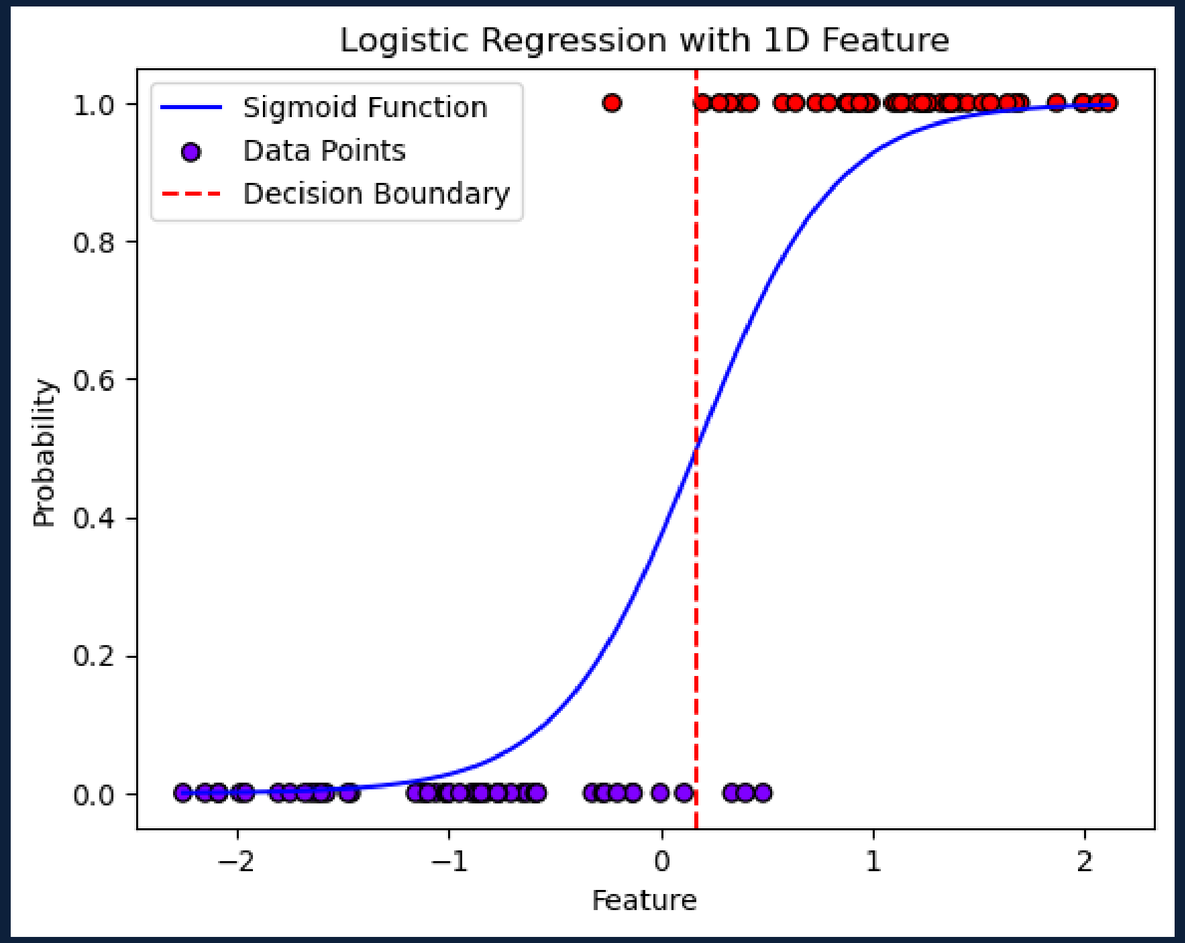
03 Focus on Textual Analysis

04 Prevent Overfitting

Model Architectures

Logistic Regression Model
BERT Transformer Model
Application to Fake News

Logistic Regression



Dataset Preparation

01 Data Preprocessing

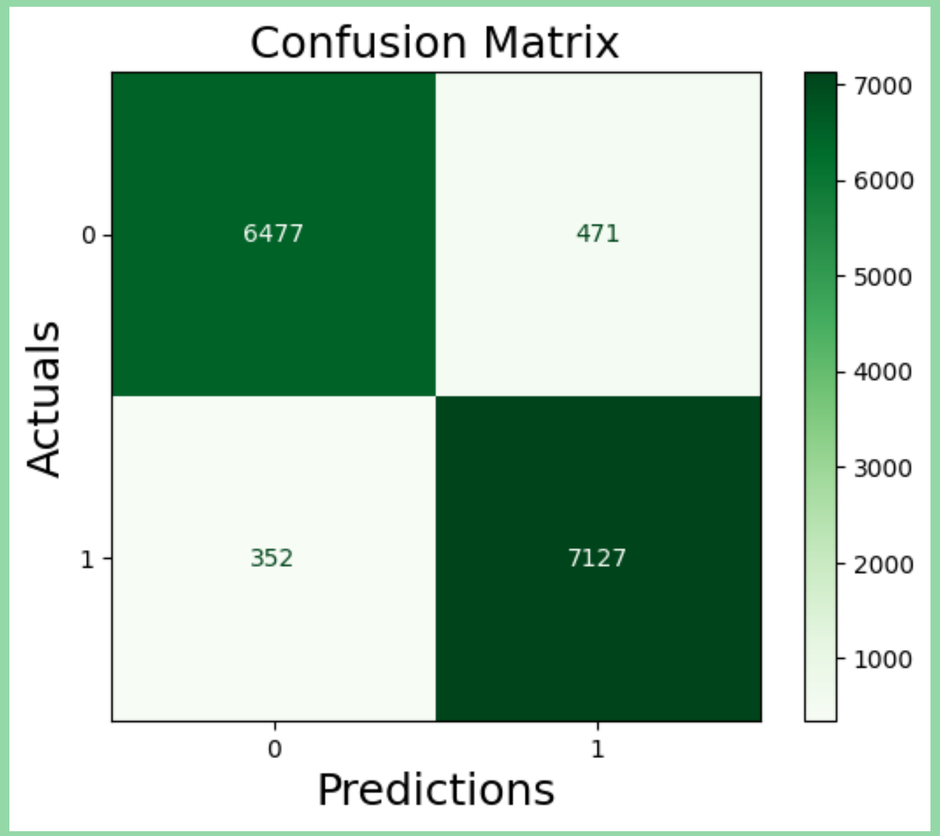
```
nltk.download('stopwords')

# Load dataset
df = pd.read_csv('WELFake_Dataset.csv')

# Convert 'title' and 'text' columns to strings
df['title'] = df['title'].astype(str)
df['text'] = df['text'].astype(str)

# Preprocessing
def preprocess(text):
    text = re.sub('[^a-zA-Z]', ' ', text)  # Remove non-alphabetic characters
    text = text.lower().split()  # Lowercase and split words
    ps = PorterStemmer()
    text = [ps.stem(word) for word in text if not word in set(stopwords.words('english'))]  # Stemming and stopword removal
    return ' '.join(text)

# Apply preprocessing
df['text'] = df['text'].apply(preprocess)
```

Dataset Preparation

02 Data split


```
# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

# Initialize the model
model = LogisticRegression()
```

Classifying Results



True Positive Rate	**7127/7479 = 95.3%**
True Negative Rate	**6477/6948 = 93.2%**

Overall Accuracy: 94.3%

Testing

Testing with unseen data:

```
# testing on unseen data
unseen_news_text = ["Baltimore bridge collapses after S'pore-flagged ship collision; search under way for survivors",     #1
                    "You Can Smell Hillary's Fear",                                                                       #0
                    "Watch The Exact Moment Paul Ryan Committed Political Suicide At A Trump Rally (VIDEO)",              #0
                    "New Changi Heritage Trail documents area's military, recreational and social history"                #1
                    ]
```

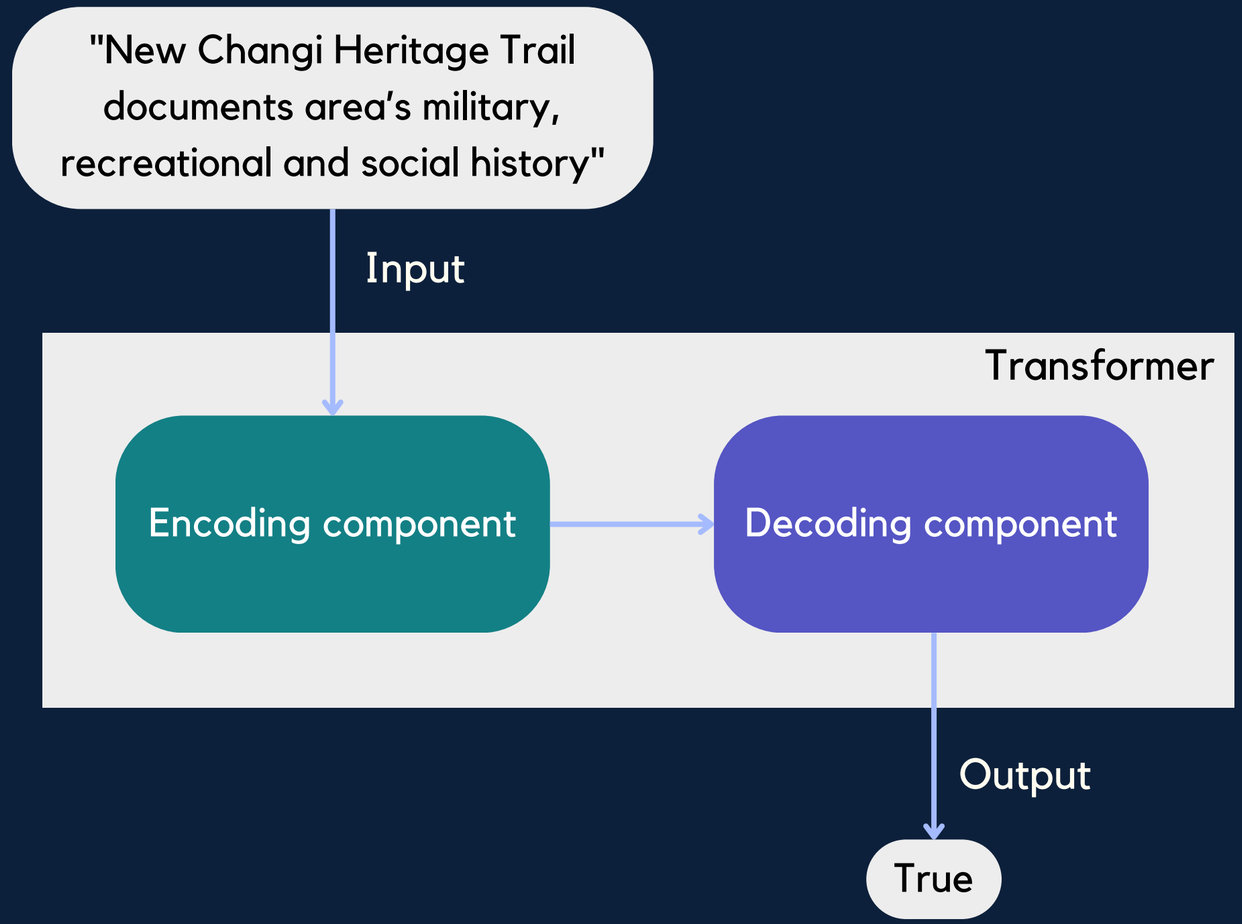
Results:

```
['True' 'True' 'True' 'True']
```

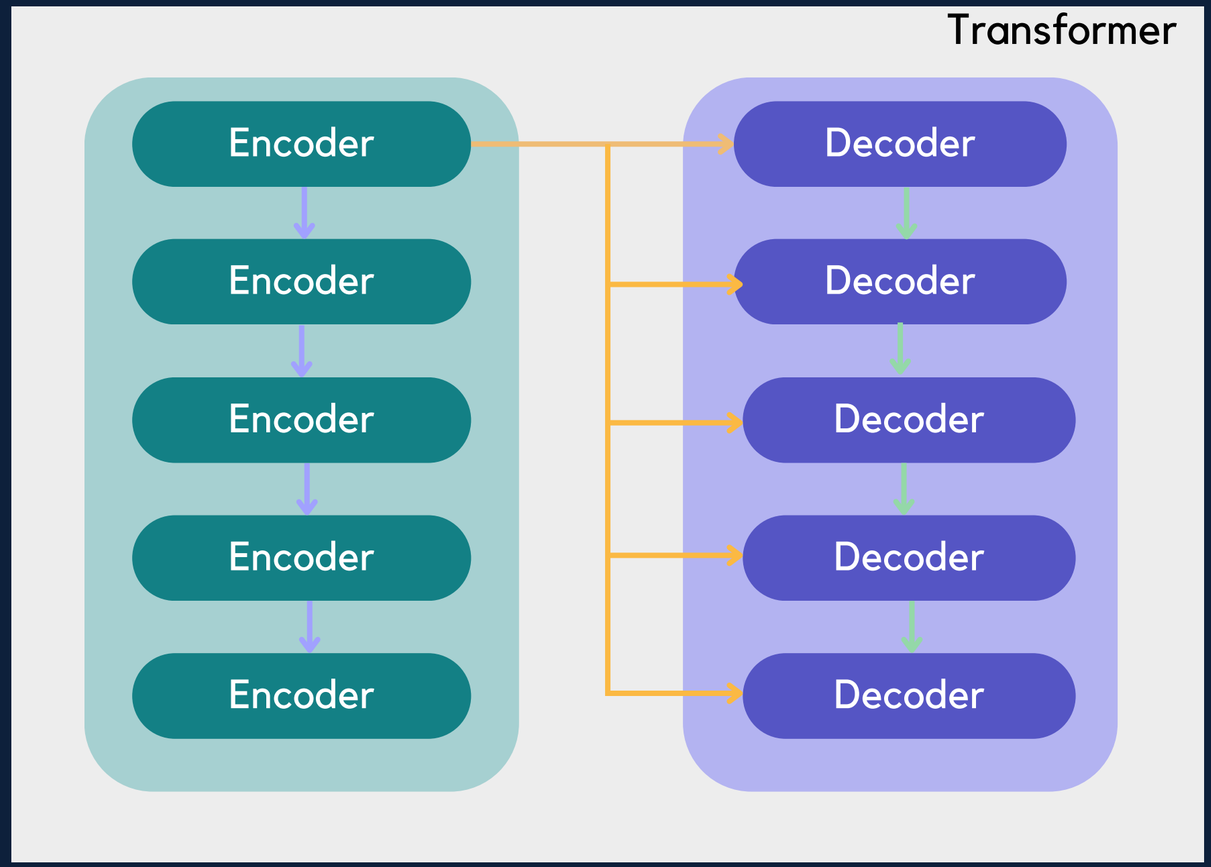
Unseen Dataset

News	Predicted Result	Actual Result
Baltimore bridge collapses after S'pore-flagged ship collision; search under way for survivors	REAL	REAL
You Can Smell Hillary's Fear	REAL	FAKE
Watch The Exact Moment Paul Ryan Committed Political Suicide At A Trump Rally (VIDEO)	REAL	FAKE
New Changi Heritage Trail documents area's military, recreational and social history	REAL	REAL

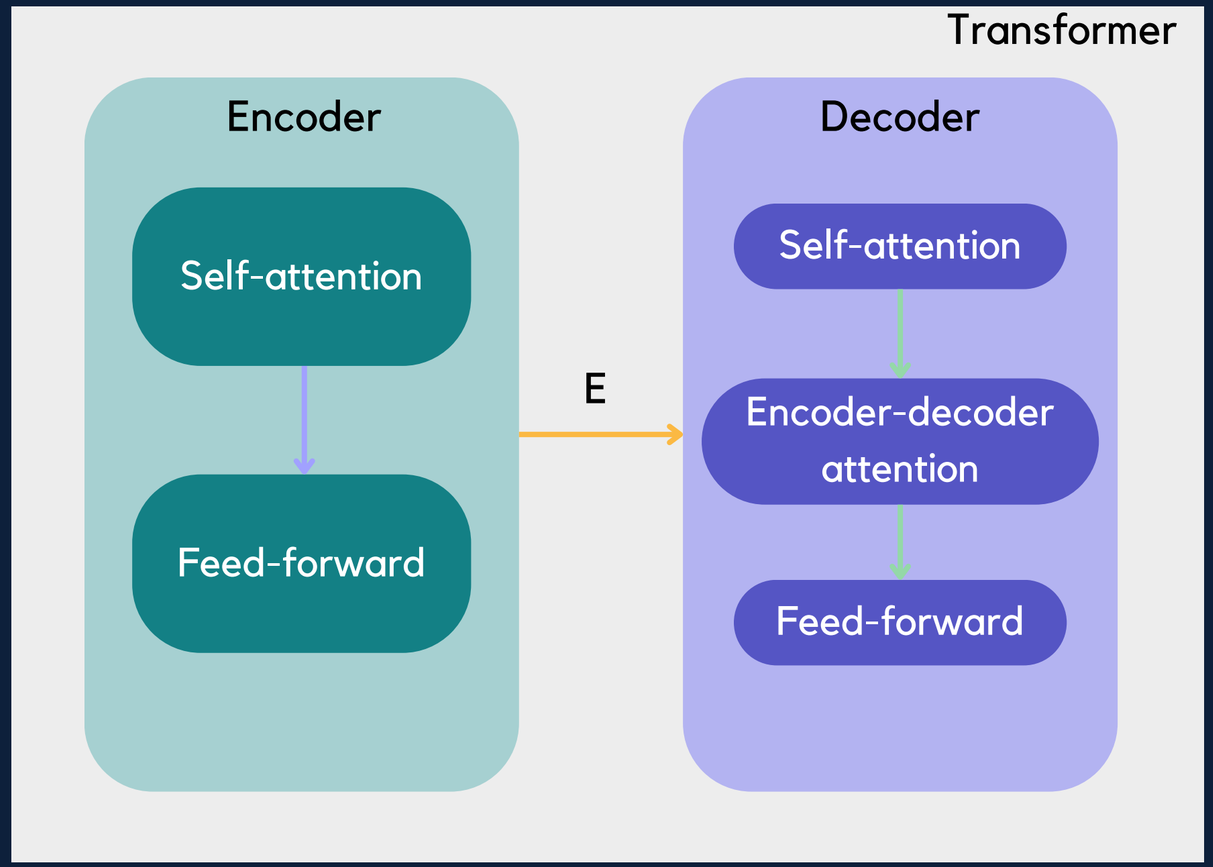
Transformers



Transformers



Transformers



BERT base

12 layers

Each layer processes tokens separately

768 Feed Forward Networks

Responsible for processing and transforming the input data

12 Attention Heads

Attend to different parts of the input simultaneously

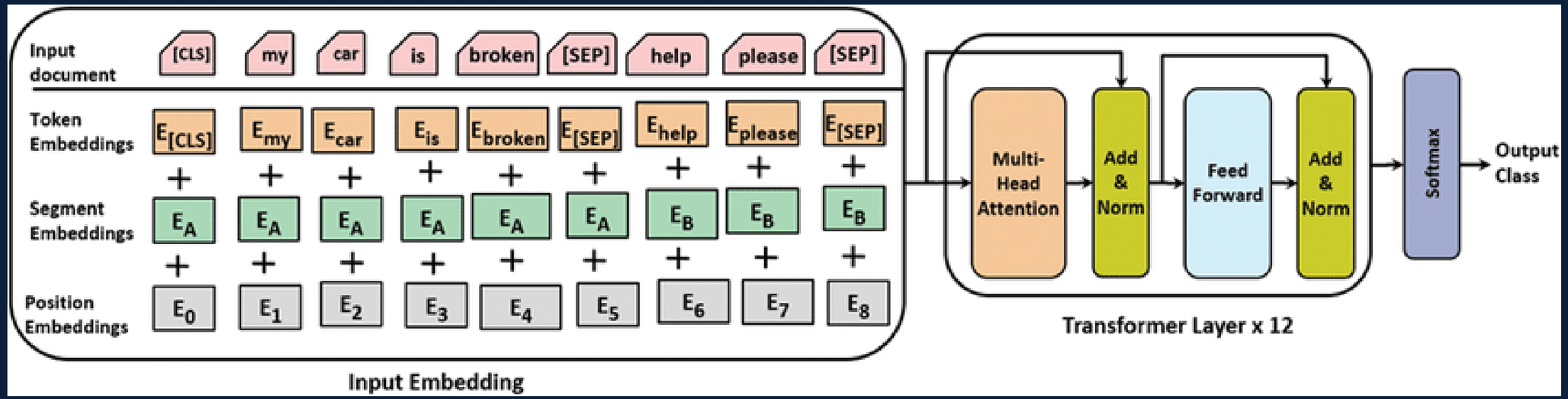


Image Credit: https://www.researchgate.net/figure/The-architecture-of-the-Fine-tuned-BERT-base-classifier_fig3_351392408

Dataset Preparation

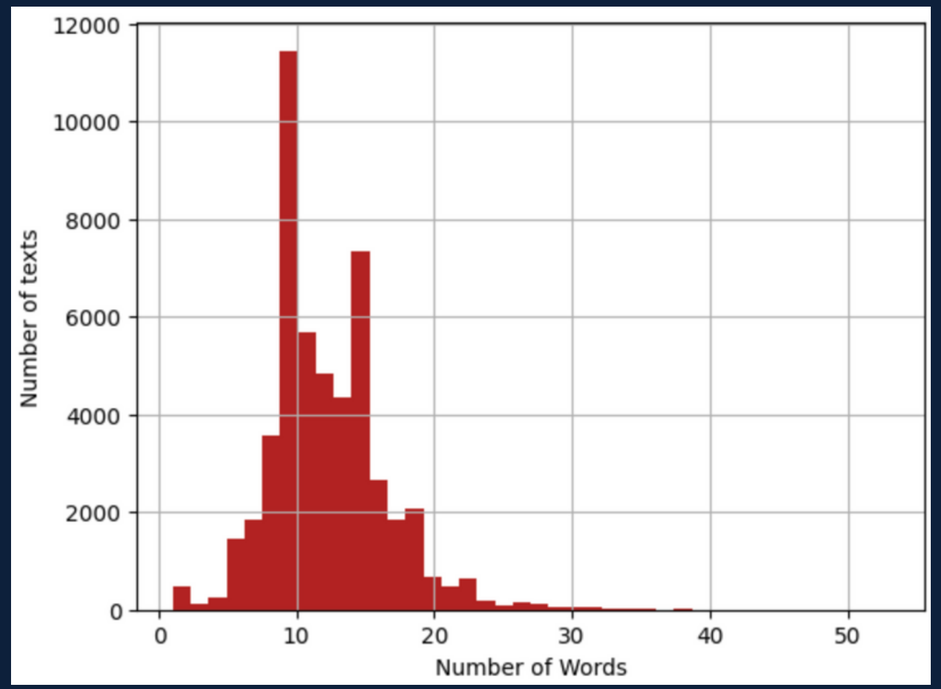
01 Data split


```
# Train-Validation-Test set split into 70:15:15 ratio
# Train-Temp split
train_text, temp_text, train_labels, temp_labels = train_test_split(merged_data['title'], merged_data['label'],
                                                                    random_state=2018,
                                                                    test_size=0.3,
                                                                    stratify=merged_data['Target'])
# Validation-Test split
val_text, test_text, val_labels, test_labels = train_test_split(temp_text, temp_labels,
                                                                random_state=2018,
                                                                test_size=0.5,
                                                                stratify=temp_labels)
```

Dataset Preparation

02 Data Tokenization

```
# Plot histogram of the number of words in train data 'title'
seq_len = [len(title.split()) for title in train_text]
pd.Series(seq_len).hist(bins = 40,color='firebrick')
plt.xlabel('Number of Words')
plt.ylabel('Number of texts')
```




```
# Majority of titles above have word length under 30.
# So, we set max title length as 30
MAX_LENGTH = 30
# Tokenize and encode sequences in the train set
tokens_train = tokenizer.batch_encode_plus(
    train_text.tolist(),
    max_length = MAX_LENGTH,
    pad_to_max_length=True,
    truncation=True
)
# tokenize and encode sequences in the validation set
tokens_val = tokenizer.batch_encode_plus(
    val_text.tolist(),
    max_length = MAX_LENGTH,
    pad_to_max_length=True,
    truncation=True
)
# tokenize and encode sequences in the test set
tokens_test = tokenizer.batch_encode_plus(
    test_text.tolist(),
    max_length = MAX_LENGTH,
    pad_to_max_length=True,
    truncation=True
)
```

Dataset Preparation

02 Conversion to tensors

```
# Convert lists to tensors
train_seq = torch.tensor(tokens_train['input_ids'])
train_mask = torch.tensor(tokens_train['attention_mask'])
train_y = torch.tensor(train_labels.tolist())
val_seq = torch.tensor(tokens_val['input_ids'])
val_mask = torch.tensor(tokens_val['attention_mask'])
val_y = torch.tensor(val_labels.tolist())
test_seq = torch.tensor(tokens_test['input_ids'])
test_mask = torch.tensor(tokens_test['attention_mask'])
test_y = torch.tensor(test_labels.tolist())
```

Dataset Preparation

03 Data Loader

```
# Data Loader structure definition
from torch.utils.data import TensorDataset, DataLoader, RandomSampler, SequentialSampler
batch_size = 32                                                 #define a batch size
train_data = TensorDataset(train_seq, train_mask, train_y)      # wrap tensors
train_sampler = RandomSampler(train_data)                       # sampler for sampling the data during training
train_dataloader = DataLoader(train_data, sampler=train_sampler, batch_size=batch_size)
                                                                # dataLoader for train set
val_data = TensorDataset(val_seq, val_mask, val_y)              # wrap tensors
val_sampler = SequentialSampler(val_data)                       # sampler for sampling the data during training
val_dataloader = DataLoader(val_data, sampler = val_sampler, batch_size=batch_size)
                                                                # dataLoader for validation set
```

PyTorch

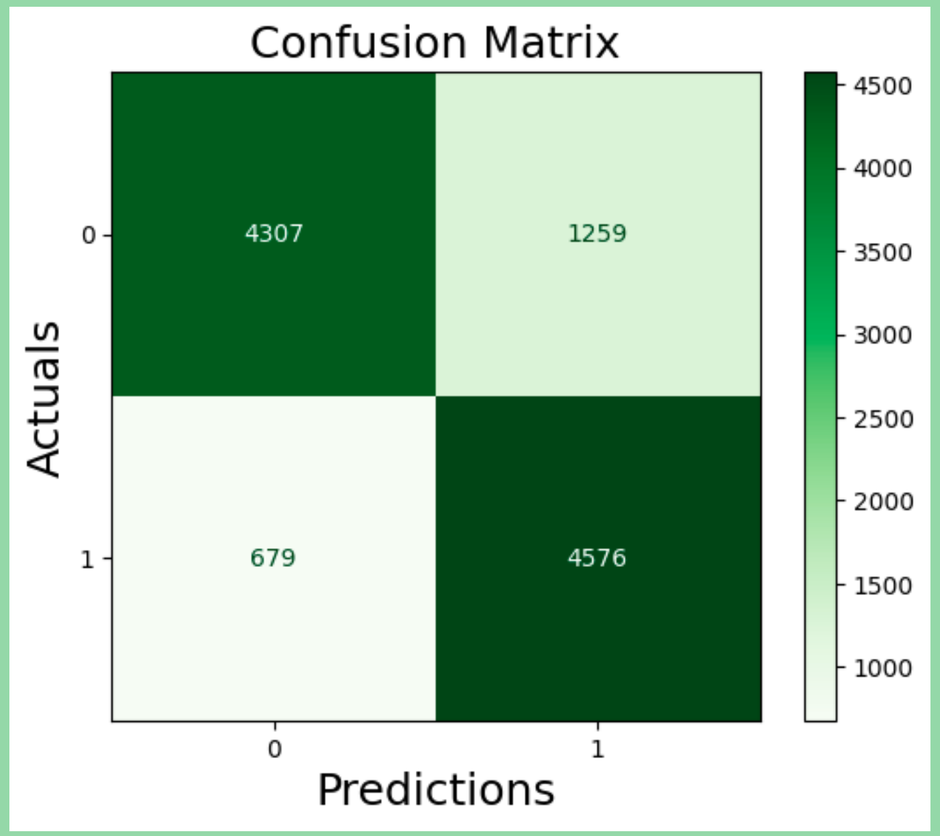
Training

```python
def train():
  model.train()
  total_loss, total_accuracy = 0, 0

  for step,batch in enumerate(train_dataloader):                # iterate over batches
    if step % 50 == 0 and not step == 0:                        # progress update after every 50 batches.
      print('  Batch {:>5,}  of  {:>5,}.'.format(step, len(train_dataloader)))
    batch = [r for r in batch]                                  # push the batch to gpu
    sent_id, mask, labels = [b.to(device) for b in batch]
    model.zero_grad()                                           # clear previously calculated gradients
    preds = model(sent_id, mask)                                # get model predictions for current batch
    loss = cross_entropy(preds, labels)                         # compute loss between actual & predicted values
    total_loss = total_loss + loss.item()                       # add on to the total loss
    loss.backward()                                             # backward pass to calculate the gradients
    torch.nn.utils.clip_grad_norm_(model.parameters(), 1.0)     # clip gradients to 1.0. It helps in preventing exploding gradient problem
    optimizer.step()                                            # update parameters
    preds=preds.detach().cpu().numpy()                          # model predictions are stored on GPU. So, push it to CPU

  avg_loss = total_loss / len(train_dataloader)                 # compute training loss of the epoch
                                                                # reshape predictions in form of (# samples, # classes)
  return avg_loss                                 # returns the loss and predictions
```

Validation

```
Training Loss: 0.433
Validation Loss: 0.395
              precision    recall  f1-score   support

           0       0.84      0.82      0.83      5566
           1       0.81      0.83      0.82      5255

    accuracy                           0.82     10821
   macro avg       0.82      0.83      0.82     10821
weighted avg       0.83      0.82      0.83     10821
```

Classifying Results



True Positive Rate	**4576/5255 = 87.1%**
True Negative Rate	**4307/5566 = 77.4%**

Overall Accuracy: 82.1%

Testing

Testing with unseen data:

```
# testing on unseen data
unseen_news_text = ["Baltimore bridge collapses after S'pore-flagged ship collision; search under way for survivors",     #1
                    "You Can Smell Hillary's Fear",                                                                       #0
                    "Watch The Exact Moment Paul Ryan Committed Political Suicide At A Trump Rally (VIDEO)",              #0
                    "New Changi Heritage Trail documents area's military, recreational and social history"                #1
                    ]
```

Results:

```
['True' 'False' 'False' 'True']
```

Blind Testing

Dataset Exploration

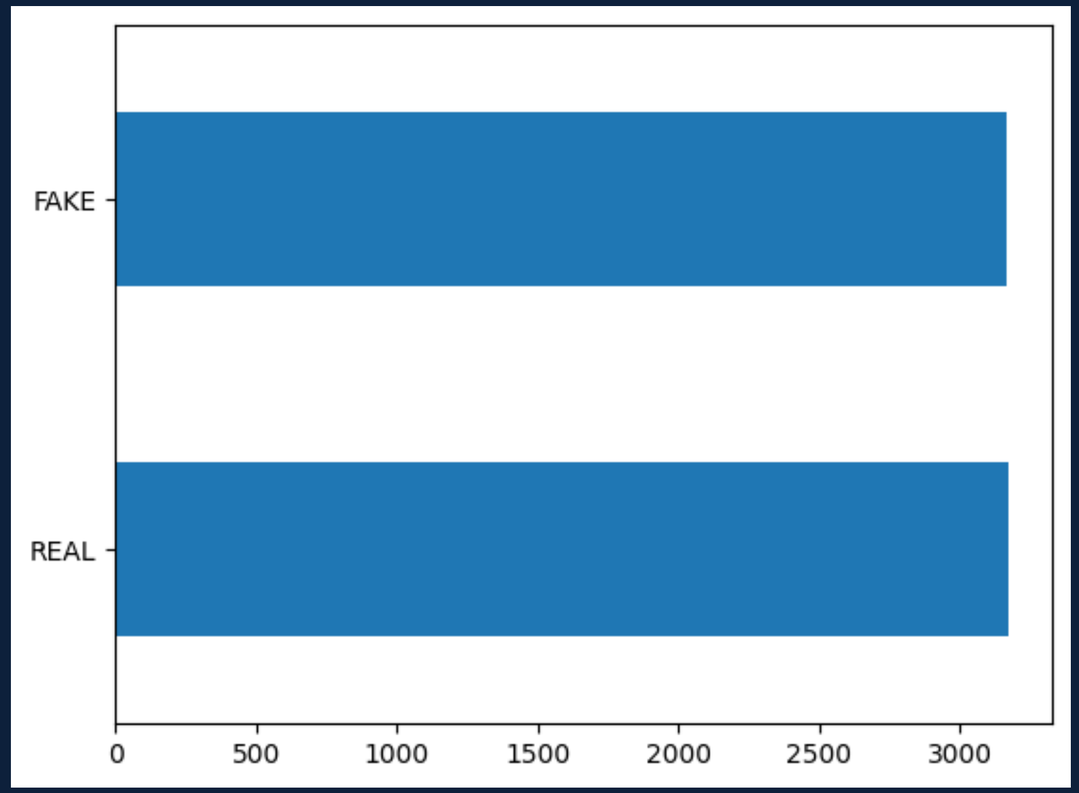
Preliminary Exploration

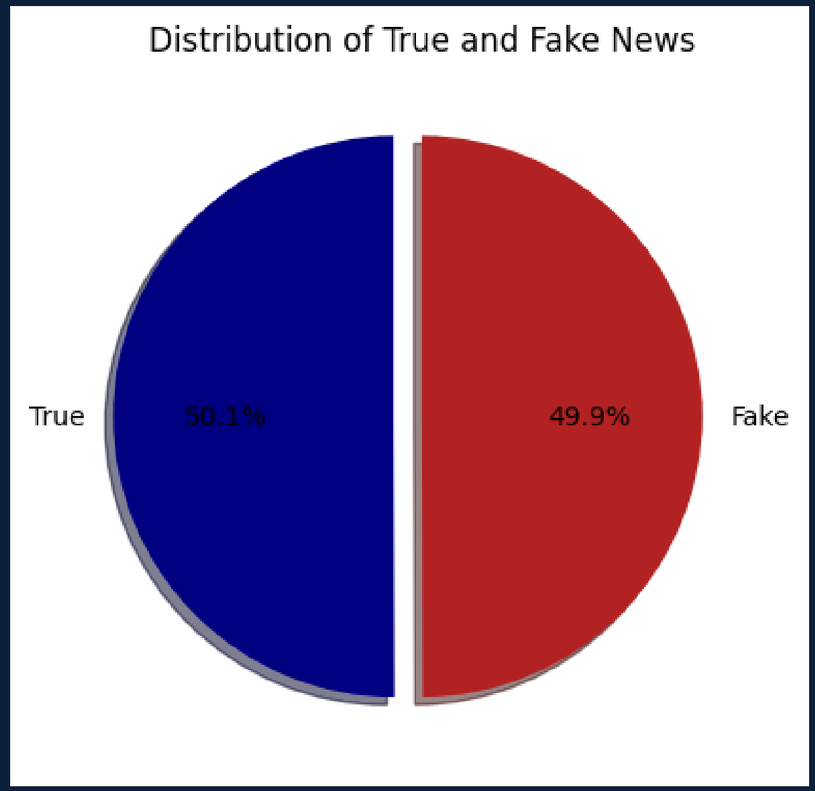
```
(6335, 4)
      Unnamed: 0                                              title  \
0           8476                       You Can Smell Hillary’s Fear
1          10294  Watch The Exact Moment Paul Ryan Committed Pol...
2           3608        Kerry to go to Paris in gesture of sympathy
3          10142  Bernie supporters on Twitter erupt in anger ag...
4            875   The Battle of New York: Why This Primary Matters
...          ...                                                ...
6330        4490  State Department says it can't find emails fro...
6331        8062  The ‘P’ in PBS Should Stand for ‘Plutocratic’ ...
6332        8622  Anti-Trump Protesters Are Tools of the Oligarc...
6333        4021  In Ethiopia, Obama seeks progress on peace, se...
6334        4330  Jeb Bush Is Suddenly Attacking Trump. Here's W...

                                                   text label
0     Daniel Greenfield, a Shillman Journalism Fello...  FAKE
1     Google Pinterest Digg Linkedin Reddit Stumbleu...  FAKE
2     U.S. Secretary of State John F. Kerry said Mon...  REAL
3     — Kaydee King (@KaydeeKing) November 9, 2016 T...  FAKE
4     It's primary day in New York and front-runners...  REAL
...                                                 ...   ...
6330  The State Department told the Republican Natio...  REAL
6331  The ‘P’ in PBS Should Stand for ‘Plutocratic’ ...  FAKE
6332   Anti-Trump Protesters Are Tools of the Oligar...  FAKE
6333  ADDIS ABABA, Ethiopia —President Obama convene...  REAL
6334  Jeb Bush Is Suddenly Attacking Trump. Here's W...  REAL

[6335 rows x 4 columns]
```

Blind Testing

Dataset Exploration

Distribution

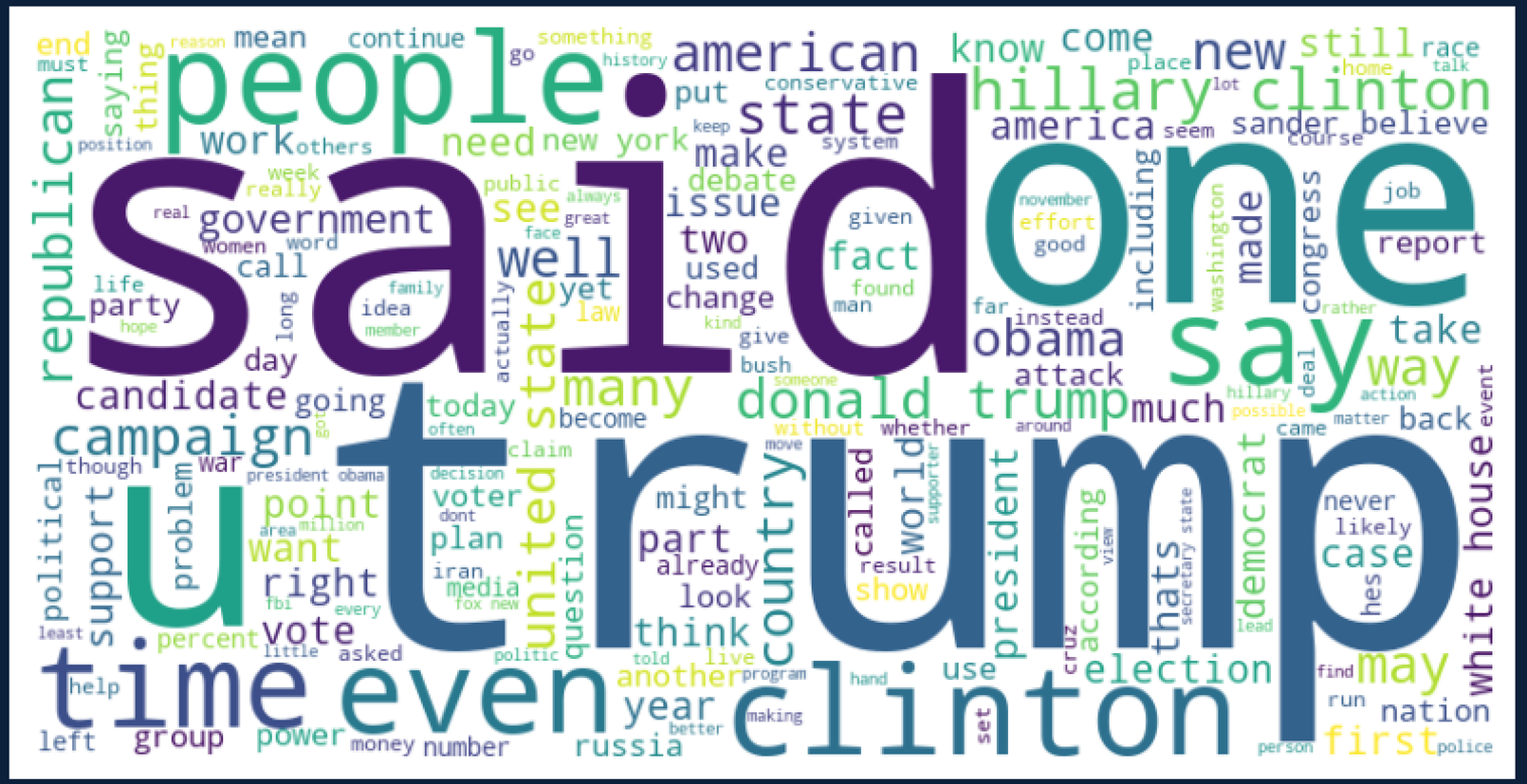




Blind Testing

Dataset Exploration

Wordcloud



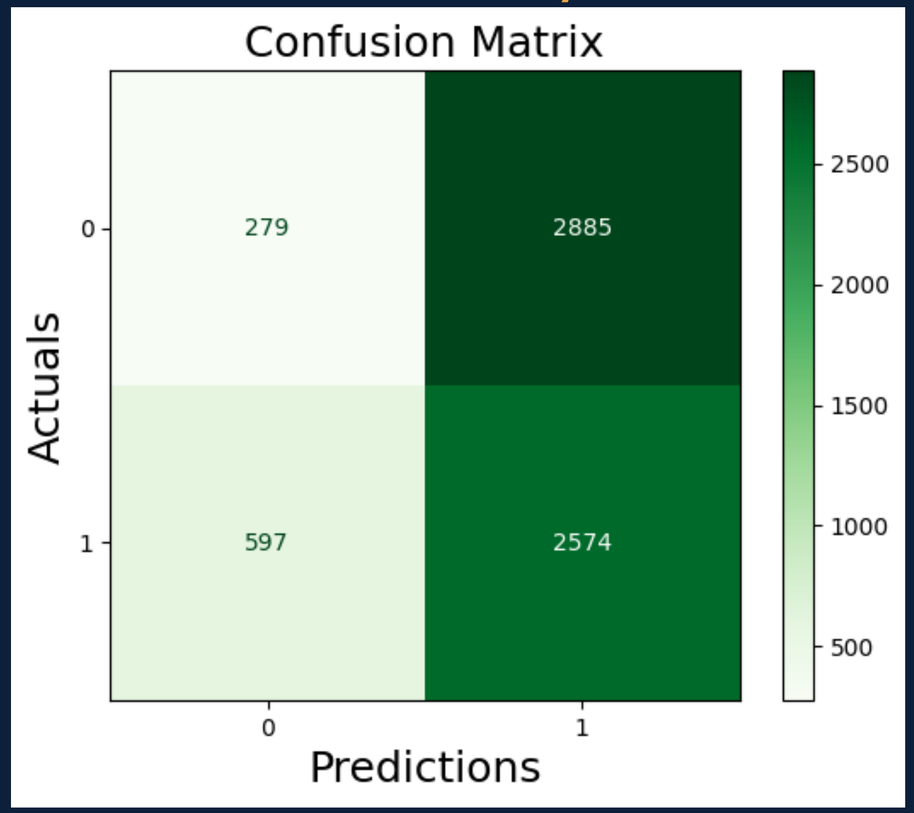
Blind Testing

Testing both models on a new dataset, “news.csv”

Results:

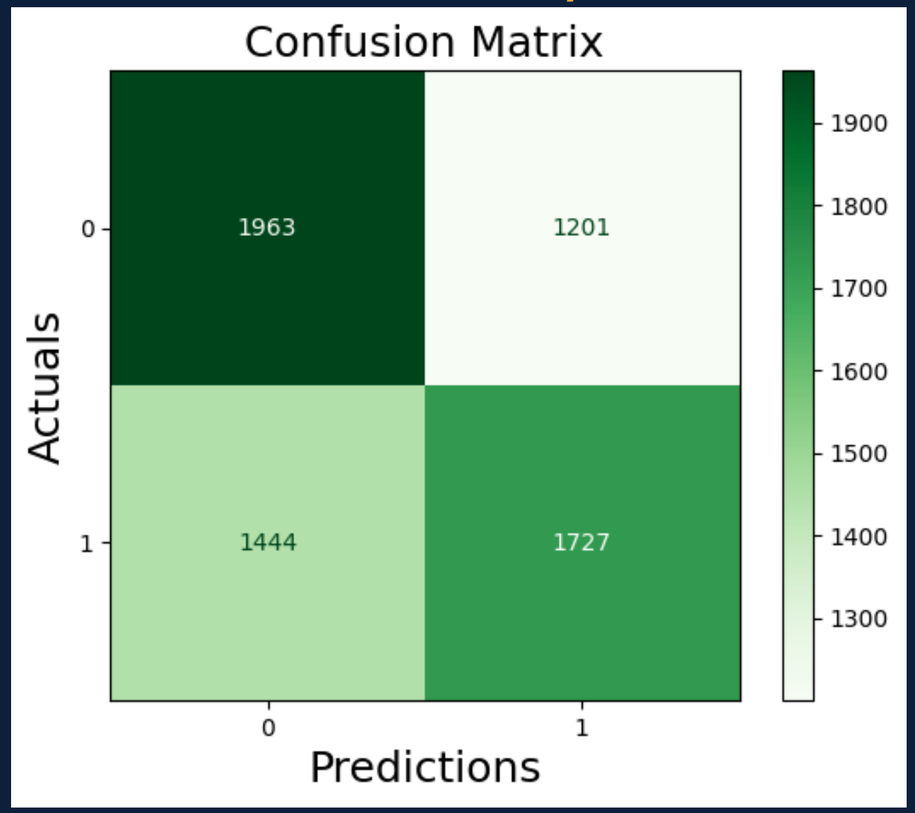
Logistic Regression

Overall Accuracy: 45.0%



BERT

Overall Accuracy: 58.2%



Insights

Effectiveness of transformer-based models like BERT

Importance of data preprocessing

Ongoing monitoring and evaluation of model performance

Conclusion

BERT is able to produce accurate models

Performs relatively well when tested on unseen datasets

Real-world application:

Flag suspicious articles for further review by human fact-checkers

Future Work

Updating and refining the model with new data

Automate the detection of fake news

Refine the model's predictions

Thank you

REFERENCES

- https://www.researchgate.net/figure/The-architecture-of-the-Fine-tuned-BERT-base-classifier_fig3_351392408