Fake News Detection Model

This code implements a fake news detection system using natural language processing and machine learning. It begins by importing necessary libraries and loading two datasets containing real and fake news articles. It then performs exploratory data analysis, including visualizing word clouds for both datasets. The code preprocesses the text data, removing special characters and converting it to lowercase. Next, it utilizes Word2Vec to create word embeddings, transforming words into numerical vectors that capture their semantic meaning. These embeddings are then used to train a deep learning model, specifically an LSTM network, to classify news articles as real or fake. Finally, the model is saved and tested on new input.

Importance:

This code is crucial for combating the spread of misinformation and fake news. By accurately identifying fake news articles, it helps users make informed decisions and avoid being misled by false information. This has significant implications for society, as the prevalence of fake news can erode trust in institutions, manipulate public opinion, and even incite violence. This code provides a practical solution for detecting fake news and mitigating its harmful effects. By leveraging advanced techniques in NLP and machine learning, it offers a valuable tool for promoting media literacy and ensuring the integrity of information.

Technologies Used:

Data Handling and Manipulation:

Pandas: Used for data loading, cleaning, transformation, and analysis. It provides data structures like DataFrames for efficient data handling. NumPy: Used for numerical computations, array operations, and mathematical functions. It's a fundamental library for scientific computing in Python. Data Visualization:

Matplotlib: Used for creating static, interactive, and animated visualizations in Python. It provides a wide range of plotting functions for various chart types. Seaborn: Built on top of Matplotlib, Seaborn provides a higher-level interface for creating statistically informative and visually appealing plots. **WordCloud:** Used for generating word clouds, which visually represent the frequency of words in a text corpus. Natural Language Processing (NLP):

NLTK: A comprehensive library for text processing tasks like tokenization, stemming, lemmatization, part-of-speech tagging, and more. spaCy: Another popular NLP library known for its speed and efficiency. It offers similar functionalities to NLTK and is often preferred for production environments. Gensim: Used for topic modeling, document similarity analysis, and word embeddings. It provides implementations of algorithms like Word2Vec and Doc2Vec. **TextBlob**: A user-friendly library for common NLP tasks like sentiment analysis, part-of-speech tagging, and noun phrase extraction. preprocess_kgptalkie: A custom preprocessing library that likely contains functions for cleaning and preparing text data for NLP tasks. Machine Learning (ML):

Scikit-learn: Used for various ML tasks like model selection, data preprocessing, and evaluation metrics. It provides a wide range of algorithms for classification, regression, clustering, and dimensionality reduction. **TensorFlow/Keras:** Used for building and training deep learning models. Keras is a high-level API that simplifies the development of neural networks.

Deep Learning:

LSTM (Long Short-Term Memory): A type of recurrent neural network (RNN) architecture well-suited for sequential data like text. It can capture long-range dependencies in the text.

Data Source:

The code utilizes two datasets from a GitHub repository:

Fake News Dataset: Obtained from https://raw.githubusercontent.com/laxmimerit/fake-real-news-dataset/refs/heads/main/data/Fake.csv

Real News Dataset: Obtained from https://raw.githubusercontent.com/laxmimerit/fake-real-news-dataset/refs/heads/main/data/True.csv

These datasets contain news articles labeled as either "fake" or "real," allowing the model to learn patterns and distinguish between them.

Explanation:

The code implements a fake news detection system using a combination of NLP and deep learning techniques. It involves the following key steps:

Data Loading and Exploration: The code loads the fake and real news datasets, explores their structure and content, and visualizes word clouds to gain insights into the language used in each type of news.

Data Preprocessing: The text data is preprocessed by removing special characters, converting to lowercase, and splitting into individual words. This prepares the data for further analysis and model training.

Word Embeddings: Word2Vec is used to create word embeddings, which represent words as numerical vectors capturing their semantic meaning. These embeddings enable the model to understand the relationships between words.

Model Building and Training: An LSTM network is built using TensorFlow/Keras. The model is trained on the preprocessed text data and word embeddings to classify news articles as real or fake. Model Evaluation and Saving: The trained model is evaluated on a test dataset to assess its performance using metrics like accuracy, precision, recall, and F1-score. The model is then saved for future use. Model Testing: The saved model is loaded and used to predict the authenticity of new input text.

```
from google.colab import drive
drive.mount('/content/drive')
```

Imports

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import nltk
import re
from wordcloud import WordCloud
```

```
from 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, SpatialDropout1D
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, accuracy_score, confusion_matrix
```

Exploring Fake News

```
fake = pd.read_csv('https://raw.githubusercontent.com/laxmimerit/fake-real-news-dataset/refs/heads/main/data/Fake.csv')
fake.text.iloc[900]
```

```
'The narrative regarding the Trump-Russia connection seems to be  nothing to see here  when it comes to supporters of the so-called  president. 
Donald Trump cannot abide anyone even suggesting that he had help to win in 2016, but the amount of smoke there with regards to the Russian hack
ing is too thick for there to be no fire. Hell, his own son just admitted something extraordinary. Donald Trump Jr. actually came out and said t
hat he, Trump s son-in-law Jared Kushner, and former campaign manager Paul Manafort actually met with a lawyer with connections to the Kremlin.A
pparently, this little secret pow wow occurred after Trump clinched the GOP nomination, and long before his many general election controversies,
during a time where it seemed sure that he would lose. Of course, Trump Jr. is saying that this meeting had nothing to with the election and eve
rything to do with adopting Russian orphans. He said of the encounter: We primarily discussed a program about the adoption of Russian c..'
```

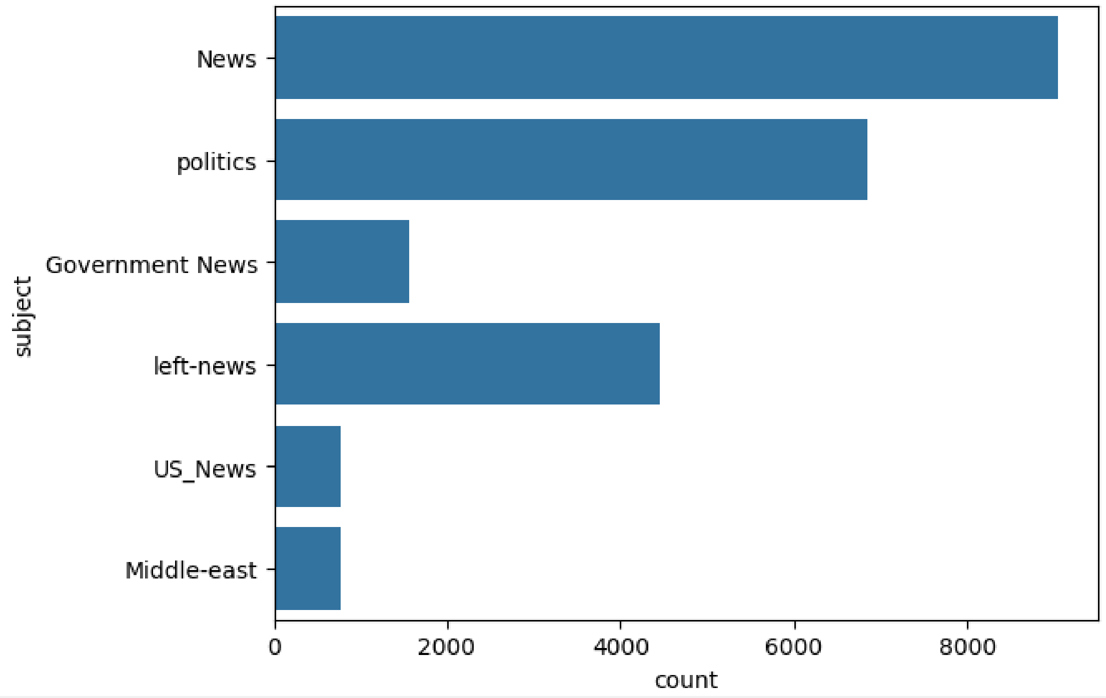
```
fake['subject'].value_counts()
```

	count
subject	
News	9050
politics	6841
left-news	4459
Government News	1570
US_News	783
Middle-east	778

dtype: int64

```
sns.countplot(fake['subject'])
```

```
<Axes: xlabel='count', ylabel='subject'>
```



Start coding or generate with AI.

WordCloud Fake News

Double-click (or enter) to edit

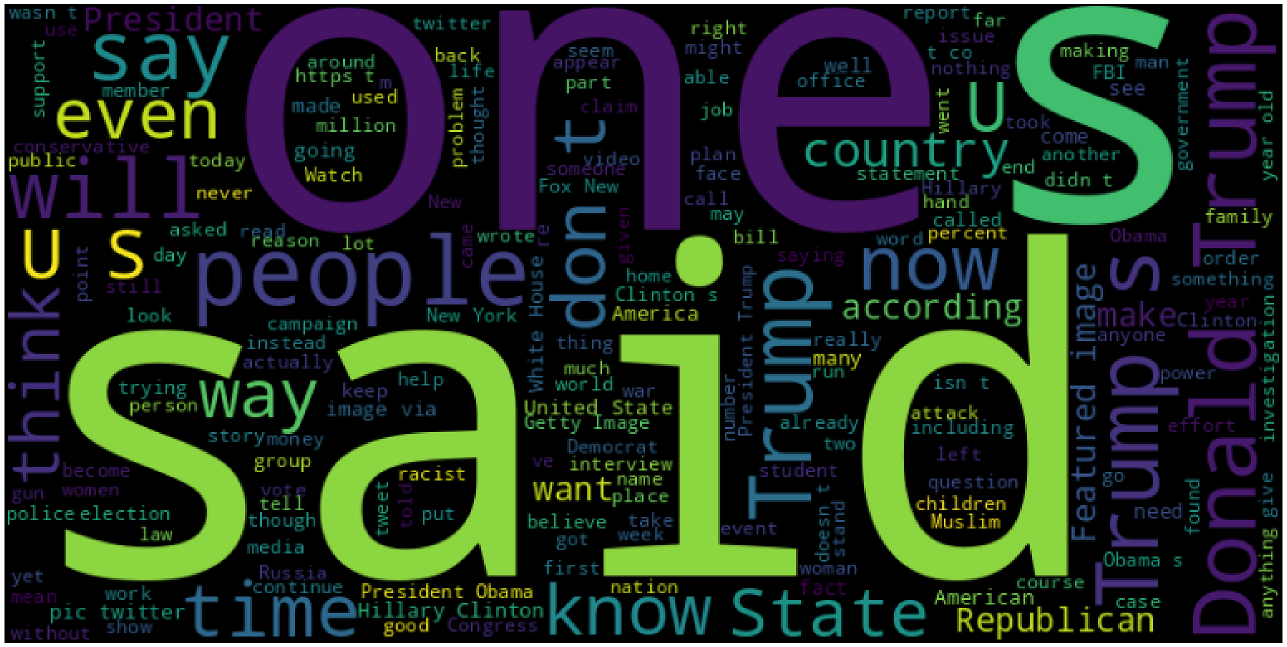
```
text = ' '.join(fake['text'].tolist())
```

```
text
```

```
'Donald Trump just couldn t wish all Americans a Happy New Year and leave it at that. Instead, he had to give a shout out to his enemies, haters
and  the very dishonest fake news media.  The former reality show star had just one job to do and he couldn t do it. As our Country rapidly grow
s stronger and smarter, I want to wish all of my friends, supporters, enemies, haters, and even the very dishonest Fake News Media, a Happy and
Healthy New Year,  President Angry Pants tweeted.  2018 will be a great year for America! As our Country rapidly grows stronger and smarter, I w
ant to wish all of my friends, supporters, enemies, haters, and even the very dishonest Fake News Media, a Happy and Healthy New Year. 2018 will
be a great year for America!  Donald J. Trump (@realDonaldTrump) December 31, 2017Trump s tweet went down about as welll as you d expect.What ki
nd of president sends a New Year s greeting like this despicable, petty, infantile gibberish? Only Trump! His lack of decency won t eve...'
```

```
wordcloud = WordCloud(width=800, height=400, background_color='black').generate(text)

# Display the word cloud
plt.figure(figsize=(10, 5))
plt.imshow(wordcloud)
plt.axis("off")
# plt.title(fontsize=26)
plt.show()
```



Exploring True News

```
real = pd.read_csv('https://raw.githubusercontent.com/laxmimerit/fake-real-news-dataset/refs/heads/main/data/True.csv')
```

```
real.text.iloc[100]
```

```
'WASHINGTON (Reuters) - Democratic Senator Elizabeth Warren is taking aim at budget chief Mick Mulvaney’s plan to fill the ranks of the U.S. con
sumer financial watchdog with political allies, according to letters seen by Reuters, the latest salvo in a broader battle over who should run t
he bureau. President Donald Trump last month appointed Mulvaney as acting director of the Consumer Financial Protection Bureau (CFPB), though th
e decision is being legally challenged by the agency’s deputy director, Leandra English, who says she is the rightful interim head. Mulvaney tol
d reporters earlier this month he planned to bring in several political appointees to help overhaul the agency, but Warren warned in a pair of l
etters sent Monday to Mulvaney and the Office of Personnel Management (OPM), which oversees federal hiring, that doing so was inappropriate and
potentially illegal. The CFPB is meant to be an independent agency staffed primarily by non-political employees. Hiring political appoint '
```

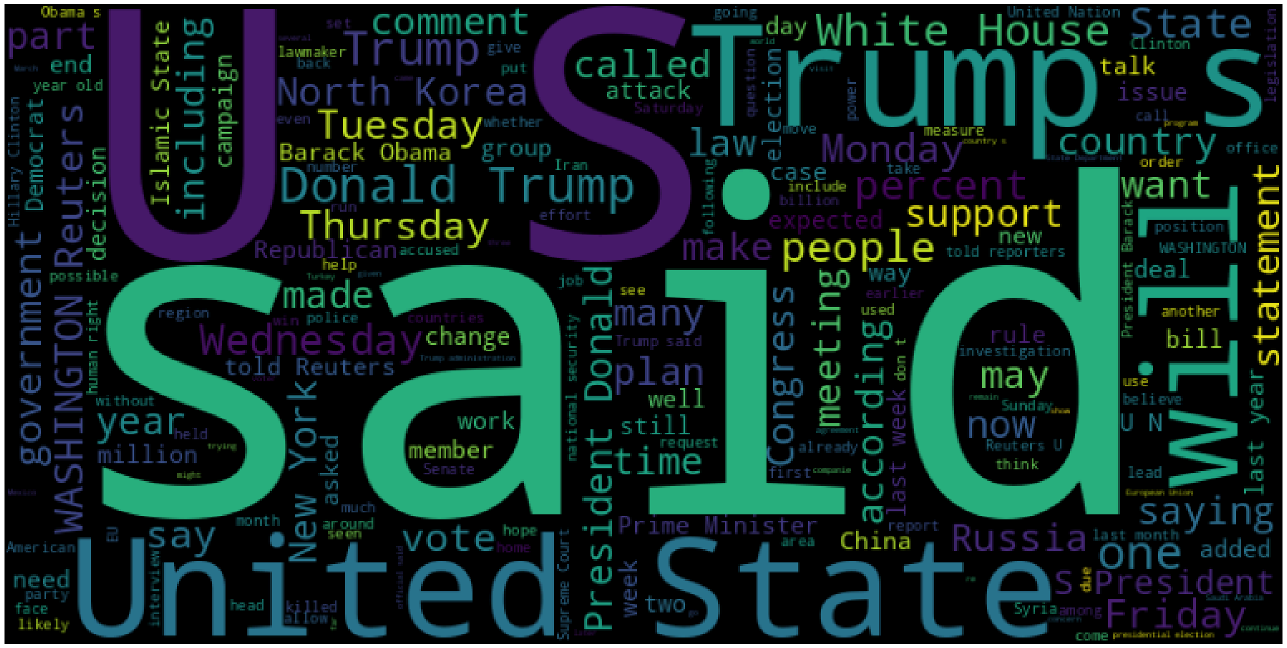
WordCloud True News

```
text = ' '.join(real['text'].tolist())


wordcloud = WordCloud(width=800, height=400, background_color='black').generate(text)

# Display the word cloud
plt.figure(figsize=(10, 5))
plt.imshow(wordcloud)
plt.axis("off")

plt.show()
```



```
unknown_publishers = []

for index, row in enumerate(real.text.values):
    try:
        record = row.split('-', maxsplit=1)
        record[1]

        assert(len(record[0])<120)

    except:
        unknown_publishers.append(index)
```

```
real.iloc[unknown_publishers].text
```

	text
7	The following statements were posted to the ve...
8	The following statements were posted to the ve...
12	The following statements were posted to the ve...
13	The following statements were posted to the ve...
14	(In Dec. 25 story, in second paragraph, corre...
...	...
20135	(Story corrects to million from billion in pa...
20500	(This Sept 8 story corrects headline, clarifi...
20667	(Story refiles to add dropped word not , in ...
21246	(Story corrects third paragraph to show Mosul...
21339	(Story corrects to fix spelling in paragraph ...

222 rows × 1 columns

dtype: object

Start coding or generate with AI.

```
publisher = []
tmp_text = []
for index, row in enumerate(real.text.values):
    if index in unknown_publishers:
        tmp_text.append(row)
        publisher.append('Unknown')
    else:
        record = row.split('-', maxsplit=1)
        publisher.append(record[0].strip())
        tmp_text.append(record[1].strip())


real['publisher'] = publisher
real['text'] = tmp_text
```

```
# real.head()
```

```
empty_fake_index = [index for index, text in enumerate(fake.text.tolist()) if str(text).strip()=='']
```

```
real['text'] = real['title'] + ' ' + real['text']
fake['text'] = fake['title'] + ' ' + fake['text']
```

```
real['text'] = real['text'].apply(lambda x: str(x).lower())
fake['text'] = fake['text'].apply(lambda x: str(x).lower())
```

Start coding or generate with AI.

Preprocesing of Text

```
real['class']=1
fake['class']=0
```

```
real = real[['text', 'class']]
```

```
fake = fake[['text', 'class']]
```

```
combined = pd.concat([real, fake], ignore_index=True)
```

```
combined.sample(5)
```

	text	class
11878	honduran president ignores new election calls,...	1
41709	yikes! shocking footage of black lives matter ...	0
9402	obama to asians worried about u.s. election: i...	1
11795	eu executive moves to punish poland over court...	1
38699	parents jailed and kids taken away for 90 minu...	0

```
!pip install spacy
!python -m spacy download en_core_web_sm
!pip install beautifulsoup4
!pip install textblob
!pip install git+https://github.com/laxmimerit/preprocess_kgptalkie.git --upgrade --force-reinstall
```

```
!pip install googletrans==4.0.0-rc1
```

```
import preprocess_kgptalkie as ps
```

```
combined['text'].apply(lambda x: ps.remove_special_chars(x))
```

	text
0	as us budget fight looms republicans flip thei...
1	us military to accept transgender recruits on ...
2	senior us republican senator let mr mueller do...
3	fbi russia probe helped by australian diplomat...
4	trump wants postal service to charge much more...
...	...
44893	mcpain john mccain furious that iran treated u...
44894	justice yahoo settles email privacy classactio...
44895	sunnistan us and allied safe zone plan to take...
44896	how to blow 700 million al jazeera america fin...
44897	10 us navy sailors held by iranian military s...

44898 rows × 1 columns

dtype: object

Start coding or generate with AI.

Word2Vec

```
import gensim
```

```
y = combined['class'].values
```

```
x = [d.split() for d in combined['text'].tolist()]
```

```
print(x[0])
```

```
['as', 'u.s.', 'budget', 'fight', 'looms,', 'republicans', 'flip', 'their', 'fiscal', 'script', 'the', 'head', 'of', 'a', 'conservative', 'republ
```

```
DIM = 100
w2v_model = gensim.models.Word2Vec(sentences=x, vector_size=DIM, window=10, min_count=1)
```

```
len(w2v_model.wv.key_to_index)
```

```
375437
```

```
w2v_model.wv.most_similar('pakistan')
```

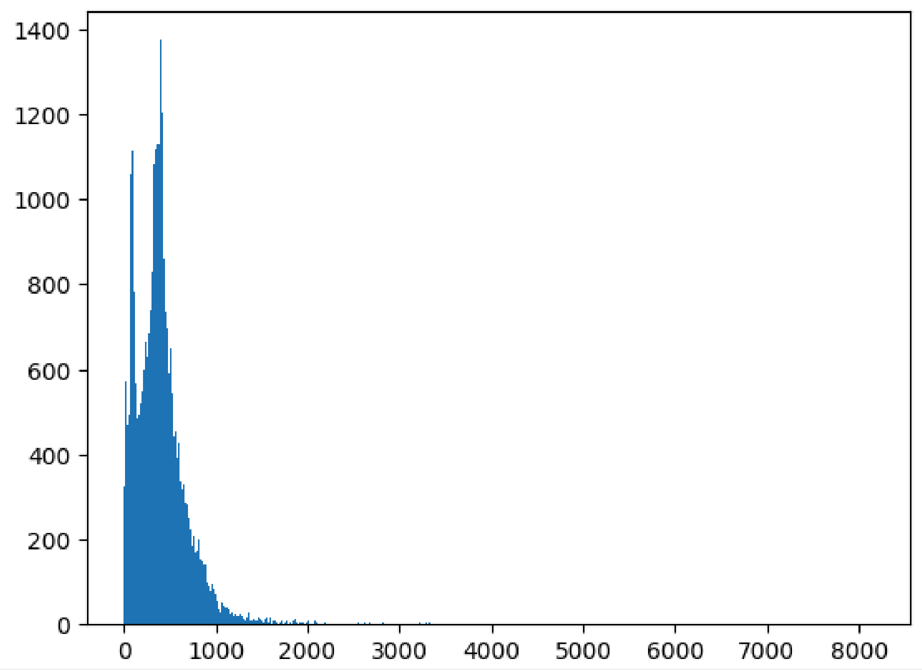
```
[('islamabad', 0.7653017640113831),
 ('afghanistan,', 0.7375413775444031),
 ('pakistan,', 0.7262160181999207),
 ('india', 0.7209123373031616),
 ('afghanistan', 0.7204774022102356),
 ('taliban', 0.7068694233894348),
 ('somalia', 0.6940649151802063),
 ('islamist', 0.6627404689788818),
 ('isis', 0.6453794240951538),
 ('afghanistan.', 0.6393990516662598)]
```

```
tokenizer = Tokenizer()
tokenizer.fit_on_texts(x)
```

```
x = tokenizer.texts_to_sequences(x)
```

```
# tokenizer.word_index
```

```
plt.hist([len(x) for x in x], bins=700)
plt.show()
```



```
nos = np.array([len(x) for x in x])
len(nos[nos>1000])
```

```
1606
```

```
max_len = 1000
x = pad_sequences(x, maxlen=max_len)
```

```
len(x[1])
```

```
1000
```

```
vocab_size = len(tokenizer.word_index) + 1
```

```
def get_weight_matrix(model, vocab):

    weight_matrix = np.zeros((vocab_size, DIM))
    for word, i in vocab.items():
      weight_matrix[i] = model.wv[word]
    return weight_matrix
```

```
embedding_vectors = get_weight_matrix(w2v_model, tokenizer.word_index)
```

```
embedding_vectors.shape
```

```
(375438, 100)
```

Model Training

```
# prompt: model training

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)
model = Sequential()
model.add(Embedding(vocab_size, output_dim=DIM, weights=[embedding_vectors], input_length=max_len, trainable=False))
model.add(SpatialDropout1D(0.3))
model.add(LSTM(100, dropout=0.3, recurrent_dropout=0.3))
model.add(Dense(1, activation='sigmoid'))
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
model.summary()
history = model.fit(x_train, y_train, epochs=6, batch_size=128, validation_data=(x_test, y_test))
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/embedding.py:90: UserWarning: Argument `input_length` is deprecated. Just remove it
  warnings.warn(
Model: "sequential"
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━┓
┃ Layer (type)                    ┃ Output Shape           ┃       Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━┩
│ embedding (Embedding)           │ ?                      │    37,543,800 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ spatial_dropout1d (SpatialDropout1D) │ ?                 │   0 (unbuilt) │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ lstm (LSTM)                     │ ?                      │   0 (unbuilt) │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense (Dense)                   │ ?                      │   0 (unbuilt) │
└─────────────────────────────────┴────────────────────────┴───────────────┘
 Total params: 37,543,800 (143.22 MB)
 Trainable params: 0 (0.00 B)
 Non-trainable params: 37,543,800 (143.22 MB)
Epoch 1/6
281/281 ━━━━━━━━━━━━━━━━━━━━ 488s 2s/step - accuracy: 0.8506 - loss: 0.3394 - val_accuracy: 0.9624 - val_loss: 0.1013
Epoch 2/6
281/281 ━━━━━━━━━━━━━━━━━━━━ 487s 2s/step - accuracy: 0.9466 - loss: 0.1469 - val_accuracy: 0.9628 - val_loss: 0.0996
Epoch 3/6
281/281 ━━━━━━━━━━━━━━━━━━━━ 498s 2s/step - accuracy: 0.9604 - loss: 0.1150 - val_accuracy: 0.9820 - val_loss: 0.0567
Epoch 4/6
281/281 ━━━━━━━━━━━━━━━━━━━━ 470s 2s/step - accuracy: 0.9638 - loss: 0.1066 - val_accuracy: 0.9843 - val_loss: 0.0500
Epoch 5/6
281/281 ━━━━━━━━━━━━━━━━━━━━ 498s 2s/step - accuracy: 0.9671 - loss: 0.0971 - val_accuracy: 0.9874 - val_loss: 0.0436
Epoch 6/6
281/281 ━━━━━━━━━━━━━━━━━━━━ 504s 2s/step - accuracy: 0.9596 - loss: 0.1113 - val_accuracy: 0.9882 - val_loss: 0.0387
```

```
model.summary()
```

```
Model: "sequential"
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━┓
┃ Layer (type)                    ┃ Output Shape           ┃       Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━┩
│ embedding (Embedding)           │ (None, 1000, 100)      │    37,543,800 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ spatial_dropout1d (SpatialDropout1D) │ (None, 1000, 100) │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ lstm (LSTM)                     │ (None, 100)            │        80,400 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense (Dense)                   │ (None, 1)              │           101 │
└─────────────────────────────────┴────────────────────────┴───────────────┘
 Total params: 37,785,305 (144.14 MB)
 Trainable params: 80,501 (314.46 KB)
 Non-trainable params: 37,543,800 (143.22 MB)
 Optimizer params: 161,004 (628.93 KB)
```

```
model.save('/content/drive/My Drive/fake/real_news_detection_model1.keras')

x = [' President Bashar al-Assad and his family have no role in the future of Syria, U.S. Secretary of State Rex Tillerson said on Thursday ahead of
x  = tokenizer.texts_to_sequences(x)
x = pad_sequences(x, maxlen=max_len)

if (model.predict(x) >= 0.5).astype(int):
  print('Real')

else:
  print('False')
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 399ms/step
Real
```

```
x = [' The narrative regarding the Trump-Russia connection seems to be  nothing to see here  when it comes to supporters of the so-called  president.
x  = tokenizer.texts_to_sequences(x)
```