Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)
Department of Computer Science and Engineering (Data Science)

Advanced Computational Linguistics

Experiment 3

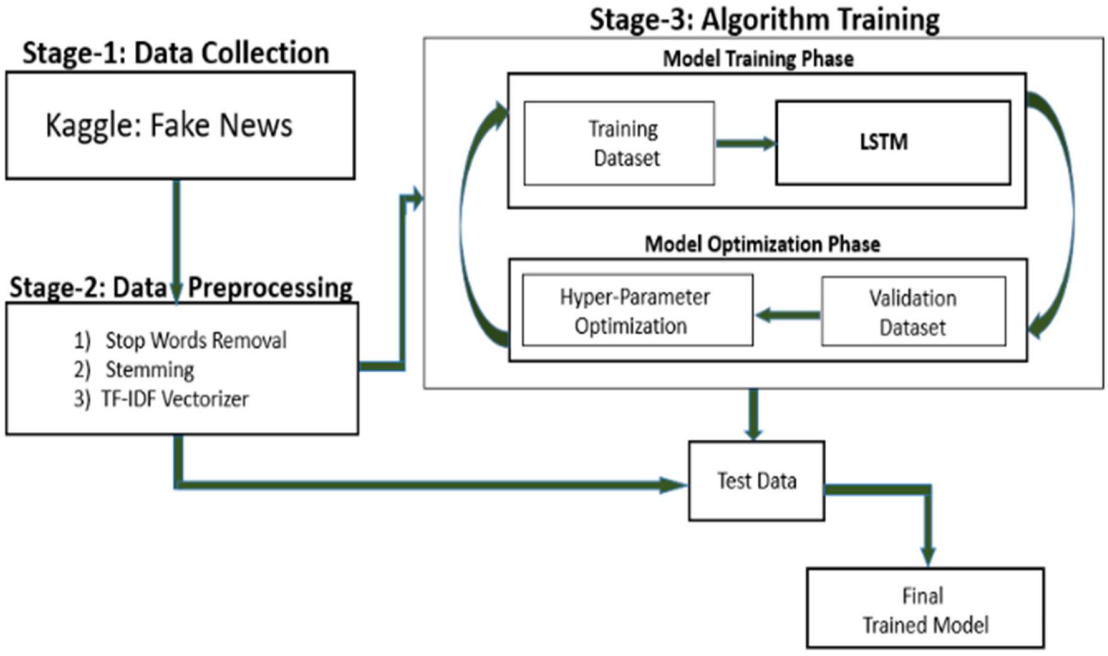
Name: Umang Kirit Lodaya **SAP ID: 60009200032**

Batch: K – K1 / D11

Aim: Implement Fake News Classifier using LSTM-Deep Learning Model

Theory:

Fake news classification is the process of identifying and categorizing news articles, stories, or information that are intentionally fabricated, misleading, or inaccurate, with the aim of deceiving readers or manipulating public opinion. This classification is usually performed using various machine learning techniques, natural language processing (NLP) algorithms, and data analysis methods. The goal is to distinguish between credible and accurate information and content that lacks authenticity.



Using LSTM For Fake News Classification:

Using an LSTM (Long Short-Term Memory) model for fake news classification involves several steps, including data preprocessing, model building, training, and evaluation. Here's a high-level guide to using an LSTM model for fake news classification:

1. **Data Preprocessing:**
   Load and preprocess your fake news dataset. This may involve tasks such as tokenization, padding, and converting text to sequences of word indices. Split your dataset into training and testing sets.
2. **Tokenization and Padding:**
   Tokenize the text data into words or subwords. Convert the tokenized sequences into integer sequences using a tokenizer. Pad the sequences to a fixed length to ensure consistent input size for the LSTM model.
3. **Building the LSTM Model:**
   Import the necessary libraries (e.g., TensorFlow, Keras). Build an LSTM-based model architecture. Typically, this includes an Embedding layer, one or more LSTM layers, and possibly additional dense layers. Compile the model, specifying the loss function (e.g., binary cross-entropy for binary classification), optimizer (e.g., Adam), and evaluation metrics (e.g., accuracy).
4. **Model Training:**
   Train the LSTM model using your preprocessed training data. Monitor training progress and consider using techniques like early stopping to prevent overfitting.
5. **Model Evaluation:**
   Evaluate the trained model's performance on the test dataset using metrics like accuracy, precision, recall, and F1-score. Analyze the confusion matrix to understand the model's performance in different categories (fake vs. real news).
6. **Fine-Tuning and Optimization:**
   Experiment with different hyperparameters (e.g., LSTM units, embedding dimensions) and architectures to improve model performance. Consider using techniques like dropout and regularization to prevent overfitting.
7. **Inference and Prediction:**
   Use the trained LSTM model to predict the authenticity of new news articles. Tokenize, pad, and preprocess the new text before passing it to the model for prediction.

Lab Experiments to be Performed in This Session: -

Exercise 1: Perform Fake New Classification for Fake News Dataset

Exercise 2: Perform Fake News Classification for Hindi News Dataset

NAME: UMANG KIRIT LODAYA

SAP ID: 60009200032

BATCH: K - K1 / D11

FAKE ENGLISH NEWS CLASSIFIER

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import collections
```

```
/opt/conda/lib/python3.10/site-packages/scipy/__init__.py:1
6: UserWarning: A NumPy version >=1.16.5 and <1.23.0 is requi
red for this version of SciPy (detected version 1.23.5
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_
maxversion}"
```

In [2]:
```
pd.set_option('display.max_columns', None)
```

In [3]:
```
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, classification_
report
from sklearn.model_selection import train_test_split
```

In [4]:
```
import tensorflow as tf

import keras

from keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_seque
nces

from keras.models import Model, Sequential
from keras.layers import GRU, Input, Dense, Activation, Repea
tVector, Bidirectional, LSTM, Dropout, Embedding
from keras.layers import Embedding

from keras.losses import sparse_categorical_crossentropy
from keras.preprocessing.text import Tokenizer
from keras.preprocessing import sequence
from keras.callbacks import EarlyStopping

from tensorflow.python.client import device_lib
tf.compat.v1.logging.set_verbosity(tf.compat.v1.logging.ERRO
R)
SEED = 10
```

In [5]:
```
data = pd.read_csv('/kaggle/input/fake-news-classification/LFake_Dataset.csv')
data = data[data.columns[1:]]
data[['title', 'text']] = data[['title', 'text']].fillna(value = " ")
data.head()
```

Out[5]:

	title	text	label
0	LAW ENFORCEMENT ON HIGH ALERT Following Threat...	No comment is expected from Barack Obama Membe...	1
1		Did they post their votes for Hillary already?	1
2	UNBELIEVABLE! OBAMA'S ATTORNEY GENERAL SAYS MO...	Now, most of the demonstrators gathered last ...	1
3	Bobby Jindal, raised Hindu, uses story of Chri...	A dozen politically active pastors came here f...	0
4	SATAN 2: Russia unvelis an image of its terrif...	The RS-28 Sarmat missile, dubbed Satan 2, will...	1

In [6]:
```
data['text'] = data['title'] + ' ' + data['text']

X = data['text']
y = data['label']
```

In [7]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, stratify = y, random_state = SEED)
```

In [8]:
```
tok = Tokenizer()
tok.fit_on_texts(X_train)

sequences = tok.texts_to_sequences(X_train)
test_sequences = tok.texts_to_sequences(X_test)

print(f'TRAIN VOCABULARY SIZE: {len(tok.word_index)}')
```

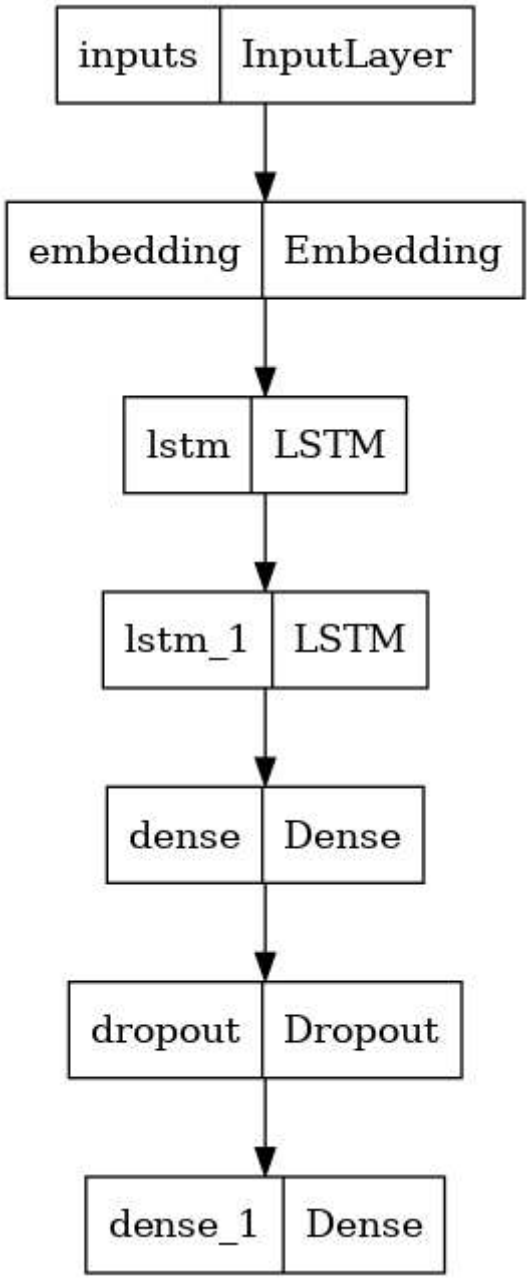
```
TRAIN VOCABULARY SIZE: 282658
```

In [9]:
```
MAX_LEN = 500

X_train_seq = pad_sequences(sequences,maxlen=MAX_LEN)
X_test_seq = pad_sequences(test_sequences,maxlen=MAX_LEN)
```

In [10]:
```
model = tf.keras.Sequential([
    Input(name='inputs',shape=[MAX_LEN]),
    Embedding(len(tok.word_index), 128),
    tf.keras.layers.LSTM(128,  return_sequences=True),
    tf.keras.layers.LSTM(64),
    Dense(64, activation='relu'),
    Dropout(0.5),
    Dense(1, activation='sigmoid')
])

model.compile(loss=tf.keras.losses.BinaryCrossentropy(),
              optimizer=tf.keras.optimizers.Adam(1e-4),
              metrics=['accuracy'])
```

In [11]:
```
tf.keras.utils.plot_model(model)
```

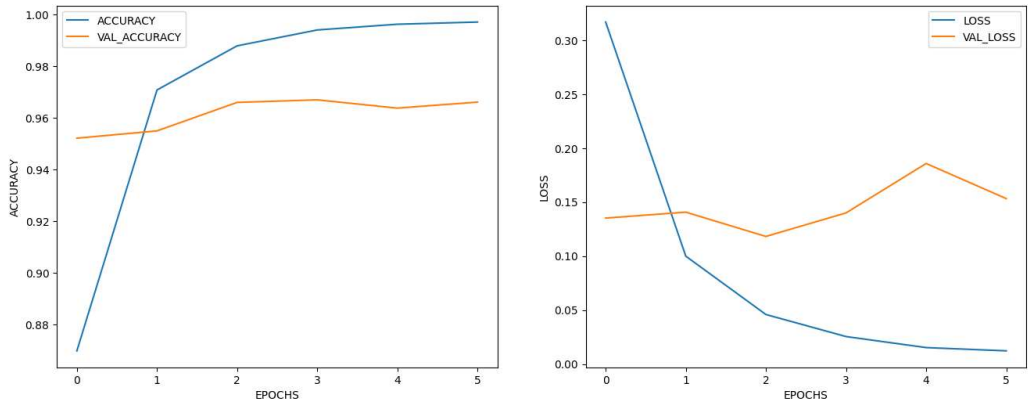
Out[11]:



In [12]:
```
history = model.fit(X_train_seq, y_train, epochs=10,
                    validation_split = 0.3, batch_size = 64,
callbacks=[
                        EarlyStopping(monitor='val_loss', mod
e='min', patience=3, verbose=1, restore_best_weights=True)
                    ])
```

```
Epoch 1/10
592/592 [==============================] - 136s 215ms/step -
loss: 0.3168 - accuracy: 0.8698 - val_loss: 0.1352 - val_accu
racy: 0.9522
Epoch 2/10
592/592 [==============================] - 82s 138ms/step - l
oss: 0.0999 - accuracy: 0.9709 - val_loss: 0.1407 - val_accur
acy: 0.9550
Epoch 3/10
592/592 [==============================] - 60s 102ms/step - l
oss: 0.0458 - accuracy: 0.9880 - val_loss: 0.1182 - val_accur
acy: 0.9661
Epoch 4/10
592/592 [==============================] - 47s 79ms/step - lo
ss: 0.0254 - accuracy: 0.9941 - val_loss: 0.1400 - val_accura
cy: 0.9670
Epoch 5/10
592/592 [==============================] - 44s 75ms/step - lo
ss: 0.0152 - accuracy: 0.9963 - val_loss: 0.1858 - val_accura
cy: 0.9638
Epoch 6/10
592/592 [==============================] - ETA: 0s - loss: 0.
0122 - accuracy: 0.9972Restoring model weights from the end o
f the best epoch: 3.
592/592 [==============================] - 40s 67ms/step - lo
ss: 0.0122 - accuracy: 0.9972 - val_loss: 0.1532 - val_accura
cy: 0.9662
Epoch 6: early stopping
```

In [13]:
```
plt.figure(figsize=(16, 6))
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.xlabel("EPOCHS")
plt.ylabel("ACCURACY")
plt.legend(['ACCURACY', 'VAL_ACCURACY'])

plt.subplot(1, 2, 2)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.xlabel("EPOCHS")
plt.ylabel("LOSS")
plt.legend(['LOSS', 'VAL_LOSS'])
plt.show()
```



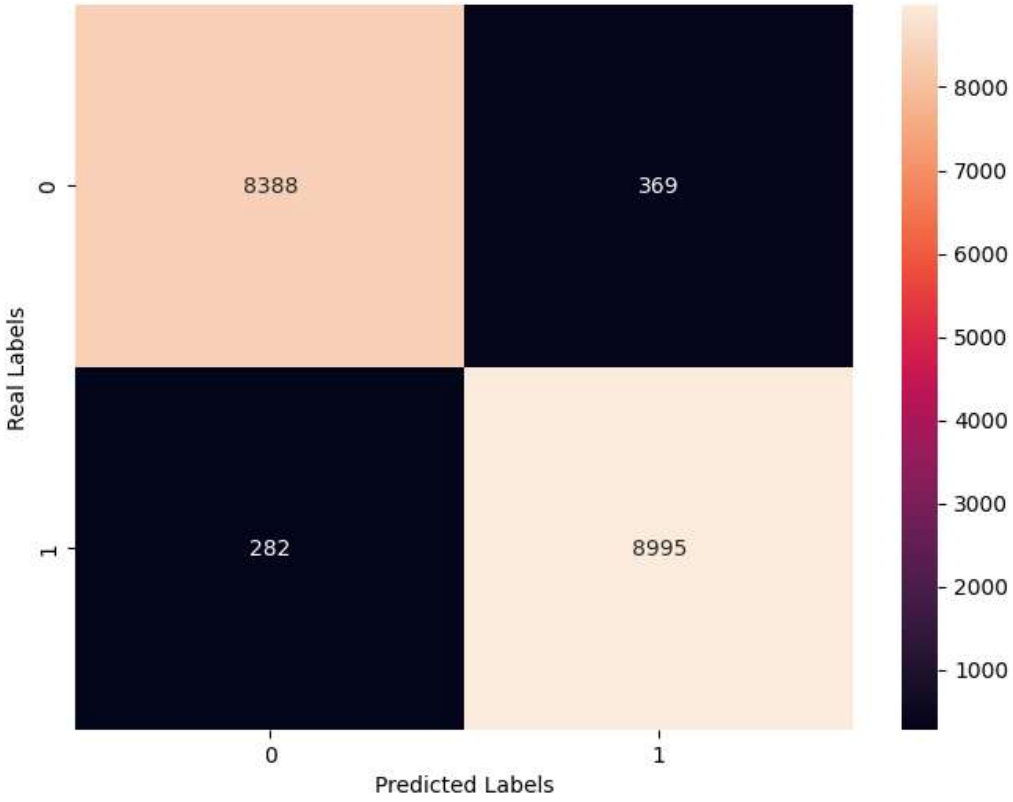
In [14]:
```
test_loss, test_acc = model.evaluate(X_test_seq, y_test)

print('Test Loss:', test_loss)
print('Test Accuracy:', test_acc)
```

```
564/564 [==============================] - 8s 14ms/step - l
s: 0.1255 - accuracy: 0.9639
Test Loss: 0.1255311220884323
Test Accuracy: 0.9639015197753906
```

In [15]:
```
y_hat = model.predict(X_test_seq)

plt.figure(figsize = (8, 6))
sns.heatmap(confusion_matrix(y_test, np.where(y_hat >= 0.5,
1, 0)), annot=True, fmt='')

plt.xlabel('Predicted Labels')
plt.ylabel('Real Labels')
```

```
564/564 [==============================] - 8s 12ms/step
```

Out[15]:
```
Text(70.72222222222221, 0.5, 'Real Labels')
```



FAKE HINDI NEWS CLASSIFIER

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import collections
```

```
/opt/conda/lib/python3.10/site-packages/scipy/__init__.py:1
6: UserWarning: A NumPy version >=1.16.5 and <1.23.0 is requi
red for this version of SciPy (detected version 1.23.5
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_
maxversion}"
```

In [2]:
```
pd.set_option('display.max_columns', None)
```

In [3]:
```
import tensorflow as tf

import keras

from keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_seque
nces
from tensorflow.keras.preprocessing.text import one_hot

from keras.models import Model, Sequential
from keras.layers import Input, Dense, Activation, LSTM, Drop
out, Embedding

from keras.losses import sparse_categorical_crossentropy
from keras.preprocessing.text import Tokenizer
from keras.preprocessing import sequence
from keras.callbacks import EarlyStopping

from tensorflow.python.client import device_lib
tf.compat.v1.logging.set_verbosity(tf.compat.v1.logging.ERRO
R)
SEED = 10
```

```
/opt/conda/lib/python3.10/site-packages/tensorflow_io/pytho
ops/__init__.py:98: UserWarning: unable to load libtensorflow
_io_plugins.so: unable to open file: libtensorflow_io_plugin
s.so, from paths: ['/opt/conda/lib/python3.10/site-packages/t
ensorflow_io/python/ops/libtensorflow_io_plugins.so']
caused by: ['/opt/conda/lib/python3.10/site-packages/tensorfl
ow_io/python/ops/libtensorflow_io_plugins.so: undefined symbo
l: _ZN3tsl6StatusC1EN10tensorflow5error4CodeESt17basic_string
_viewIcSt11char_traitsIcEENS_14SourceLocationE']
  warnings.warn(f"unable to load libtensorflow_io_plugins.so:
{e}")
/opt/conda/lib/python3.10/site-packages/tensorflow_io/python/
ops/__init__.py:104: UserWarning: file system plugins are not
loaded: unable to open file: libtensorflow_io.so, from paths:
['/opt/conda/lib/python3.10/site-packages/tensorflow_io/pytho
n/ops/libtensorflow_io.so']
caused by: ['/opt/conda/lib/python3.10/site-packages/tensorfl
ow_io/python/ops/libtensorflow_io.so: undefined symbol: _ZTVN
10tensorflow13GcsFileSystemE']
  warnings.warn(f"file system plugins are not loaded: {e}")
```

In [4]:
```
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, classification_
report
from sklearn.model_selection import train_test_split
```

In [ ]:
```
!pip install indic-nlp-library
```

In [6]:
```
import nltk
import re
from nltk.corpus import stopwords
import nltk

import string

from nltk.corpus import stopwords
from indicnlp.tokenize import sentence_tokenize, indic_tokeni
ze
```

In [7]:
```
df = pd.read_csv('/kaggle/input/combined/combined_news_data
t.csv')
df.head()
```

Out[7]:

	Unnamed: 0	short_description	label
0	394	बूम एम्स दिल्ल के पैथोलॉज विभाग के वरिष्ठ प्रॉ...	1
1	538	शीतल आमट पहल सोशल मीडिय सेव समित के प्रबंधन सव...	0
2	888	मिसाइल 400 किलोमीटर ज़्याद मार सक	0
3	357	उत्तर प्रदेश के संभल ज़िल किसान शांत भंग आशंक ...	0
4	600	बूम पाय वायरल वीडिय कुमार नरेंद्र मोद जिक्र रह...	1

In [8]:
```
df = df.dropna()
```

In [9]:
```
column_names = df.columns
print(column_names)
```

```
Index(['Unnamed: 0', 'short_description', 'label'], dtype='
ject')
```

In [10]:
```
X = df.drop('label', axis = 1)
y = df['label']
```

In [11]:
```
X.shape, y.shape
```

Out[11]:
```
((2010, 2), (2010,))
```

In [12]:
```
voc_size = 5000
```

In [13]:
```
messages = X.copy()
```

In [14]:
```
messages['short_description'][2]
```

Out[14]:
```
'मिसाइल 400 किलोमीटर ज़्याद मार सक '
```

In [15]:
```
messages.reset_index(inplace=True)
```

In [16]:
```
def preprocess_text(text):
    sentences = sentence_tokenize.sentence_split(text, lang
='hi')
    tokens = [indic_tokenize.trivial_tokenize(sentence) for s
entence in sentences]
    tokens = [token for sublist in tokens for token in sublis
t]
    tokens = [token for token in tokens if token not in strin
g.punctuation and not token.isdigit()]

    stopwords_hi = ['तुम','मेरी','मुझे','क्योंकि','हम','प्रति','अब
की','आगे','माननीय','शहर','बताएं','कौनसी','क्लिक','किसकी','ब
ड़े','मैं','and','रही','आज','लें','आपके','मिलकर','स
ब','मेरे','जी','श्री','वैसा','आपका','अंदर', 'अत', 'अपना', 'अपनी',
'अपने', 'अभी', 'आदि', 'आप', 'इत्यादि', 'इन', 'इनका', 'इन्हीं', 'इ
न्हें', 'इन्हों', 'इस', 'इसका', 'इसकी', 'इसके', 'इसमें', 'इसी', 'इसे',
'उन', 'उनका', 'उनकी', 'उनके', 'उनको', 'उन्हीं', 'उन्हें', 'उन्हों', 'उ
स', 'उसके', 'उसी', 'उसे', 'एक', 'एवं', 'एस', 'ऐसे', 'और', 'क
ई', 'कर','करता', 'करते', 'करना', 'करने', 'करें', 'कहते', 'कहा',
'का', 'काफ़ी', 'कि', 'कितना', 'किन्हें', 'किन्हों', 'किया', 'किर',
'किस', 'किसी', 'किसे', 'की', 'कुछ', 'कुल', 'के', 'को', 'कोई',
'कौन', 'कौनसा', 'गया', 'घर', 'जब', 'जहाँ', 'जा', 'जितना', 'जिन',
'जिन्हें', 'जिन्हों', 'जिस', 'जिसे', 'जीधर', 'जैसा', 'जैसे', 'जो', 'तक',
'तब', 'तरह', 'तिन', 'तिन्हें', 'तिन्हों', 'तिस', 'तिसे', 'तो', 'था',
'थी', 'थे', 'दबारा', 'दिया', 'दुसरा', 'दूसरे', 'दो', 'द्वारा', 'न', 'न
हीं', 'ना', 'निहायत', 'नीचे', 'ने', 'पर', 'पर', 'पहले', 'पूरा', 'पे',
'फिर', 'बनी', 'बही', 'बहुत', 'बाद', 'बाला', 'बिलकुल', 'भी', 'भीत
र', 'मगर', 'मानो', 'मे', 'में', 'यदि', 'यह', 'यहाँ', 'यही', 'या',
'यिह', 'ये', 'रखें', 'रहा', 'रहे', 'ऱ्वासा', 'लिए', 'लिये', 'लेकिन',
'व', 'वर्ग', 'वह', 'वह', 'वहाँ', 'वहीं', 'वाले', 'वुह', 'वे', 'वग़ैरह',
'संग', 'सकता', 'सकते', 'सबसे', 'सभी', 'साथ', 'साबुत', 'साभ',
'सारा', 'से', 'सो', 'ही', 'हुआ', 'हुई', 'हुए', 'है', 'हैं', 'हो',
'होता', 'होती', 'होते', 'होना', 'होने', 'अपनि', 'जेसे', 'होति', 'सभि',
'तिंहों', 'इंहों', 'दवारा', 'इसि', 'किंहें', 'थि', 'उंहों', 'ओर', 'जिंहें',
'वहिं', 'अभि', 'बनि', 'हि', 'उंहिं', 'उंहें', 'हें', 'वगेरह', 'एसे', 'र
वासा', 'कोन', 'निचे', 'काफि', 'उसि', 'पुरा', 'भितर', 'हे', 'बहि',
'वहां', 'कोइ', 'यहां', 'जिंहों', 'तिंहें', 'किसि', 'कइ', 'यहि', 'इंहिं',
'जिधर', 'इंहें', 'अदि', 'इतयादि', 'हुइ', 'कोनसा', 'इसकि', 'दुसरे', 'ज
हां', 'अप', 'किंहों', 'उनकि', 'भि', 'वरग', 'हुअ', 'जेसा', 'नहिं']
    stopwords_en = ['i', 'me', 'my', 'myself', 'we', 'our',
'ours', 'ourselves', 'you', "you're", "you've", "you'll", "yo
u'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him',
'his', 'himself', 'she', "she's", 'her', 'hers', 'herself',
'it', "it's", 'its', 'itself', 'they', 'them', 'their', 'thei
rs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 't
hat', "that'll", 'these', 'those', 'am', 'is', 'are', 'was',
'were', 'be', 'been', 'being', 'have', 'has', 'had', 'havin
g', 'do', 'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'b
ut', 'if', 'or', 'because', 'as', 'until', 'while', 'of', 'a
t', 'by', 'for', 'with', 'about', 'against', 'between', 'int
o', 'through', 'during', 'before', 'after', 'above', 'below',
'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over',
'under', 'again', 'further', 'then', 'once', 'here', 'there',
'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each',
'few', 'more', 'most', 'other', 'some', 'such', 'no', 'nor',
'not', 'only', 'own', 'same', 'so', 'than', 'too', 'very',
's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "s
hould've", 'now', 'd', 'll', 'm', 'o', 're', 've', 'y', 'ai
n', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't",
'doesn', "doesn't", 'hadn', "hadn't", 'hasn', "hasn't", 'have
```

```
n', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'm
ustn', "mustn't", 'needn', "needn't", 'shan', "shan't", 'shou
ldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't", 'wo
n', "won't", 'wouldn', "wouldn't"]
    stop_words = stopwords_hi + stopwords_en
    tokens = [token for token in tokens if token not in stop_
words]
    cleaned_text = ' '.join(tokens)
    return cleaned_text

df['cleaned_text'] = df['short_description'].apply(preprocess
_text)
```

In [17]:
```
corpus = df['cleaned_text']
```

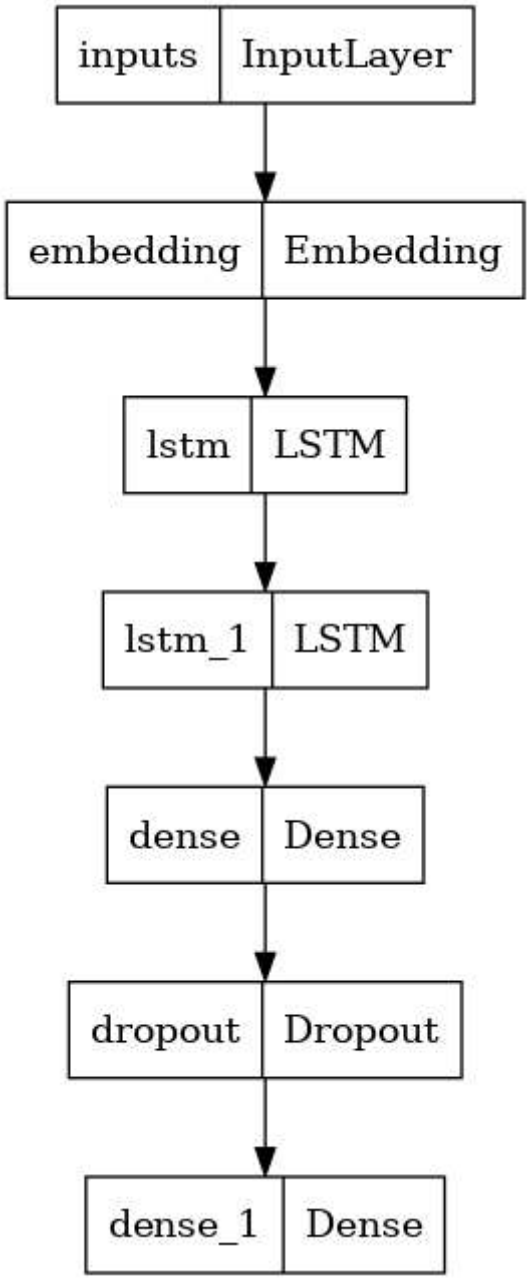
In [18]:
```
onehot_repr = [one_hot(words, voc_size) for words in corpus
```

In [19]:
```
sent_length = 20
embedded_docs = pad_sequences(onehot_repr, padding = 'pre', m
axlen = sent_length)
```

In [20]:
```
model = tf.keras.Sequential([
    Input(name='inputs',shape=[sent_length]),
    Embedding(voc_size, 40),
    tf.keras.layers.LSTM(128,  return_sequences=True),
    tf.keras.layers.LSTM(64),
    Dense(64, activation='relu'),
    Dropout(0.5),
    Dense(1, activation='sigmoid')
])

model.compile(loss=tf.keras.losses.BinaryCrossentropy(),
              optimizer=tf.keras.optimizers.Adam(1e-4),
              metrics=['accuracy'])
```

In [21]:
```
tf.keras.utils.plot_model(model)
```

Out[21]:



In [22]:
```
X_final = np.array(embedded_docs)
y_final = np.array(y)
```

In [23]:
```
X_final.shape, y_final.shape
```

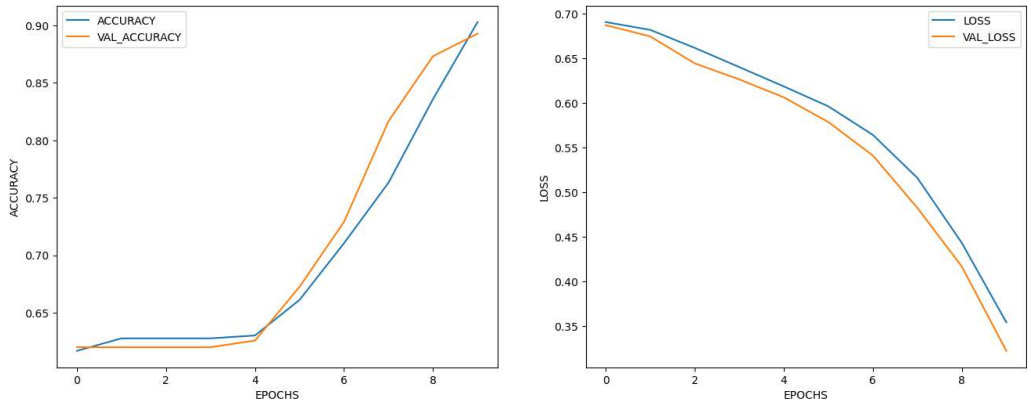
Out[23]:
```
((2010, 20), (2010,))
```

In [24]:
```
X_train, X_test, y_train, y_test = train_test_split(X_final
y_final, test_size=0.15, random_state=42)
```

In [25]:
```
history = model.fit(X_train, y_train, epochs=10,
                    validation_split = 0.3, batch_size = 64,
callbacks=[
                        EarlyStopping(monitor='val_loss', mod
e='min', patience=3, verbose=1, restore_best_weights=True)
                    ])
```

```
Epoch 1/10
19/19 [==============================] - 7s 114ms/step - los
s: 0.6904 - accuracy: 0.6167 - val_loss: 0.6869 - val_accurac
y: 0.6199
Epoch 2/10
19/19 [==============================] - 1s 61ms/step - loss:
0.6816 - accuracy: 0.6276 - val_loss: 0.6743 - val_accuracy:
0.6199
Epoch 3/10
19/19 [==============================] - 1s 62ms/step - loss:
0.6614 - accuracy: 0.6276 - val_loss: 0.6440 - val_accuracy:
0.6199
Epoch 4/10
19/19 [==============================] - 1s 63ms/step - loss:
0.6399 - accuracy: 0.6276 - val_loss: 0.6262 - val_accuracy:
0.6199
Epoch 5/10
19/19 [==============================] - 1s 62ms/step - loss:
0.6183 - accuracy: 0.6301 - val_loss: 0.6061 - val_accuracy:
0.6257
Epoch 6/10
19/19 [==============================] - 1s 64ms/step - loss:
0.5960 - accuracy: 0.6611 - val_loss: 0.5784 - val_accuracy:
0.6725
Epoch 7/10
19/19 [==============================] - 1s 62ms/step - loss:
0.5641 - accuracy: 0.7105 - val_loss: 0.5410 - val_accuracy:
0.7290
Epoch 8/10
19/19 [==============================] - 1s 62ms/step - loss:
0.5159 - accuracy: 0.7632 - val_loss: 0.4824 - val_accuracy:
0.8168
Epoch 9/10
19/19 [==============================] - 1s 61ms/step - loss:
0.4431 - accuracy: 0.8360 - val_loss: 0.4166 - val_accuracy:
0.8733
Epoch 10/10
19/19 [==============================] - 1s 62ms/step - loss:
0.3542 - accuracy: 0.9029 - val_loss: 0.3223 - val_accuracy:
0.8928
```

In [26]:
```
plt.figure(figsize=(16, 6))
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.xlabel("EPOCHS")
plt.ylabel("ACCURACY")
plt.legend(['ACCURACY', 'VAL_ACCURACY'])

plt.subplot(1, 2, 2)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.xlabel("EPOCHS")
plt.ylabel("LOSS")
plt.legend(['LOSS', 'VAL_LOSS'])
plt.show()
```



In [27]:
```
test_loss, test_acc = model.evaluate(X_test, y_test)

print('Test Loss:', test_loss)
print('Test Accuracy:', test_acc)
```

```
10/10 [==============================] - 0s 14ms/step - los
0.3446 - accuracy: 0.8775
Test Loss: 0.34456345438957214
Test Accuracy: 0.8774834275245667
```

In [28]:
```
y_hat = model.predict(X_test)

plt.figure(figsize = (8, 6))
sns.heatmap(confusion_matrix(y_test, np.where(y_hat >= 0.5,
1, 0)), annot=True, fmt='')

plt.xlabel('Predicted Labels')
plt.ylabel('Real Labels')
```

```
10/10 [==============================] - 1s 14ms/step
```

Out[28]:
```
Text(70.72222222222221, 0.5, 'Real Labels')
```