Fake News Detection Using NLP

Model Development & Model Evaluation

Model Development and Evaluation:

Model development typically refers to the process of creating, training, and refining a mathematical or computational model to solve a specific problem or make predictions based on data. This process is commonly used in various fields, including machine learning, statistics, engineering, and the natural sciences.

1. Model Selection: Choose an appropriate modelling technique based on the nature of the problem. Common modelling techniques include linear regression, decision trees, neural networks, support vector machines, and more.
2. Feature Selection: Select the most relevant features or variables from your data that are likely to have a significant impact on the model's performance. Feature selection helps reduce dimensionality and improve model efficiency.
3. Model Training: Use the prepared data to train the selected model. This process involves optimizing the model's parameters to make predictions as accurate as possible. For machine learning models, this typically involves using a training dataset.
4. Model Evaluation: Assess the model's performance using evaluation metrics appropriate to the problem, such as accuracy, precision, recall, F1-score, mean squared error, etc. Cross-validation or holdout testing datasets are commonly used to evaluate model performance.
5. Model Validation: Ensure that the model performs well on new, unseen data. Overfitting, where a model performs well on the training data but poorly on new data, should be avoided. Regularization and validation techniques help prevent overfitting.
6. Model Refinement: If the model's performance is unsatisfactory, you may need to refine it. This could involve tuning hyperparameters, collecting more data, changing the model architecture, or using a different modelling technique.

7. Interpretation and Visualization: Understand the model's internal workings and make sense of its predictions. Visualize the results and insights to communicate findings effectively.
8. Deployment: If the model meets your requirements, deploy it for use in your application or decision-making process. This could involve integration into software, automation, or providing recommendations based on the model's output.
9. Maintenance and Monitoring: Regularly monitor the model's performance in a production environment, as data distributions may change over time. Re-train or update the model as needed to maintain its accuracy and relevance.

Model development is an iterative process, and it may involve going back and forth between these steps as you gain a better understanding of the system you're modelling and how to improve the model's performance. Additionally, ethical considerations, data privacy, and bias mitigation should be addressed throughout the model development process.

Model Building (LSTM)

In [34]:
```
max_features = 10000
maxlen = 300
```

Tokenization

In [35]:
```
tokenizer = text.Tokenizer(num_words=max_features)
tokenizer.fit_on_texts(X_train)
tokenized_train = tokenizer.texts_to_sequences(X_train)
X_train = sequence.pad_sequences(tokenized_train, maxlen=maxlen)
```

In [36]:
```
tokenized_test = tokenizer.texts_to_sequences(X_test)
X_test = sequence.pad_sequences(tokenized_test, maxlen=maxlen)
```

In [37]:
```
batch_size = 256
epochs = 3
embed_size = 100
```

Model Development

In [38]:
```
model = Sequential()
#Non-trainable embeddidng layer
model.add(Embedding(max_features, output_dim=embed_size, input_length=maxlen, trainable=False))
#LSTM
model.add(LSTM(units=128 , return_sequences = True , recurrent_dropout = 0.25 , dropout = 0.25))
model.add(LSTM(units=64 , recurrent_dropout = 0.1 , dropout = 0.1))
model.add(Dense(units = 32 , activation = 'relu'))
model.add(Dense(1, activation='sigmoid'))
model.compile(optimizer=keras.optimizers.Adam(lr = 0.01), loss='binary_crossentropy', metrics=['accuracy'])
```

```
WARNING:absl:`lr` is deprecated in Keras optimizer, please use `learning_rate` or use the legacy optimizer, e.g.,tf.keras.optimizers.legacy.Adam.
```

In [39]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 300, 100)          1000000

 lstm (LSTM)                 (None, 300, 128)          117248

 lstm_1 (LSTM)               (None, 64)                49408

 dense (Dense)               (None, 32)                2080

 dense_1 (Dense)             (None, 1)                 33

=================================================================
Total params: 1168769 (4.46 MB)
Trainable params: 168769 (659.25 KB)
Non-trainable params: 1000000 (3.81 MB)
_________________________________________________________________
```

Evaluation of Model

Training Model

In [40]:
```
history = model.fit(X_train, y_train, validation_split=0.3, epochs=3, batch_size=batch_size, shuffle=True, verbose = 1)
```

```
Epoch 1/3
92/92 [==============================] - 2766s 30s/step - loss: 0.5078 - accuracy: 0.7501 - val_loss: 0.3909 - val_accuracy: 0.
8483
Epoch 2/3
92/92 [==============================] - 3924s 43s/step - loss: 0.3860 - accuracy: 0.8269 - val_loss: 0.2903 - val_accuracy: 0.
8797
Epoch 3/3
92/92 [==============================] - 2090s 23s/step - loss: 0.3485 - accuracy: 0.8425 - val_loss: 0.2189 - val_accuracy: 0.
9116
```

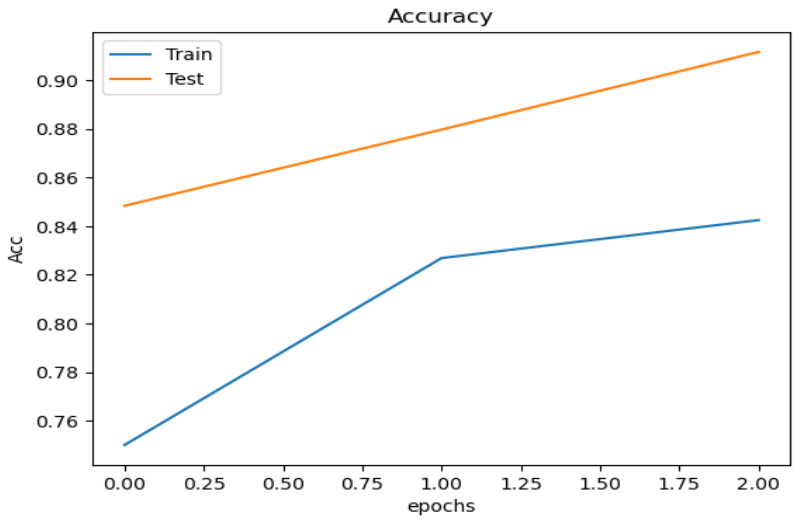
Evaluating Accuracy Of Model

In [42]:
```
print("Accuracy of the model on Training Data is - " , model.evaluate(X_train,y_train)[1]*100 , "%")
print("Accuracy of the model on Testing Data is - " , model.evaluate(X_test,y_test)[1]*100 , "%")
```

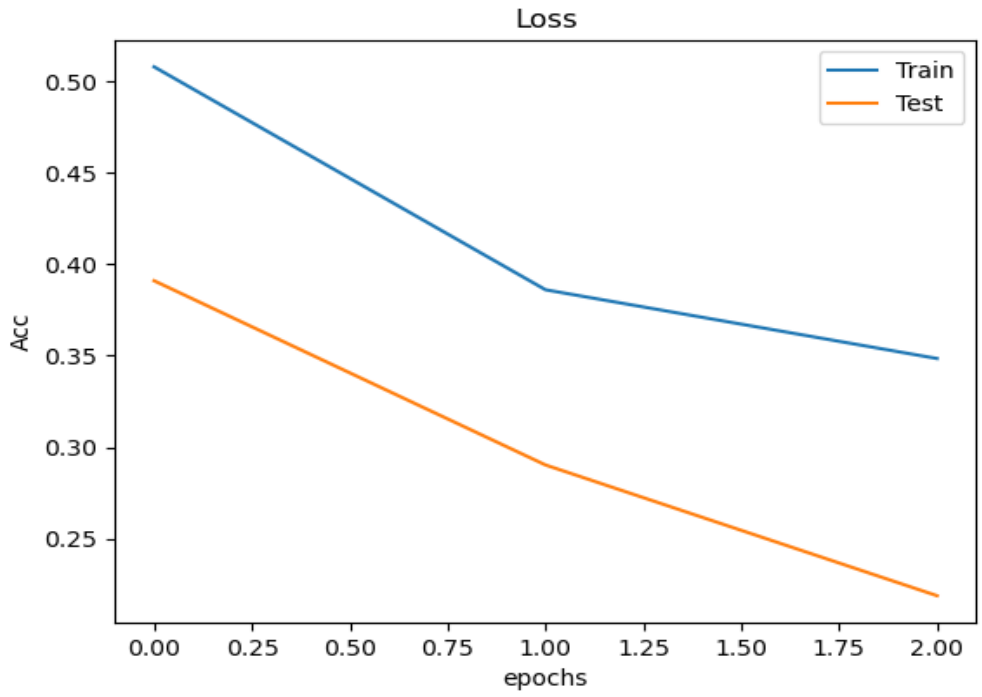
```
1048/1048 [==============================] - 111s 106ms/step - loss: 0.2139 - accuracy: 0.9169
Accuracy of the model on Training Data is -  91.68755412101746 %
350/350 [==============================] - 37s 105ms/step - loss: 0.2150 - accuracy: 0.9159
Accuracy of the model on Testing Data is -  91.58686399459839 %
```

Model Visualization

In [43]:
```
plt.figure()
plt.plot(history.history["accuracy"], label = "Train")
plt.plot(history.history["val_accuracy"], label = "Test")
plt.title("Accuracy")
plt.ylabel("Acc")
plt.xlabel("epochs")
plt.legend()
plt.show()
```



In [44]:
```
plt.figure()
plt.plot(history.history["loss"], label = "Train")
plt.plot(history.history["val_loss"], label = "Test")
plt.title("Loss")
plt.ylabel("Acc")
plt.xlabel("epochs")
plt.legend()
plt.show()
```



Prediction with Evaluating Precision

In [45]:
```
pred = (model.predict(X_test)>0.5).astype("int32")
print(classification_report(y_test, pred, target_names = ['Fake','Real']))
```

```
350/350 [==============================] - 39s 110ms/step
              precision    recall  f1-score   support

        Fake       0.91      0.91      0.91      5303
        Real       0.92      0.92      0.92      5870

    accuracy                           0.92     11173
   macro avg       0.92      0.92      0.92     11173
weighted avg       0.92      0.92      0.92     11173
```