Song classification into gender using Recurrent Neural Network deep

In [76]:
```
import numpy as np
import pandas as pd
import re
import nltk
from nltk.corpus import stopwords
import matplotlib.pyplot as plt
import string
import multiprocessing
import os
import sklearn
import pprint
import seaborn as sns
nltk.download('stopwords')
%matplotlib inline
stop = stopwords.words('english')
from subprocess import check_output
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\LENOVO\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

In [77]:
```
df1= pd.read_csv('lyrics_6.csv')
df1.head(3)
len(df1)
df1
```

Out[77]:

	Artist	Gender	Lyrics
0	The Weeknd	male	Yeah\r\r\nYeah\r\r\nYeah\r\r\nYour man on the ...
1	The Weeknd	male	I'm tryna put you in the worst mood, ah\r\r\nP...
2	The Weeknd	male	And I know she'll be the death of me\r\r\nAt l...
3	The Weeknd	male	We found each other\r\r\nI helped you out of a...
4	The Weeknd	male	NaN
...	...	...	...
1045	Taylor Swift	Female	Once upon a time, a few mistakes ago\r\r\nI wa...
1046	Taylor Swift	Female	We're all bored, we're all so tired of everyth...
1047	Taylor Swift	Female	We could leave the Christmas lights up 'til Ja...
1048	Taylor Swift	Female	It feels like a perfect night\r\r\nTo dress up...
1049	Taylor Swift	Female	Flashing lights, and we\r\r\nTook a wrong turn...

1050 rows × 3 columns

In [78]:
```
df1['Lyrics']=df1['Lyrics'].astype(str)
```

Removing special characters

In [79]:
```
REPLACE_BY_SPACE_RE = re.compile('[/(){}\[\]\|@,;]')
BAD_SYMBOLS_RE = re.compile('[^0-9a-z #+_]')
STOPWORDS = set(stopwords.words('english'))

def clean_text(text):
    """
        text: a string

        return: modified initial string
    """
    text = text.lower() # lowercase text
    text = REPLACE_BY_SPACE_RE.sub(' ', text) # replace REPLACE_BY_SPACE_RE s
    text = BAD_SYMBOLS_RE.sub('', text) # remove symbols which are in BAD_SYM
    text = text.replace('x', '')
#    text = re.sub(r'\W+', '', text)
    text = ' '.join(word for word in text.split() if word not in STOPWORDS) #
    return text
df1['Lyrics'] = df1['Lyrics'].apply(clean_text)
```

In [80]:
```
df1
```

Out[80]:

	Artist	Gender	Lyrics
0	The Weeknd	male	yeahyeahyeahyour man road doin promoyou said k...
1	The Weeknd	male	im tryna put worst mood ahp1 cleaner church sh...
2	The Weeknd	male	know shell death meat least well numband shell...
3	The Weeknd	male	found otheri helped broken placeyou gave comfo...
4	The Weeknd	male	nan
...	...	...	...
1045	Taylor Swift	Female	upon time mistakes agoi sights got aloneyou fo...
1046	Taylor Swift	Female	bored tired everythingwe wait trains arent com...
1047	Taylor Swift	Female	could leave christmas lights til januaryand pl...
1048	Taylor Swift	Female	feels like perfect nightto dress like hipsters...
1049	Taylor Swift	Female	flashing lights wetook wrong turn wefell rabbi...

1050 rows × 3 columns

Dictionary generation for songs

In [81]:
```
from keras.preprocessing.text import Tokenizer
# The maximum number of words to be used. (most frequent)
vacob_size = 5000
# Max number of words in each song.
song_length = 500
# This is fixed.
embedding_dim= 100
tokenizer = Tokenizer(num_words=vacob_size, filters='!"#$%&()*+,-./:;<=>?@[\]
tokenizer.fit_on_texts(df1['Lyrics'].values)
word_index = tokenizer.word_index
print('Found %s unique tokens.' % len(word_index))
```

```
Found 33294 unique tokens.
```

In [82]:
```
from keras.preprocessing.sequence import pad_sequences
X = tokenizer.texts_to_sequences(df1['Lyrics'].values)
X = pad_sequences(X, maxlen=song_length)
print('Shape of data tensor:', X.shape)
```

```
Shape of data tensor: (1050, 500)
```

In [84]:
```
Y = pd.get_dummies(df1['Gender']).values
print('Shape of label tensor:', Y.shape)
Y
```

```
Shape of label tensor: (1050, 2)
```

Out[84]:
```
array([[0, 1],
       [0, 1],
       [0, 1],
       ...,
       [1, 0],
       [1, 0],
       [1, 0]], dtype=uint8)
```

Splitting data into train and test

In [85]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size = 0.20, ra
print(X_train.shape,Y_train.shape)
print(X_test.shape,Y_test.shape)
```

```
(840, 500) (840, 2)
(210, 500) (210, 2)
```

RNN (LSTM) deep learning modelling

In [86]:
```
from keras import Sequential
from keras.layers import Dense, Embedding, LSTM, GRU, Dropout, Activation
from keras.layers.embeddings import Embedding
from keras.callbacks import EarlyStopping


model = Sequential()
model.add(Embedding(vacob_size, embedding_dim, input_length=X.shape[1]))
model.add(LSTM(100, recurrent_dropout=0.2))
model.add(Dropout(0.2))
model.add(Dense(2))
model.add(Activation('softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['ac
batch_size = 128
```

In [87]:
```

epochs = 5

history = model.fit(X_train, Y_train, epochs=epochs, batch_size=batch_size,va
```

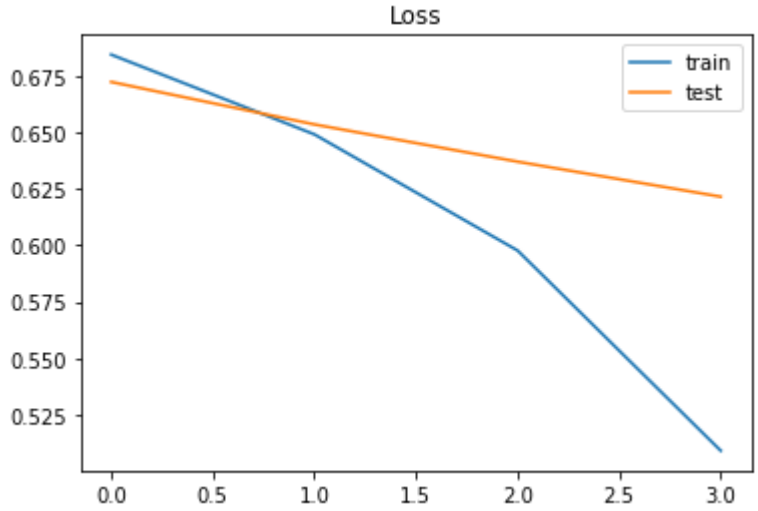
```
C:\Users\LENOVO\Anaconda3\lib\site-packages\tensorflow_core\python\framewor
k\indexed_slices.py:433: UserWarning: Converting sparse IndexedSlices to a
dense Tensor of unknown shape. This may consume a large amount of memory.
  "Converting sparse IndexedSlices to a dense Tensor of unknown shape. "

Train on 672 samples, validate on 168 samples
Epoch 1/4
672/672 [==============================] - 12s 17ms/step - loss: 0.6847 - a
ccuracy: 0.5982 - val_loss: 0.6726 - val_accuracy: 0.5952
Epoch 2/4
672/672 [==============================] - 12s 18ms/step - loss: 0.6493 - a
ccuracy: 0.6161 - val_loss: 0.6537 - val_accuracy: 0.5952
Epoch 3/4
672/672 [==============================] - 13s 20ms/step - loss: 0.5978 - a
ccuracy: 0.6205 - val_loss: 0.6372 - val_accuracy: 0.6012
Epoch 4/4
672/672 [==============================] - 14s 21ms/step - loss: 0.5089 - a
ccuracy: 0.7232 - val_loss: 0.6216 - val_accuracy: 0.6488
```

In [29]:
```
y_p=model.predict(X_test,verbose=0)
y_pclass=model.predict_classes(X_test,verbose=0)
```

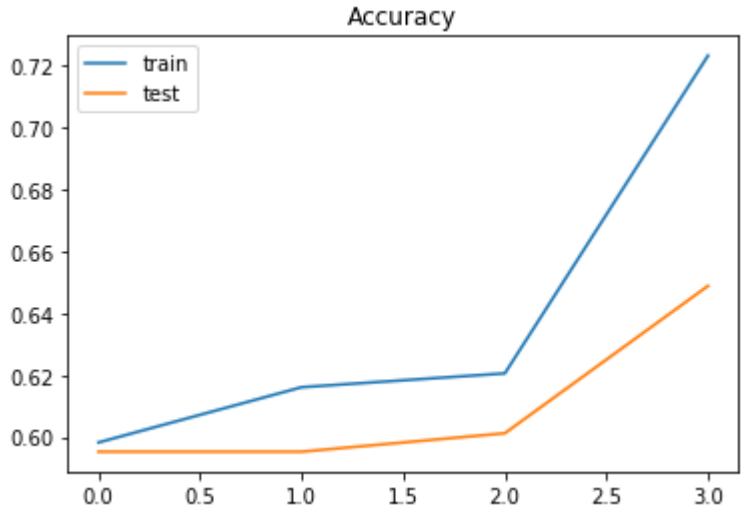
In [90]:
```
accr = model.evaluate(X_test,Y_test)
print('Test set\n  Loss: {:0.3f}\n  Accuracy: {:0.3f}'.format(accr[0],accr[1]
```

```
210/210 [==============================] - 1s 6ms/step
Test set
  Loss: 0.618
  Accuracy: 0.652
```

In [88]:
```
plt.title('Loss')
plt.plot(history.history['loss'], label='train')
plt.plot(history.history['val_loss'], label='test')
plt.legend()
plt.show();
```



In [89]:
```
plt.title('Accuracy')
plt.plot(history.history['accuracy'], label='train')
plt.plot(history.history['val_accuracy'], label='test')
plt.legend()
plt.show();
```



In [ ]:
```
```