Music Genre Classification from Lyrics

Team members:
Animi Reddy
Sri Keshav
Raghuchandra
Sushman

Objective

To predict the genre of a song based on the lyrics.

Data collection

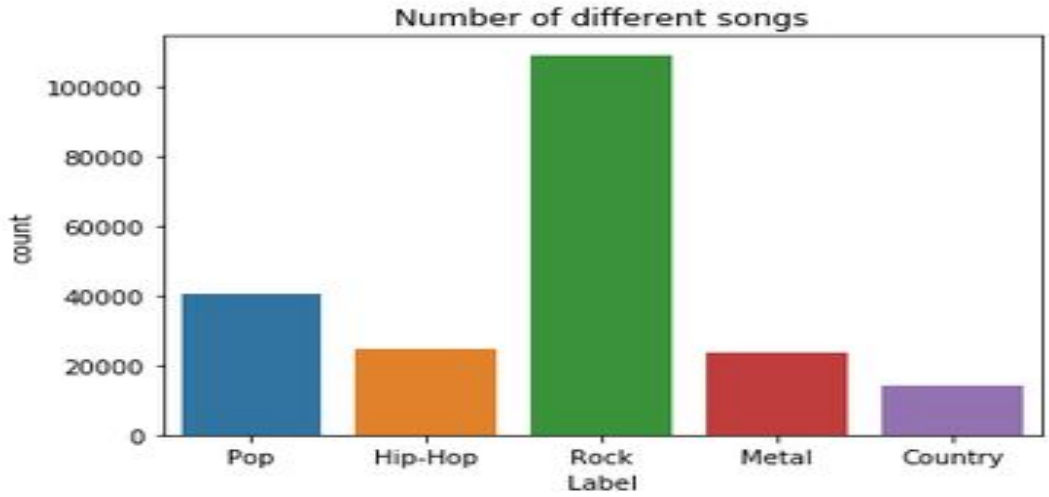
The data was collected from the following website: https://www.kaggle.com/gyani95/380000-lyrics-from-metrolyrics

There are around 380,000+ lyrics in the data set from a lot of different artists from a lot of different genres arranged by year.

Data Analysis

```
sns.countplot(df.genre)
plt.xlabel('Label')
plt.title('Number of different songs')
```

```
Text(0.5, 1.0, 'Number of different songs')
```



Data preprocessing

Our text preprocessing will include the following steps:

- Convert multiline text to single line text by replacing “\n” with “\t”.
- Convert all text to lower case.
- Replace all these symbols '[/(){}\[\]\|@,;]' by space in text.
- Remove all these symbols '[^0-9a-z #+_]' from text.
- Remove stop words.

Model

We have used LSTM recurrent neural network models in Python using Keras deep learning library.

Steps in LSTM modelling:

- Vectorize lyrics, by turning each text into either a sequence of integers or into a vector.
- Limit the data set to the top 50,000 words.
- Set the max number of words in each text at 250.

- Truncate and pad the input sequences so that they are all in the same length for modeling.
- Converting categorical labels to numbers.
- The first layer is the embedded layer that uses 100 length vectors to represent each word.
- SpatialDropout1D performs variational dropout in NLP models.
- The next layer is the LSTM layer with 100 memory units.
- The output layer must create 8 output values, one for each class.
- Activation function is softmax for multi-class classification.
- Because it is a multi-class classification problem, categorical_crossentropy is used as the loss function.

Evaluating the model

Accuracy for the RNN model:

Training(at epoch 5): Accuracy: 0.744, Loss : 0.6020

Testing: Accuracy: 0.732, Loss: 0.790

Accuracy for the SVM model:

Test accuracy: 0.6244

In [15]:
```
epochs = 5
batch_size = 64

history = model.fit(X_train, Y_train, epochs=epochs, batch_size=batch_size,validation_split=0.1,callbacks=[Ea
```

```
WARNING:tensorflow:From /home/animi/.local/lib/python3.5/site-packages/tensorflow/python/ops/math_ops.py:306
6: to_int32 (from tensorflow.python.ops.math_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.cast instead.
Train on 172284 samples, validate on 19143 samples
Epoch 1/5
172284/172284 [==============================] - 1309s 8ms/step - loss: 1.0279 - acc: 0.6124 - val_loss: 0.8
929 - val_acc: 0.6530
Epoch 2/5
172284/172284 [==============================] - 1288s 7ms/step - loss: 0.8624 - acc: 0.6722 - val_loss: 0.8
636 - val_acc: 0.6667
Epoch 3/5
172284/172284 [==============================] - 1566s 9ms/step - loss: 0.7511 - acc: 0.7150 - val_loss: 0.8
246 - val_acc: 0.6892
Epoch 4/5
172284/172284 [==============================] - 1683s 10ms/step - loss: 0.6692 - acc: 0.7482 - val_loss: 0.
8309 - val_acc: 0.6868
Epoch 5/5
172284/172284 [==============================] - 1507s 9ms/step - loss: 0.6020 - acc: 0.7744 - val_loss: 0.8
646 - val_acc: 0.6855
```

In [21]:
```
accr = model.evaluate(X_test,Y_test)
print('Test set\n  Loss: {:0.3f}\n  Accuracy: {:0.3f}'.format(accr[0],accr[1]))
```

```
21270/21270 [==============================] - 37s 2ms/step
Test set
  Loss: 0.790
  Accuracy: 0.732
```

Thank you