In [1]:
```
import os
import json
import numpy as np
import pandas as pd
from keras.models import Sequential
from keras.layers import LSTM, Dropout, Dense, Activation, Embedding
```

```
C:\Users\GauravP\Anaconda3\lib\site-packages\h5py\__init__.py:36: FutureWarning: Conversion of the second argument of i
ssubdtype from `float` to `np.floating` is deprecated. In future, it will be treated as `np.float64 == np.dtype(float).
type`.
  from ._conv import register_converters as _register_converters
Using TensorFlow backend.
```

In [2]:
```
data_directory = "../Data2/"
data_file = "Data_Tunes.txt"
charIndex_json = "char_to_index.json"
model_weights_directory = '../Data2/Model_Weights/'
BATCH_SIZE = 16
SEQ_LENGTH = 64
```

In [3]:
```
def make_model(unique_chars):
    model = Sequential()

    model.add(Embedding(input_dim = unique_chars, output_dim = 512, batch_input_shape = (1, 1)))

    model.add(LSTM(256, return_sequences = True, stateful = True))
    model.add(Dropout(0.2))

    model.add(LSTM(256, return_sequences = True, stateful = True))
    model.add(Dropout(0.2))

    model.add(LSTM(256, stateful = True))
    #remember, that here we haven't given return_sequences = True because here we will give only one character to generat
    #sequence. In the end, we just have to get one output which is equivalent to getting output at the last time-stamp. S
    #in last layer there is no need of giving return sequences = True.
    model.add(Dropout(0.2))

    model.add((Dense(unique_chars)))
    model.add(Activation("softmax"))

    return model
```

In [14]:
```
def generate_sequence(epoch_num, initial_index, seq_length):
    with open(os.path.join(data_directory, charIndex_json)) as f:
        char_to_index = json.load(f)
    index_to_char = {i:ch for ch, i in char_to_index.items()}
    unique_chars = len(index_to_char)

    model = make_model(unique_chars)
    model.load_weights(model_weights_directory + "Weights_{}.h5".format(epoch_num))

    sequence_index = [initial_index]

    for _ in range(seq_length):
        batch = np.zeros((1, 1))
        batch[0, 0] = sequence_index[-1]

        predicted_probs = model.predict_on_batch(batch).ravel()
        sample = np.random.choice(range(unique_chars), size = 1, p = predicted_probs)

        sequence_index.append(sample[0])

    seq = ''.join(index_to_char[c] for c in sequence_index)

    cnt = 0
    for i in seq:
        cnt += 1
        if i == "\n":
            break
    seq1 = seq[cnt:]
    #above code is for ignoring the starting string of a generated sequence. This is because we are passing any arbitrary
    #character to the model for generating music. Now, the model start generating sequence from that character itself whi
    #have passed, so first few characters before "\n" contains meaningless word. Model start generating the music rhythm
    #next line onwards. The correct sequence it start generating from next line onwards which we are considering.

    cnt = 0
    for i in seq1:
        cnt += 1
        if i == "\n" and seq1[cnt] == "\n":
            break
    seq2 = seq1[:cnt]
    #Now our data contains three newline characters after every tune. So, the model has leart that too. So, above code is
    #ignoring all the characters that model has generated after three new line characters. So, here we are considering on
```

```
    #tune of music at a time and finally we are returning it..

    return seq2
```

In [23]:
```
ep = int(input("1. Which epoch number weight you want to load into the model(10, 20, 30, ..., 90). Small number will gene
ar = int(input("\n2. Enter any number between 0 to 86 which will be given as initial charcter to model for generating seq
ln = int(input("\n3. Enter the length of music sequence you want to generate. Typical number is between 300-600. Too smal

music = generate_sequence(ep, ar, ln)

print("\nMUSIC SEQUENCE GENERATED: \n")

print(music)
```

```
1. Which epoch number weight you want to load into the model(10, 20, 30, ..., 90). Small number will generate more erro
rs in music: 90

2. Enter any number between 0 to 86 which will be given as initial charcter to model for generating sequence: 25

3. Enter the length of music sequence you want to generate. Typical number is between 300-600. Too small number will ge
nerate hardly generate any sequence: 450

MUSIC SEQUENCE GENERATED:

"(37)"E2E D2)|"Am"E2c "G7"B=GB|"Am"D2c cBc|
"D"d2A ABd|"G"g2d e2G|"Am"B2A "D7"A2G|"G"G3 -G2:|
P:B
|:A|"G"BGD "D7"G2A|"G"BGB dBd|"C"efg "B7"b2g|"Em"gfe "Am"dBG|
"D"DEF AGF|"G"GBd g2a|"G"g2d "D7"c2B|"G"G3 -G2:|
P:B
d|"G"dBd gfg|"C"e^de g2e|"F"dBA "D7"ABA|"G"G3 G2:|
```