Modelagem de Letras de Músicas - Criação de Letra com Inteligência Artificial - CBJR

Benicio Ramos Magalhaes

Motivação:

Realizar uma primeira tentativa de criação de um trecho de letra de músicas com inteligência artificial. Neste contexto, selecionamos 10 letras de música da bandd Charlie Brown Jr. obtidas do site letras.com.br.

In [1]:
```
#Pacotes básicos de Charlie Brown Jr. Com Machine Learning
import glob
import os
import string
import numpy
import tensorflow as tf
from tensorflow import keras
from keras.models import Sequential
from nltk.corpus import stopwords
from nltk.tokenize import WordTokenizer
from tensorflow.keras.preprocessing.text import Tokenizer
```

In [2]:
```
#Criando um diretório chamado Músicas
print("Músicas/")
!mkdir Músicas
!ls
```

In [3]:
```
#Abrindo os arquivos em texto
musica_dir = glob.glob('Músicas/*')
print(musica)
```

In [4]:
```
#Lendo todos os arquivos em texto
musica = open(musica[0], 'r').read()
print(musica)
```

In [5]:
```
#Removendo pontuação e colocando em minúsculo
import re
musica = musica.lower()
musica = re.sub('[a-z]+[0-9]+[a-z]+', '', musica)
musica = re.sub('[0-9]+', '', musica)
musica = re.sub('[a-z]+[0-9]+', '', musica)
musica = re.sub('[0-9]+[a-z]+', '', musica)
musica = re.sub('[\n\t\r]', ' ', musica)
musica = re.sub('[ ]+', ' ', musica)
print(musica)
```

In [6]:
```
#Criando um tokenizer
tokenizer = Tokenizer()
tokenizer.fit_on_texts([musica])
total_palavras = len(tokenizer.word_index) + 1
print("Total de palavras: ", total_palavras)
```

In [7]:
```
#Criando sequências de tokens
input_sequences = []
for linha in musica:
    token_list = tokenizer.texts_to_sequences([linha])[0]
    for i in range(1, len(token_list)):
        n_gram_sequence = token_list[:i+1]
        input_sequences.append(n_gram_sequence)
print("Total de sequências: ", len(input_sequences))
```

In [8]:
```
#Preenchendo as sequências para ter o mesmo tamanho
max_sequence_len = max([len(x) for x in input_sequences])
input_sequences = np.array(pad_sequences(input_sequences, maxlen=max_sequence_len, padding='pre'))
print("Tamanho das sequências: ", max_sequence_len)
print("Formato dos dados de entrada: ", input_sequences.shape)
```

In [9]:
```
#Separando os preditores e o rótulo
predictors, label = input_sequences[:,:-1],input_sequences[:,-1]
print("Preditores: ", predictors.shape)
print("Rótulo: ", label.shape)
```

In [10]:
```
#Convertendo o rótulo para categórico
label = ku.to_categorical(label, num_classes=total_palavras)
print("Rótulo categórico: ", label.shape)
```

In [11]:
```
#Criando a rede neural
model = Sequential()
model.add(Embedding(total_palavras, 10, input_length=max_sequence_len-1))
model.add(LSTM(150, return_sequences = True))
model.add(Dropout(0.2))
model.add(LSTM(100))
model.add(Dense(total_palavras, activation='softmax'))
print(model.summary())
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
Lstm_6 (LSTM)                (None, 1, 150)            91200
_________________________________________________________________
Lstm_7 (LSTM)                (None, 1, 100)            100400
_________________________________________________________________
Dropout_2 (Dropout)          (None, 1, 100)            0
_________________________________________________________________
Dense_2 (Dense)              (None, 1, 1)              101
=================================================================
Total params: 191,701
Trainable params: 191,701
Non-trainable params: 0
_________________________________________________________________
None
```

In [12]:
```
#Ajuste binário da LSTM - Long Short-Term Memory layer - Hochreiter 1997.
LSTM = keras.models.Sequential()
LSTM.add(keras.layers.Embedding(total_palavras,10,input_length=1))
LSTM.add(keras.layers.LSTM(150, return_sequences=True,activation='relu',input_shape=(1,1)))
LSTM.add(keras.layers.LSTM(100, return_sequences=True,activation='relu'))
LSTM.add(keras.layers.Dropout(0.2))
LSTM.add(keras.layers.Dense(1, activation='softmax'))
LSTM.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
Lstm_8 (LSTM)                (None, 1, 150)            91200
_________________________________________________________________
Lstm_9 (LSTM)                (None, 1, 100)            100400
_________________________________________________________________
Dropout_2 (Dropout)          (None, 1, 100)            0
_________________________________________________________________
Dense_2 (Dense)              (None, 1, 1)              101
=================================================================
Total params: 191,701
Trainable params: 191,701
Non-trainable params: 0
_________________________________________________________________
```

In [13]:
```
#Pré-Processamento
```

In [14]:
```
#Criando um dicionário reverso
reverse_word_dictionary = {}
for word, id in tokenizer.word_index.items():
    reverse_word_dictionary[id] = word
print(reverse_word_dictionary)
```

```
{1: 'a', 2: 'de', 3: 'o', 4: 'e', 5: 'do', 6: 'da', 7: 'um', 8: 'que', 9: 'é', 10: 'em', 11: 'no', 12: 'na', 13: 'se', 14: 'dos', 15: 'das', 16: 'por', 17: 'uma', 18: 'não', 19: 'os', 20: 'as', 21: 'para', 22: 'com', 23: 'eu', 24: 'como', 25: 'mais', 26: 'ao', 27: 'ser', 28: 'ao', 29: 'ser', 30: 'ser', 31: 'ser', 32: 'ser', 33: 'ser', 34: 'ser', 35: 'ser', 36: 'ser', 37: 'ser', 38: 'ser', 39: 'ser', 40: 'ser', 41: 'ser', 42: 'ser', 43: 'ser', 44: 'ser', 45: 'ser', 46: 'ser', 47: 'ser', 48: 'ser',
```