Pop Lyric Generator

Text Generation is a type of Language Modelling problem.

Language Modelling is the core problem for a number of Natural Language Processing tasks such as speech to text, conversational system, and text summarization, etc.

Text Generation is a task which can be be architectured using deep learning models, particularly Recurrent Neural Networks.

In this project, we will experiment to generate new pop lyrics based on this Dataset (https://www.kaggle.com/neisse/scrapped-lyrics-from-6-genres).

Import Libraries

We have to import the libraries we're going to use, the most relevant are:

- **Pandas** in order to manipulate and analyze the data
- **Matplotlib** to visualize the data in graphics
- **Wordcloud** to help us when we are going to generate the next word of a sentence
- **Tensorflow & Keras** will help us in the process of machine learning and keras for deep learning.

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
import string, os
import tensorflow as tf

# keras module for building LSTM
from keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.layers import Embedding, Dropout, LSTM, Dense, Bi
directional
from keras.preprocessing.text import Tokenizer
from keras.callbacks import EarlyStopping
from keras.models import Sequential
```

Loading the Dataset

Once the Data was precleaned, it's going to be imported to the cloud. As it's mentioned in the documentation, it's on .csv because is easier to manipulate the data using Pandas library.

In [2]:
```
# Import csv file
df = pd.read_csv('../input/pop-lyrics/Pop.csv')
```

In [3]:
```
# Print first 10 rows to confirm is the right dataset
df.head()
```

Out[3]:

	Unnamed: 0	Artist	Songs	Popularity	ALink	Genre	Genres	SName	SLink
0	0	Adam Lambert	110	1.4	/adam-lambert/	Pop	Pop; Pop/Rock; Rock; RomÃ¢ntico; Dance; Electr...	Whataya Want From Me	/adam-lambert want-fr me.htm
1	1	Adam Lambert	110	1.4	/adam-lambert/	Pop	Pop; Pop/Rock; Rock; RomÃ¢ntico; Dance; Electr...	Ghost Town	/adam-lambert town.ht
2	2	Adam Lambert	110	1.4	/adam-lambert/	Pop	Pop; Pop/Rock; Rock; RomÃ¢ntico; Dance; Electr...	We Are The Champions (feat. Kris Allen & Queen)	/adam-lambert the-cha feat-kri
3	3	Adam Lambert	110	1.4	/adam-lambert/	Pop	Pop; Pop/Rock; Rock; RomÃ¢ntico; Dance; Electr...	If I Had You	/adam-lambert you.htm
4	4	Adam Lambert	110	1.4	/adam-lambert/	Pop	Pop; Pop/Rock; Rock; RomÃ¢ntico; Dance; Electr...	Never Close Our Eyes	/adam-lambert close-o eyes.ht

Data Cleaning

In this step the dataset is going to be cleaned using Pandas library to remove the columns that are not going to be used and keep only the unique column called "Lyrics".

In [4]:
```
# Drop all the columns except Lyrics
df.drop(['Artist','Songs','Popularity','Genre','Genres','Idiom','ALink'
,'SName','SLink'],axis=1,inplace=True)
df.drop(df.columns[df.columns.str.contains('unnamed',case = False)],axi
s = 1, inplace = True)
```

In [5]:
```
# Print first 10 rows to confirm the drops were made correctly
df.head()
```

Out[5]:

	Lyric
0	Hey, slow it down. What do you want from me. W...
1	Died last night in my dreams. Walking the stre...
2	I've paid my dues. Time after time. I've done ...
3	So I got my boots on,. Got the right amount of...
4	I wish that this night would never be over. Th...

Shaping the Dataset

Since kaggle does not have enough memory (16gb RAM), disk usage (73 GB) and GPU (13GB) to process the entire dataset, we will have to limit our model from 28441 to 700 lyrics. Kaggle give It's worth mentioning that this same test was performed locally and took approximately 100 hours to perform.

In [6]:
```
# Dataframe shape to show how many columns and rows the dataframe have
df.shape
```

Out[6]:
```
(28441, 1)
```

In [7]:
```
# Take first 700 rows
df = df[:700]
```

In [8]:
```
df.head()
```

Out[8]:

	Lyric
0	Hey, slow it down. What do you want from me. W...
1	Died last night in my dreams. Walking the stre...
2	I've paid my dues. Time after time. I've done ...
3	So I got my boots on,. Got the right amount of...
4	I wish that this night would never be over. Th...

In [9]:
```
# Dataframe shape to show how many columns and rows the dataframe have a
fter last cell
df.shape
```

Out[9]:
```
(700, 1)
```

Now, to get Statistical information, the next cell is going to calculate the number of words per row, this is going to help us to determine the Frequency Distribution of number of words for each text extracted

In [10]:
```
df['Number_of_words'] = df['Lyric'].apply(lambda x:len(str(x).split()))
df.head()
```

Out[10]:

	Lyric	Number_of_words
0	Hey, slow it down. What do you want from me. W...	314
1	Died last night in my dreams. Walking the stre...	224
2	I've paid my dues. Time after time. I've done ...	153
3	So I got my boots on,. Got the right amount of...	365
4	I wish that this night would never be over. Th...	295

Statistical information

In [11]:
```
df['Number_of_words'].describe()
```

Out[11]:
```
count    700.000000
mean     248.065714
std       97.323252
min       31.000000
25%      178.000000
50%      233.000000
75%      309.000000
max      589.000000
Name: Number_of_words, dtype: float64
```

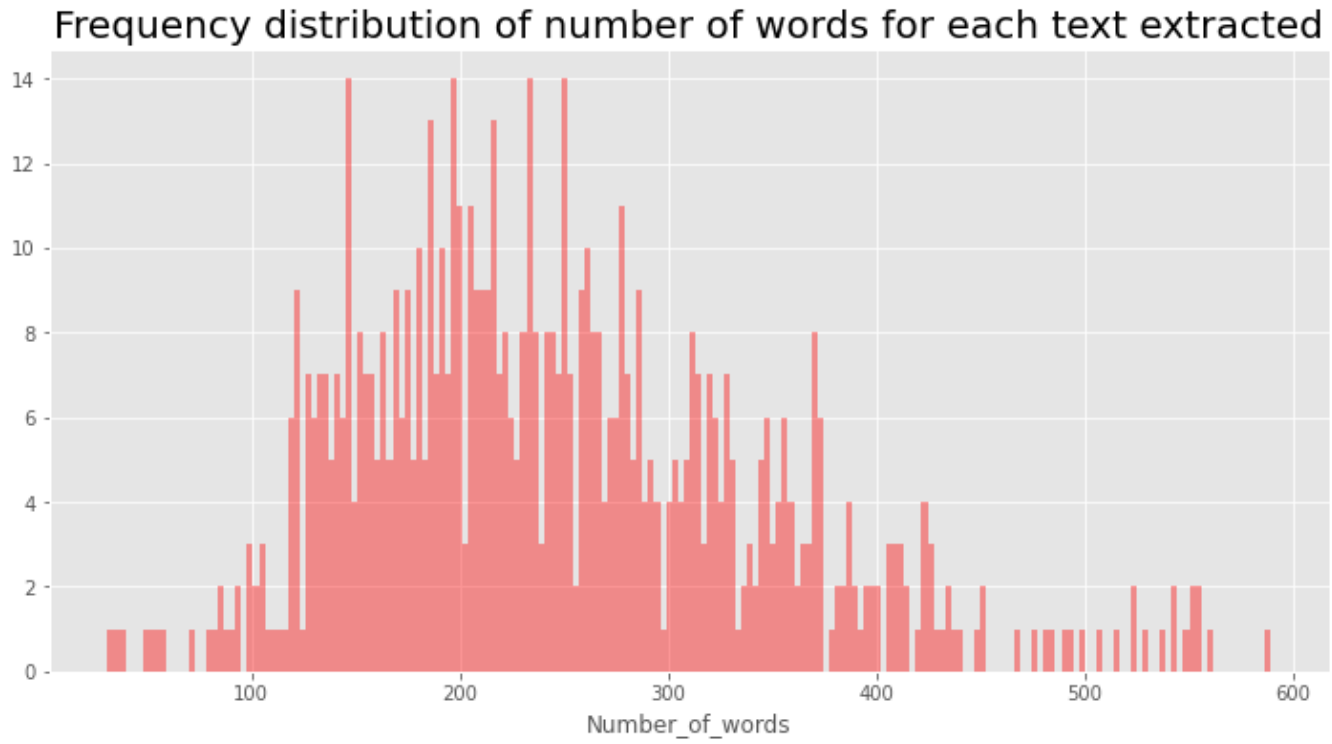
- **count**: Number of rows to evaluate
- **mean**: Average of words per row in the dataset
- **std**: Word's standard deviation
- **min**: Minimum amount of words found in a lyric or row
- **max**: Maximum amount of words found.

In [12]:
```
import matplotlib.pyplot as plt
plt.style.use('ggplot')
plt.figure(figsize=(12,6))
sns.distplot(df['Number_of_words'],kde = False,color="red",bins=200)
plt.title("Frequency distribution of number of words for each text extr
acted", size=20)
```

```
/opt/conda/lib/python3.7/site-packages/seaborn/distributions.py:2557: F
utureWarning: `distplot` is a deprecated function and will be removed i
n a future version. Please adapt your code to use either `displot` (a f
igure-level function with similar flexibility) or `histplot` (an axes-l
evel function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[12]:

```
Text(0.5, 1.0, 'Frequency distribution of number of words for each text
extracted')
```



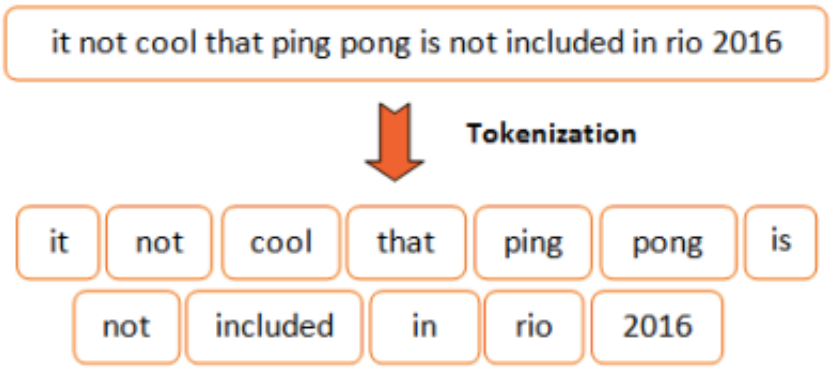
- **Y axis**: Represent how many times a word was found in the dataset
- **X axis**: Represent the amount of different words found;

Tokenization

The data from the column named **Lyric** must be preprocessed and to achieve these, the words will have to be converted into numbers. This process is called "tokenization".

Keras uses the class ***Tokenizer()*** to do this job, this class has two important methods to be used:

- **_fit_on*text()*** : Update the internal vocabulary based on a given list of texts or in this case, "Lyrics" column, where each entry of the list is going to be a token.
- **_texts_to*sequences()*** : Transform each text within the list of texts supplied to a sequence of integers; only words known by the tokenizer will be considered.



In [13]:
```
tokenizer = Tokenizer()
tokenizer.fit_on_texts(df['Lyric'].astype(str).str.lower())

total_words = len(tokenizer.word_index)+1
tokenized_sentences = tokenizer.texts_to_sequences(df['Lyric'].astype(s
tr))
tokenized_sentences[0]
```

Out[13]:
```
[128,
 338,
 9,
 48,
```

```
31,
33,
1,
45,
72,
6,
31,
33,
1,
45,
72,
6,
35,
13,
459,
31,
33,
1,
45,
72,
6,
31,
33,
1,
45,
72,
6,
75,
387,
50,
95,
7,
43,
2,
108,
88,
232,
```

```
70,
202,
368,
653,
7,
43,
2,
290,
88,
7,
1023,
23,
40,
80,
25,
52,
18,
31,
33,
1,
45,
72,
6,
31,
33,
1,
45,
72,
6,
28,
16,
88,
32,
13,
1073,
9,
49,
```

```
139,
16,
88,
10,
2,
92,
66,
1,
48,
9,
1878,
6,
32,
69,
7,
784,
4,
372,
28,
125,
204,
112,
128,
31,
33,
1,
45,
72,
6,
31,
33,
1,
45,
72,
6,
35,
26,
```

```
1074,
4,
61,
11,
47,
39,
379,
5,
26,
124,
193,
42,
1,
26,
6,
2456,
13,
7,
3010,
23,
2116,
22,
832,
6,
109,
39,
527,
9,
2117,
75,
387,
50,
95,
7,
43,
30,
2,
```

```
108,
66,
1,
506,
70,
2,
593,
198,
155,
23,
2,
82,
1,
85,
340,
8,
81,
28,
16,
88,
32,
13,
1073,
9,
49,
139,
16,
88,
10,
2,
92,
66,
1,
48,
9,
1878,
6,
```

```
32,
69,
7,
784,
4,
372,
28,
125,
204,
112,
128,
31,
33,
1,
45,
72,
6,
31,
33,
1,
45,
72,
6,
28,
16,
88,
32,
14,
6,
2,
92,
66,
1,
48,
34,
2,
92,
```

```
66,
1,
48,
18,
28,
16,
88,
32,
13,
1073,
9,
49,
139,
16,
88,
10,
2,
92,
66,
1,
48,
9,
1878,
6,
32,
69,
7,
784,
4,
372,
28,
125,
204,
112,
128,
31,
33,
```

```
1,
45,
72,
6,
28,
16,
88,
32,
13,
1073,
9,
49,
139,
16,
88,
10,
2,
92,
66,
1,
48,
9,
1878,
6,
32,
69,
7,
784,
4,
372,
28,
125,
204,
112,
128,
2118,
45,
```

```
72,
6,
2118,
45,
72,
6,
2118,
45,
72,
6,
2118,
45,
72,
6]
```

Slash sequences into a n-gram sequence

After the text is tokenized, the words will be sorted to represent them numerically by creating an input sequence using the created tokens.

In [14]:

```
input_sequences = list()
for i in tokenized_sentences:
    for t in range(1, len(i)):
        n_gram_sequence = i[:t+1]
        input_sequences.append(n_gram_sequence)
```

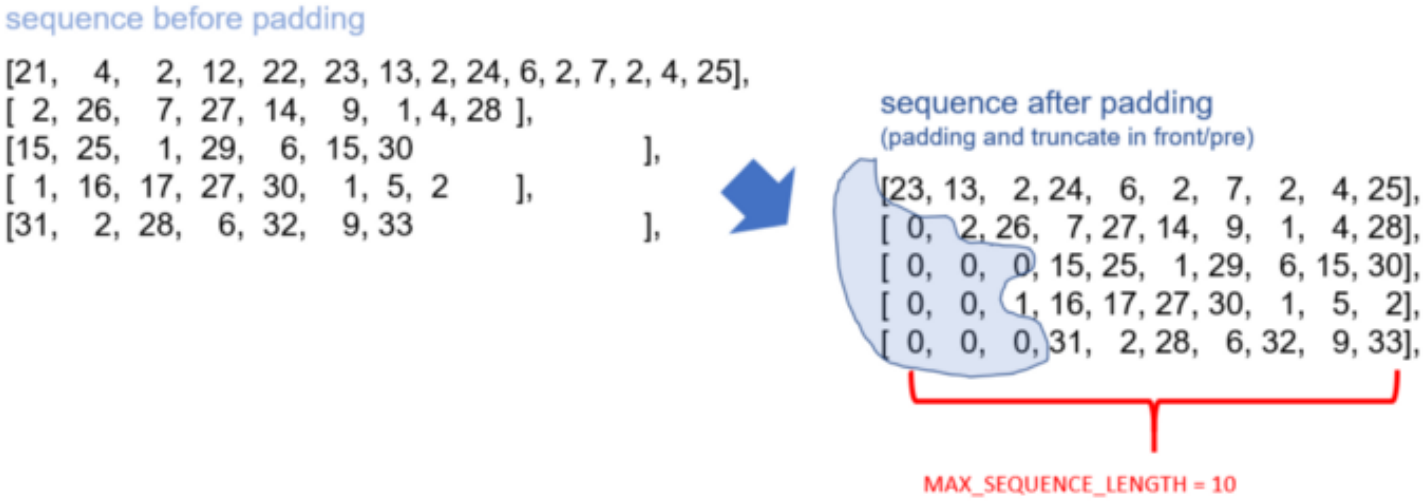
Padding

Before the model generation, is necessary to normalize all sentences to the same standard lenght, to avoid the memory overflow and to get the layers of the model way more deep, this is a simple process to add a 0's in the beginning of the text, resulting in layers of the same size.

_pad*sequences* Transform a list of sequences that is a lists of integers into a 2D array, in this case the list is called input_sequences.

Sequences resulting shorter than **maxlen** are padded with 0's until they have the same length.

The position where the zeros will be added is determined by the argument **padding**, in this case, it will be done at the beginning of the sequence.



In [15]:

```
max_sequence_len = max([len(x) for x in input_sequences])
input_sequences = np.array(pad_sequences(input_sequences, maxlen=max_se
quence_len, padding='pre'))
```

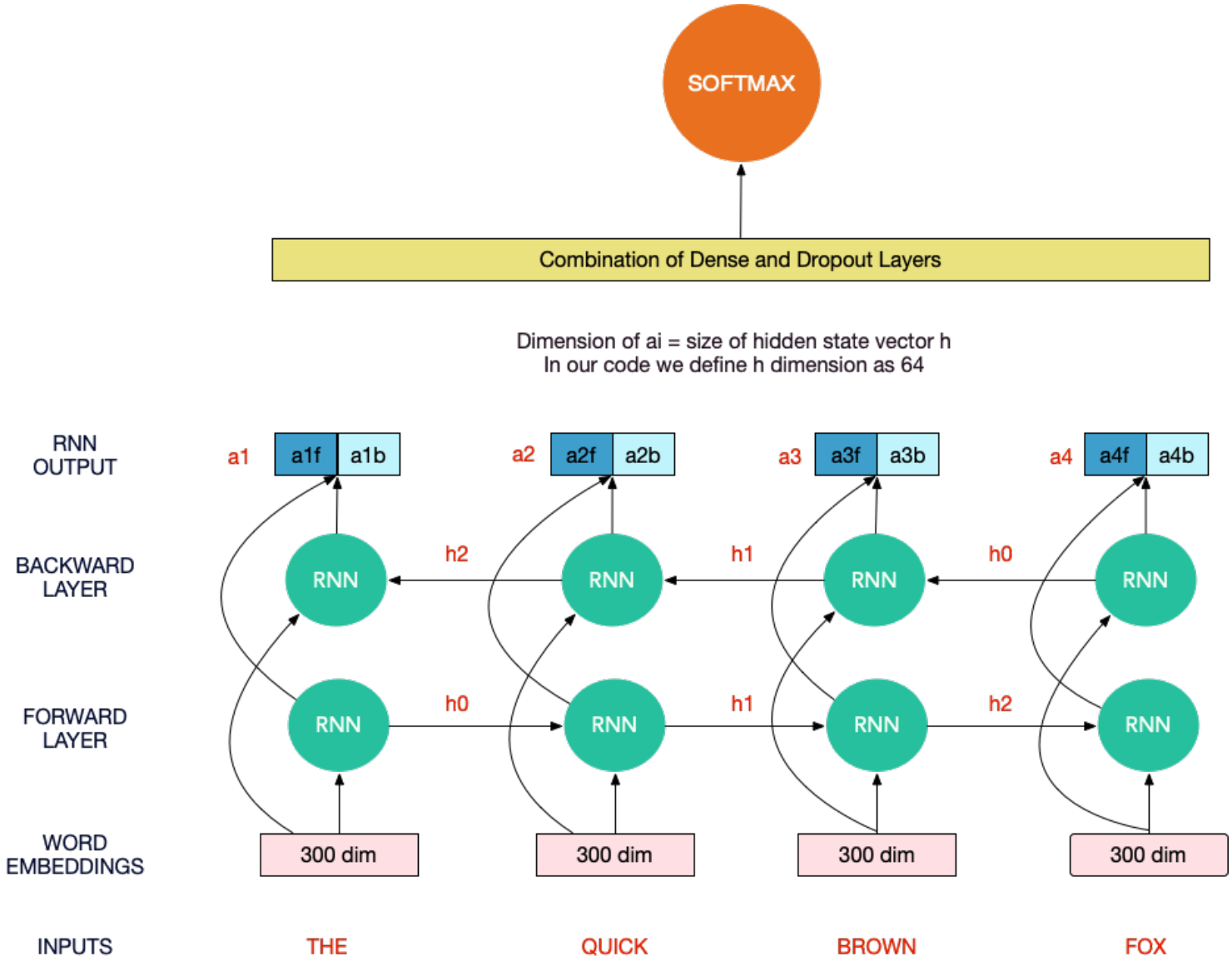
In [16]:
```
input_sequences[:20]
```

Out[16]:
```
array([[  0,   0,   0, ...,   0, 128, 338],
       [  0,   0,   0, ..., 128, 338,   9],
       [  0,   0,   0, ..., 338,   9,  48],
       ...,
       [  0,   0,   0, ...,  35,  13, 459],
       [  0,   0,   0, ...,  13, 459,  31],
       [  0,   0,   0, ..., 459,  31,  33]], dtype=int32)
```

In [17]:
```
# create predictors and label
X, labels = input_sequences[:,:-1],input_sequences[:,-1]
y = tf.keras.utils.to_categorical(labels, num_classes=total_words)
```

Creating the Model

This test will use the Bidirectional LSTM model, this kind of neural networks run as the name says: in two ways. This is from past to future and vice versa, this is how the model preserves the information of both states at any point. LSTM neural networks are mostly used where context is involved.



In [ ]:
```
model = Sequential()
```

Models in Keras are defined as a sequence of layers, and the Sequential model is about adding layers one at a time. Layers are the basic building block of neural network.

Layers:

Embedding

- Is a core layer, can only be used as the first layer in a model, turns positive integers into dense vectors of fixed size (first parameter is the size of the vocabulary, second parameter is the dimension of the dense embedding, and third parameter is about the length of sequences which is required because we will use a Dense layer later)

In [ ]:
```
model.add(Embedding(total_words, 40, input_length=max_sequence_len-1))
```

Bidirectional:

- Is a recurrent layer, a bidirectional wrapper for RNN's which will recibe a layer as an input being LSTM layer the one we chose, it will receive a positive integer as it's input which refers to the amount of output nodes that should be returned.

In [ ]:
```
model.add(Bidirectional(LSTM(250)))
```

Dropout:

- Is a regularization layer. This layer randomly sets input units to 0, with a frequency of the value we pass it,at each step during training time which helps prevent overfitting.

In [ ]:
```
model.add(Dropout(0.1))
```

Dense:

- Is a core layer, and a densely-connected neural network layer. Receives as first parameter a positive integer which refers to the amount of output nodes that should be returned. Second parameter is the one named **activation** which defines the type of predictions the model can make; for the kind of problem we are abording the one which suits the better is softmax which outputs a vector of values (input) that can be interpretated as probabilities of being used.

In [ ]:
```
model.add(Dense(total_words, activation='softmax'))
```

Compile method.

Takes two relevant parameters:

- **loss**: Also known as a cost function; works during the optimization process and it's role is to calculate the error of the model. Cross-entropy is used to estimate the difference between an esimated and predicted probability distributions. categorical_cross-entropy will be used because it's best suited for this kind of problems and is almost universally used to train deep learning neural networks due to the results it produces.
- **optimizer**: Is responsible for reducing the losses and to provide the most accurate results possible. Adam is the option chosen because is the best choice offered by Keras to train the neural network in less time and more efficiently. Earlystop will stop the training if the model has stopped improving, this will be checked at the end of every epoch.

Optimization algorithm and Performance metrics

The optimization algorithm added in the configuration layer will be **Adam**, because of the good perfomance and results in other projects it has.

In this case, the 'accuracy' will be measured as perfomance metric, which gives closeness of calculated value to the actual value.

In [ ]:
```
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

fit method trains the model for the fixed number of epochs given (iterations on the dataset).

In [ ]:
```
earlystop = EarlyStopping(monitor='loss', min_delta=0, patience=3, verbose=0, mode='auto')
history = model.fit(X, y, epochs=20, verbose=1, callbacks=[earlystop])
```

Evaluating the Model

In this block a new graphic will be generated and displayed, the Y axis will stand for accuracy and X axis will stand for the amount of epochs; as shown, in order to increase the accuracy, you have to increase the number of epochs during the training.

In [ ]:
```
# plot the accuracy

plt.plot(history.history['accuracy'], label='train acc')
plt.legend()
plt.show()
plt.savefig('AccVal_acc')
```

Import the Trained Model

Once the training process is completed it only remains the import of our model so we can test how does it work, in our case the trained model is called 'song_lyrics_generator'.

In [18]:
```
import matplotlib.pyplot as plt
import seaborn as sns
from keras.models import load_model
model = load_model('../input/songlyricmodel/song_lyrics_generator.h5')
```

Function to Generate the song

This step is in charge of preparing the function that will be used to complete a song given the model previously trained, it will predict the next words based on the input words suministrated as 'seed_text'. For this to work a tokenization must be applied to the seed_text, then a padding will be applied to the sequences generated and passed into the trained model so the next word can be predicted.

In [19]:

```
def complete_this_song(seed_text, next_words):
    for _ in range(next_words):
        token_list = tokenizer.texts_to_sequences([seed_text])[0]
        token_list = pad_sequences([token_list], maxlen=max_sequence_le
n-1, padding='pre')
        predicted = model.predict_classes(token_list, verbose=0)
        output_word = ""
        for word, index in tokenizer.word_index.items():
            if index == predicted:
                output_word = word
                break
        seed_text += " " + output_word
    return seed_text
```

Examples

In [20]:

```
complete_this_song("I am missing you", 200)
```

```
/opt/conda/lib/python3.7/site-packages/tensorflow/python/keras/engine/sequential.py:450: UserWarning: `model.predict_classes()` is deprecated and will be removed after 2021-01-01. Please use instead:* `np.argmax(model.predict(x), axis=-1)`,   if your model does multi-class classification   (e.g. if it uses a `softmax` last-layer activation).* `(model.predict(x) > 0.5).astype("int32")`,   if your model does binary classification   (e.g. if it uses a `sigmoid` last-layer activation).
  warnings.warn('`model.predict_classes()` is deprecated and '
```

Out[20]:

```
"I am missing you i believe in the things you were so far away from me i could not see the words that i was so cold and i never need to know your love is good i don't know how to tell you i'm not a fortune teller i never change but i want you to stay i don't know what to tell you i'm not a fortune teller i never change but i want you to stay i don't know what to tell you i'm not a fortune teller i never change but i want you to stay i don't know what to tell you i'm not a fortune teller i never change but i want you to stay i don't know what to tell you i'm not a fortune teller i never change but i want you to stay i don't know what to tell you i'm not a fortune teller i never change but i want you to stay i don't know what to tell you i'm not a fortune teller i never change but i want you to"
```

In [21]:

```
complete_this_song("It's a cruel and random world", 80)
```

Out[21]:

```
"It's a cruel and random world is not much between how how could you do and i see your life is a ghost and my heart is a ghost town my heart is a ghost town my heart is a ghost town my heart is a ghost town my heart is a ghost town my heart is a ghost town my heart is a ghost town my heart is a ghost town my heart is a ghost town my heart is a ghost town my heart is a ghost town my heart is a ghost town my heart"
```

In [22]:
```
complete_this_song("I want a piece of pizza", 80)
```

Out[22]:
```
"I want a piece of pizza your eyes the bad are burning hot the sea you know you feel it up you are calling to you you look at me tonight you t hink of me of us how how how do how do we do how do we do how do we do how do we do how do we do how do we do how do we do how or now and now you couldn't see the ocean of hate your life you reply if"
```

In [23]:
```
complete_this_song("Love flowers", 80)
```

Out[23]:
```
"Love flowers get to her love and you make it okay but i'm not here i s till still make you wanna be the one for you ooh oh oh oh oh baby baby baby baby baby baby baby baby baby you're gonna get you home and you go standing in your way and i don't want to miss a thing because i'm happy clap along if you know what happiness is is enough for me and you can d o i know that"
```

In [24]:
```
complete_this_song("Hate flowers", 80)
```

Out[24]:
```
"Hate flowers love me love me love me how do you do you know what you d o you love me if you love me if you love me love you don't want me to m e and maybe i want you to trust me again i want to be your fantasy mayb e i'm gonna be the one who understands you through me i know you unders tand me i want you to take me back to find to take to take you the swee test"
```