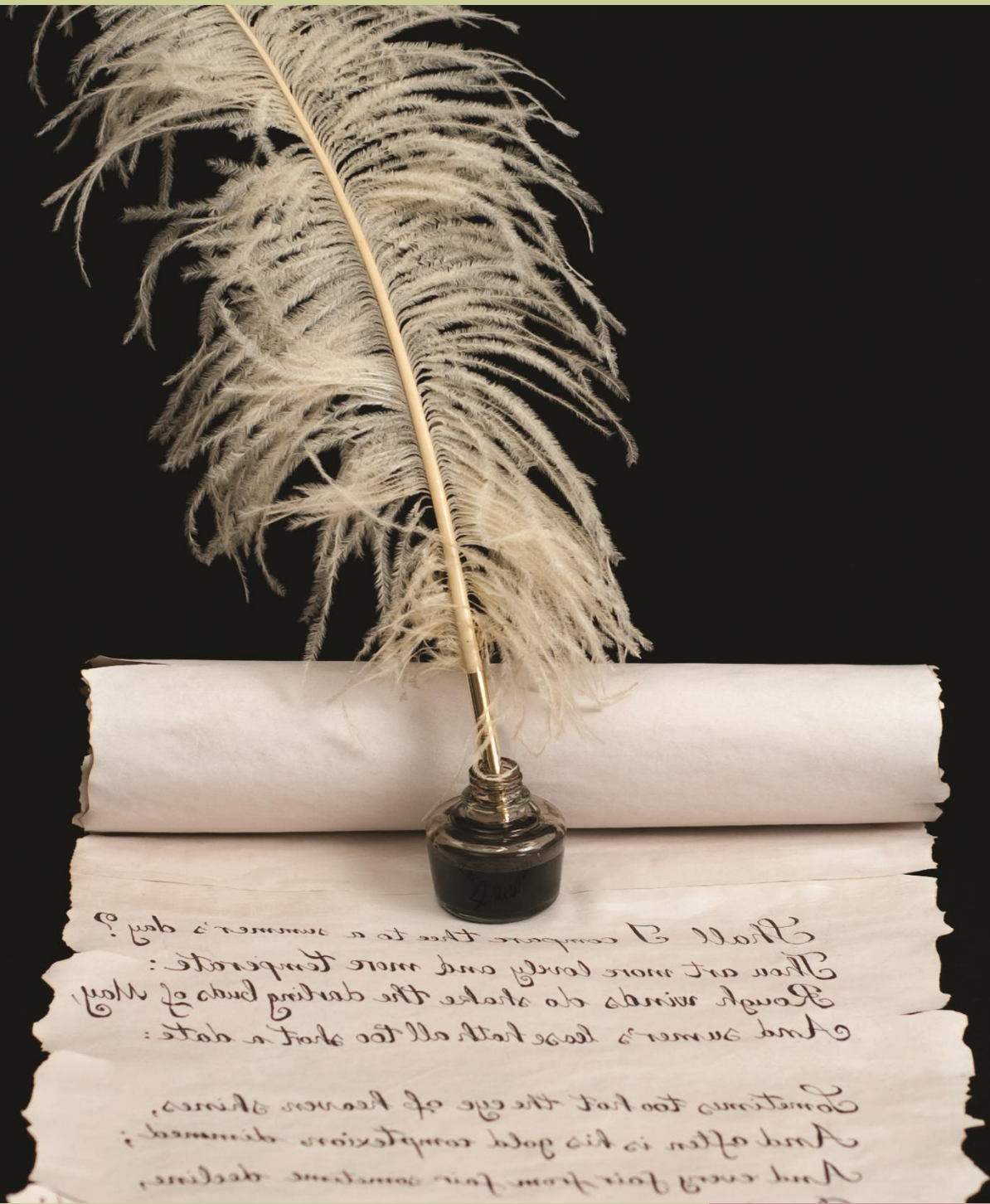


# Poem Generation



## Neural Network for Poem Generation

**Students:**  
Salem Allosh  
Aws Abdul Hamed

# Introduction

Main aim of this project is to create a model to generate a poem depending on specific poet pattern.

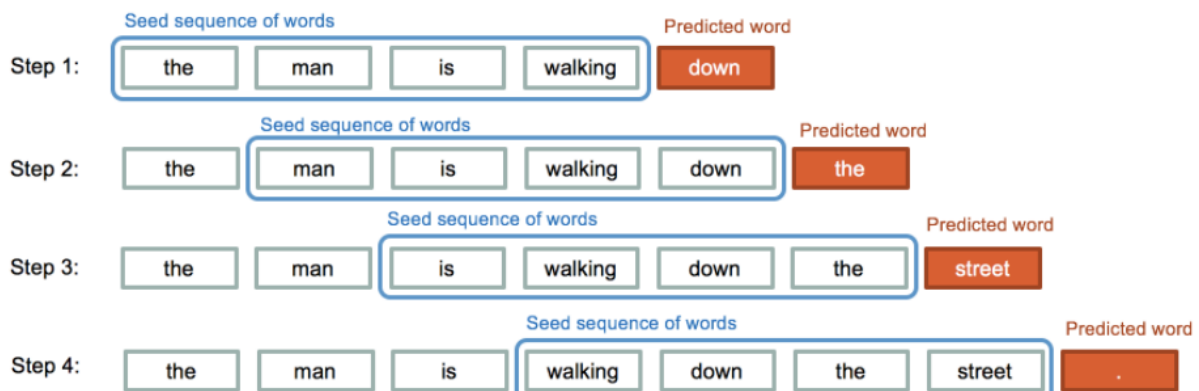
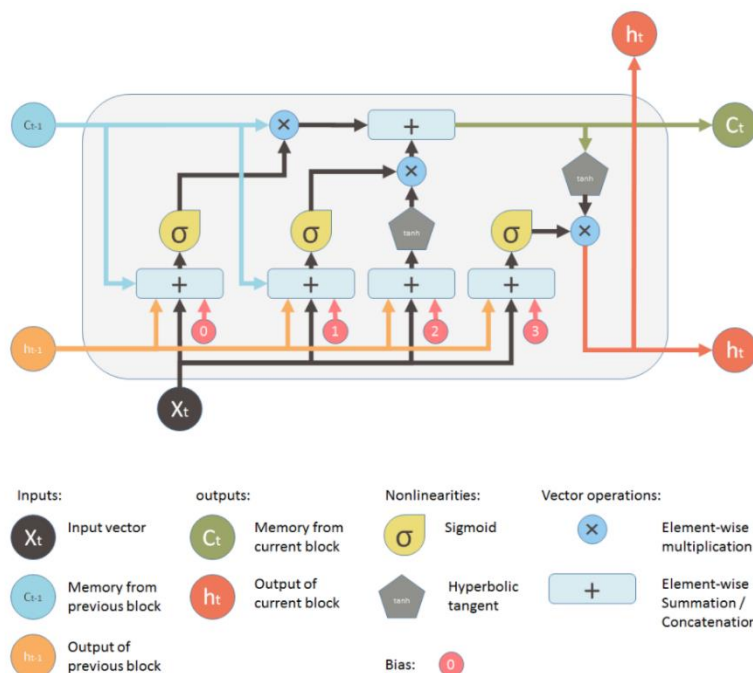
To develop this project, we are going through several steps:

- 1- Choosing the algorithms that achieve the goal.
- 2- Get the Dataset.
- 3- Preprocessing Dataset
- 4- Create network model
- 5- Evaluate the result

# Algorithms

Since the sentences in text can be representing in embedding and how the semantics in text over long stretch might be learned using LSTM ..

In text generation, we try to predict the next character or word of the sequence. The text data generally considered as sequence of data. For predicting data in sequence we used deep learning models like RNN or LSTM. LSTM are preferred over RNN in this because of RNN vanishing and exploding gradients problem. Since in text generation we have to memorize large amount of previous data. So for this purpose LSTM are preferred.



# Data set

We depended on Arabic\_potery data set ,we found it on Kaggle

Data set properties:

- Consist of 3 columns :
  - Category :
    - This dataset contain ( 28 ) different types of poem ages and types such as :

```
[ ] categories = GetPoemCategories(df)
print(categories)
```

```
['الإمارات', 'البحرين', 'الجزائر', 'السعودية', 'السودان', 'العراق', 'المغرب',
'اليمن', 'تونس', 'سوريا', 'عمان', 'فلسطين', 'لبنان', 'ليبيا', 'مصر', 'الأردن',
'الكويت', 'قطر', 'موريتانيا', 'العصر الجاهلي', 'العصر الإسلامي',
'العصر العباسي', 'العصر الأيوبي', 'العصر العثماني', 'عصر الممضرمون',
'العصر الأموي', 'العصر الأندلسي', 'العصر المملوكي']
```

- Poet\_name:

- And contains of 538 poet name

```
poets = GetPoetNames(df)
print(poets)
```

[ 'خلفان بن مصبح' 'سالم بن عبدالله الكراني' 'حماد بن سعيد'  
 'مبارك بن حمد المقيلي' 'علي بن محمد الشحي' 'محمد بن حمود الشحي'  
 'محمد بن صالح المنتفقي' 'يوسف بن رويسم' 'هدى السعدي' 'مانع سعيد العتيبة'  
 'عارف الخاجة' 'حمد بن خليفة أبو شهاب' 'كريم معتوق'  
 'سالم أبو جمهور القبيسي' 'قاسم حداد' 'أحلام مستغانمي' 'سهام آل براهيم'  
 'جلواح' 'عبد القادر الجزائري' 'سليمان بن سحمان' 'عبد الرحمن بن مساعد'  
 'أحمد اللهيب' 'عبد الرحمن العشماوي' 'شريف بقنه' 'ابن بشير الإحساني'  
 'سامي المالكي' 'غازي القصيبي' 'محمد بن عبود العمودي'  
 'مدثر بن إبراهيم بن الحجاز' 'محمد عبد الباري' 'إدريس جمّاع'  
 'حمزة الملك طمبل' 'أحمد مطر' 'نازك الملائكة' 'بدر شاكر السياب'  
 'محمد مهدي الجواهري' 'عبد الرزاق عبد الواحد' 'بهاء الدين الصيادي'  
 'معروف الرصافي' 'أبو الفيض الكتاني' 'محمد اسموني' 'شاعر الحمراء'  
 'عبدالله البردوني' 'عبد الولي الشميري' 'ابن شهاب العلوي' 'محمد الشوكاني'  
 'ابن طاهر' 'ابن رشيق القيرواني' 'الشاذلي خزنة دار' 'أبو القاسم الشابي'  
 'سليم عبدالقادر' 'نزار قباني' 'خليل مردم بك' 'بطرس كرامة'  
 'أبو الهدى الصيادي' 'ابن شيخان السالمي' 'سيف الرحبي' 'صالح الفهدي'  
 'هلال بن سعيد العماني' 'توفيق زياد' 'جورج جريس فرح' 'محمود درويش'  
 'بديع القشاعلة' 'تميم البرغوثي' 'أسامة محمد زامل' 'إبراهيم طوقان'

– Poem\_text:

- Consists of 54767 poem text

```
[ ] SyrianPoem = SplitByCategory(df, "سوريا")
SyrianPoem.head()
```

	category	poet_name	poem_text
4061	سوريا	سليم عبدالقادر	...كلّ وجهٍ منهم\إنهم يطْلعون من كلّ أَقْبى
4062	سوريا	سليم عبدالقادر	...في\إنّ لِي: سِرٌّ، وَرَاحَ نَعْتُو أَمَامِي
4063	سوريا	سليم عبدالقادر	... والموت يرقص لي\ماضٍ ، وأعرف ما دربي وما هدفي
4064	سوريا	سليم عبدالقادر	...فلي\إنّما تَخَافُونَنِي، لَسْتُ أَدْرِي
4065	سوريا	سليم عبدالقادر	...هَذَا الحِصَارُ الزَّهِيْبُ الْمُتَعَبُ القَاسِي

# Preprocessing

since the data set is large and there is not a suitable hardware to train such model, so we will split our model to parts and train a model for generate a poem depending on poet name, to achieve that we developed many functions to help us in preprocessing:

- Load data set from google drive

```
from pydrive.auth import GoogleAuth
from pydrive.drive import GoogleDrive
from google.colab import auth
from oauth2client.client import GoogleCredentials

[3] auth.authenticate_user()
    gauth = GoogleAuth()
    gauth.credentials = GoogleCredentials.get_application_default()
    drive = GoogleDrive(gauth)

[5] fileDownloaded = drive.CreateFile({'id':'1_vfqCzvNaR_ApgaVwCCqbWDa21s95W5T'})

[6] fileDownloaded.GetContentFile('Arabic_poetry_dataset.csv')
```

- Load Dataset To Data Frame

- With ensure that the data set has no empty cell

```
import numpy as np
import pandas as pd

missing_values = ["missing",np.nan]
df = pd.read_csv('Arabic_poetry_dataset.csv',na_values= missing_values,usecols = ['poet_name','poem_text','category'])
```

- Head Example

df.head()

	category	poet_name	poem_text
0	الإمارات	خلفان بن مصبح	...وجدت بالزيارة والنو...بيت تختال في خلل الجمال
1	الإمارات	خلفان بن مصبح	...أسع...يا وحي إلهام الصدور...يا طائر الشعر القرير
2	الإمارات	خلفان بن مصبح	...د...أن أرى فيك جمال العرب...بيت حجات أرى من عجب
3	الإمارات	خلفان بن مصبح	...وقد كما الأرض بال...هذا الربيع بلور الحسن واقانا
4	الإمارات	خلفان بن مصبح	...أخفك ما بي أم اطن...اروحي فذاك وإن مُنحتُ صدوداً

- **GetPoetNames(DataFrame)**

- Take One arguments DataFrame And return all poets names in this dataframe

```
def GetPoetNames(DataFrame):
    poet_name = DataFrame.poet_name.unique()
    return poet_name
```

- **GetPoemCategory(DataFrame)**

- Take One arguments DataFrame And return all Categories names in this data frame

```
def GetPoemCategories(Dataframe):
    poem_categories = Dataframe.category.unique()
    return poem_categories
```

- **GetPoetsNamesByCategories(DataFrame,Category)**

- Take Two arguments Dataframe and Category and return all poets that belong to Category

```
def GetPoetsNamesByCategories(Dataframe,Category):
    PoemCategory = Dataframe.loc[Dataframe.category==Category]
    poetName = PoemCategory.poet_name.unique()
    return poetName
```

- Example :

```
GetPoetsNamesByCategories(df,"سوريا")

array(['سليم عبدالقادر', 'نزار قبّاني', 'خليل مردم بك', 'يونس كرامة',
      'أبو الهدى الصيادي'], dtype=object)
```

- **GetCategoryByPoetName(DataFrame,PoetName)**

- Take Two arguments Dataframe and Category and return all Category that contain poems for that Poet

```
def GetCategoryByPoetName(Dataframe,PoetName):
    PoetName = Dataframe.loc[Dataframe.poet_name==PoetName]
    CategoryName = PoetName.category.unique()
    return CategoryName
```

- Example :

```
GetCategoryByPoetName(df,"إيليا أبو ماضي")

array(['لبنان'], dtype=object)
```

- **SplitByCategory(DataFrame,Category)**

- Create a new Dataset that only contains poems and poets of that belong to Category



```
def SplitByCategory(Dataframe,CategoryName):
    SubDataframe = Dataframe.loc[Dataframe["category"]==CategoryName]
    SubDataframe.to_csv(CategoryName+'.csv',encoding='utf-8-sig')
    return SubDataframe
```

– Example :

```
SyrianPoem = SplitByCategory(df,"سوريا")
SyrianPoem.head()
```

	category	poet_name	poem_text
4061	سوريا	سليم عبدالقادر	...كُلُّ وَجْهِ مَتَّهِمٍ \n إِلَيْهِمْ يَطْلَعُونَ مِنْ كُلِّ أَفْقٍ
4062	سوريا	سليم عبدالقادر	...فِي \n الْإِلَهِ لِي: سِرٌّ، وَرَاحَ يُغْدُو أَقْلَوِي
4063	سوريا	سليم عبدالقادر	... وَالْمَوْتُ يَرْقُصُ لِي \n ماضٍ ، وَأَعْرِفُ مَا دَرَيْي وَمَا هَدَفِي
4064	سوريا	سليم عبدالقادر	...فَلْيَ \n لِمَاذَا تَخَافُونَنِي، أَسَيْتُ أَثْرِي
4065	سوريا	سليم عبدالقادر	... هَذَا الْجَمْرُ الْوُجَيْبُ الْمُتَجِدِّبُ الْقَاسِي

- SplitByPoetName(DataFrame,PoetName)

– Create a new Dataset that only contains data that belong to PoetName.

```
def SplitByPoetName(Dataframe , PoetName ):
    SubDataframe = Dataframe.loc[Dataframe["poet_name"]==PoetName]
    SubDataframe.to_csv(PoetName+'.csv',encoding='utf-8-sig')
    return SubDataframe
```

– Example :

```
AboAlqasemDataframe = SplitByPoetName(df,"أبو القاسم الشابي")
AboAlqasemDataframe.head()
```

	category	poet_name	poem_text
3970	تونس	أبو القاسم الشابي	...كَأَنَّ آتِخْرُجْنَ مِنْ فُرْجَاتِ النُّعْمِ دَامِيَةً
3971	تونس	أبو القاسم الشابي	...يَدْمُوعَ عَيْنِكَ\مَا كُلُّمَا بَلَ رُبُّمَا عَيْتَ الْبُكََا
3972	تونس	أبو القاسم الشابي	... كَالْمُسْرِ\مُنَاعِيضُ رَغَمِ الدَّاءِ وَالْأَعْدَاءِ
3973	تونس	أبو القاسم الشابي	...وَهُمُومِي\لَيْهَا الْخُبُّ لَنْتَ سِرُّ بِلَاثِي
3974	تونس	أبو القاسم الشابي	...نَفْسٍ مِنْ\إِذَا لَنْتَ شِعْرِي هَلْ لِلْئَلِ الن

```
EleaAboMadiDataFrame = SplitByPoetName(df,"إيليا ابو ماضي")
EleaAboMadiDataFrame.nunique()
```

```
category      1
poet_name     1
poem_text    172
dtype: int64
```

- **GetPoems(Dataframe)**

- Take One Argument and return a string of all poems in that data frame

```
def GetPoems(Dataframe):
    PoemDF = Dataframe.poem_text
    Poems = ""
    for i in range(len(PoemDF)):
        Poems += PoemDF.iloc[i]

    return Poems
```

```
AboAlqasemPoems = GetPoems(AboAlqasemDataframe)

print(AboAlqasemPoems)
```

بِرَبِّهِمْ يَسْتَسْقِمْ  
 فَقَدْ قُتِّ فِي زُلُو الْبَيِّنَةِ مَحْضَرُ  
 أَتَلَوْا عَلَى الْإِسْلَامِ مَنْ قَدْ يُهَاجِمُ  
 فَوَالْحَقِّ مَا هَذِي الزُّوَايَا وَأَهْلُهَا  
 مَيَّوَى مَصْنَعٍ فِيهِ تُصَاغُ السَّخَائِلُ  
 لَحَى اللَّهِ مَنْ لَمْ تُسْتَبْرِهْ حَمِيَّةُ  
 عَلَى بَيْنِهِ إِنَّ دَاهِمَتُهُ الْغَطَايُ تُرْجَوُ السَّعَادَةُ يَا قَلْبِي وَلَوْ وُجِدَتْ  
 فِي الْكَوْنِ لَمْ يَسْتَعِزْ خُزْنٌ وَلَا أَلَمٌ  
 وَلَا اسْتَحَالَتْ حِيلَةُ النَّاسِ أَجْمَعِهَا  
 وَزُلْزِلَتْ هَلَاكُهُ الْأَكْوَانُ وَالنُّظُمُ  
 فَمَا السَّعَادَةُ فِي الدُّنْيَا مَيَّوَى مُنَى  
 نَاءٍ تُضَيِّقُ لَهُ أَيْمَانُهَا الْأُمَمُ  
 نَاجَتْ بِهِ النَّاسُ أَوْهَامٌ مُخْرِجَةٌ  
 لَمَّا تَخَسَّنَتْهُمُ الْأَحْلَامُ وَالظُّلُمُ  
 فَهَبْ كُلَّ يَدَايِهِ وَيَسُدَّهُ  
 كَلَّمَ النَّاسُ مَا تَأَمَّلُوا وَلَا حَطَمُوا

- **TokenizePoems(Poem,Print=False)**

- Take two arguments String of all poems And a print Boolean Value to print details if want .

- **Return :**

**Corpus :** Set of each line in poems

**Tokenizer :** Tokenizer Object

**total\_words :** number of unrepeated words

**len(corpus) :** number of lines in poems

```
def TokenizePoems(Poem,Print=False):
    tokenizer = Tokenizer()
    corpus = Poem.split("\n")
    tokenizer.fit_on_texts(corpus)
    total_words = len(tokenizer.word_index) + 1
    if(Print):
        print("Words numbers ",total_words)
        print("Tokenizer word index ",tokenizer.word_index)

    return corpus,tokenizer,total_words,len(corpus)
```

- Example :

```

TokenizePoems(EleaAboMadiPoems,Print=True)

Words numbers 11148
Tokenizer word index {'11147': 'وَتَلَّيْهِ', '11146': 'فَتَلَّيْهَا'}
(['وَيَخْرُجْنَ مِنْ فَرْجَاتِ النَّعَمِ دَامِيَةً',
  'كَأَنَّ أَذَانَهَا أَطْرَفَتْ أَقْلَامًا كُلَّمَا بَلَ رِيْمًا عَيْتَ الْبُكَاءِ',
  'وَيَدْمُوعَ عَيْنَيْكَ مِنْ بُكَاءٍ خَمَامٍ',
  'وَإِذَا السَّمَاءُ مَعَ الْحَبِيِّ تَتَسَمَّتْ',
  'هَاجَ التَّنَسُّمُ لِي نَفِيْنٍ مَقَامِسًا عَيْتُ رَغَمِ الدَّاءِ وَالْأَعْدَاءِ',
  'كَالْأَسْرِ فَوْقَ الْقَيْمَةِ السَّمَاءِ',
  'أَرْزُو إِلَى السَّمْسِ الْمُضِيئَةِ هَارِئًا',
  'بِالسُّخْبِ وَالْأَمْطَارِ وَالْأَنْوَاءِ',
  'لَا أَرْمُقُ الظِّلَّ الْكَثِيبَ وَلَا أَرَى',
  'مَا فِي فَرَارِ الْهُؤُورِ السَّوْدَاءِ',
  'وَأَسِيرُ فِي دُنْيَا الْمُتَنَاعِرِ حَالِمًا',
  'تَرْدًا وَتِلْكَ مَعَادَةُ السَّعْرَاءِ',
  'أَصْنَعِي لِمُوسِيقَى الْخَيَاةِ وَوَحْيَهَا',
  'وَأَتْنِيبُ رُوحَ الْكَوْنِ فِي إِنْشَائِي',
  'وَأَصْبِيحُ لِلصَّوْتِ الْإِلَهِيِّ الَّذِي']

```

- **GenerateInputSequence(Poem,Print)**

- Take two arguments String of all poems And a print Boolean Value to print details .

- How it works:

Call **TokenizePoems(Poem,Print=False)** and receive Corpus  
 Tokenize , TotalWords , len(corpus)

For each corpus assign int value to each word in that corpus using  
 tokenizer.texts\_to\_sequences([line " From Corpus " ])

After getting list of integers that represent the corpus

Going to split that list in sequence of consecutive integer list

- Retrun

**Corpus\_num**: number of lines line in poems

**Tokenizer** : Tokenizer Object

**total\_words** : number of unrepeated words

**input\_sequences**: list of consecutive integer

```

def GenerateInputSequence(Poem,Print=False):
    corpus,tokenizer,total_words,curpus_num = TokenizePoems(Poem,Print)
    input_sequences = []
    for line in corpus:
        # مصفوفة من توالي الكلمات بصيغة انجيز
        token_list = tokenizer.texts_to_sequences([line])[0]
        for i in range(1, len(token_list)):
            n_gram_sequence = token_list[:i+1]
            input_sequences.append(n_gram_sequence)

    if(Print):
        print("Corpus length ",corpus)
        print("Corpus Sequence ",input_sequences)
        print("Corpus Sequence Length ",len(input_sequences))

    return input_sequences,tokenizer,total_words,curpus_num

```

Eplanation :

Line:

[4 2 66 8 67 68 69 70]

Input Sequences:

[4 2]

[4 2 66]

[4 2 66 8]

[4 2 66 8 67]

[4 2 66 8 67 68]

[4 2 66 8 67 68 69]

[4 2 66 8 67 68 69 70]

- **Generate\_X\_Y(Poem,Print=False)**

- Take two arguments String of all poems And a print Boolean Value to print details .

- How it works:

Call **GenerateInputSequence(Poem,Print)** and receive input sequences And Tokenizer, Total\_words And Corpus\_num .

X same As input Sequence But Missing the last item in each sublist.

Y represent the last item in each SubList .

Then padding the X to Longest line in Corpus.

Then One Hot Encoding For Y .

```
def Generate_X_Y(Poem,Print=False):
    sequences,tokenizer,words,corpus = GenerateInputSequence(Poem,Print)
    max_sequence_len = max([len(x) for x in sequences])
    sequences = np.array(pad_sequences(sequences, maxlen=max_sequence_len, padding='pre'))
    xs, labels = sequences[:, :-1], sequences[:, -1]
    ys = to_categorical(labels, num_classes=words)
    if(Print):
        print("Max Sequence Length " ,max_sequence_len )
    return xs,ys,tokenizer,words,max_sequence_len
```

- Eplanation :



# Build Model

as known there is no rules to build a sufficient NN So we try a several architectures with several Poems .

First try :

Build a neural network consists of 3 Layers :

1. Embedding Layer
2. Bidirectional(LSTM) layer
3. Dense Layer

Hyper Parameters are set as shown

```
def BuildModel(PrintDetails=False):
    model = Sequential()
    model.add(Embedding(WordsNumber, 100, input_length=MaxSequenceLength-1))
    model.add(Bidirectional(LSTM(150)))
    model.add(Dense(WordsNumber, activation='softmax'))
    adam = Adam(learning_rate=0.01)
    model.compile(loss='categorical_crossentropy', optimizer=adam, metrics=['accuracy'])
    #earlystop = EarlyStopping(monitor='val_loss', min_delta=0, patience=5, verbose=0, mode='auto')
    if(PrintDetails):
        print(model.summary())
        tf.keras.utils.plot_model(model, to_file='model.png', show_shapes=True, show_layer_names=True)
    return model
```

Second Try :

Build a neural network consists of 5 Layers :

1. Embedding Layer
2. LSTM
3. Bidirectional(LSTM) layer
4. Dense Layer
5. Dense Layer

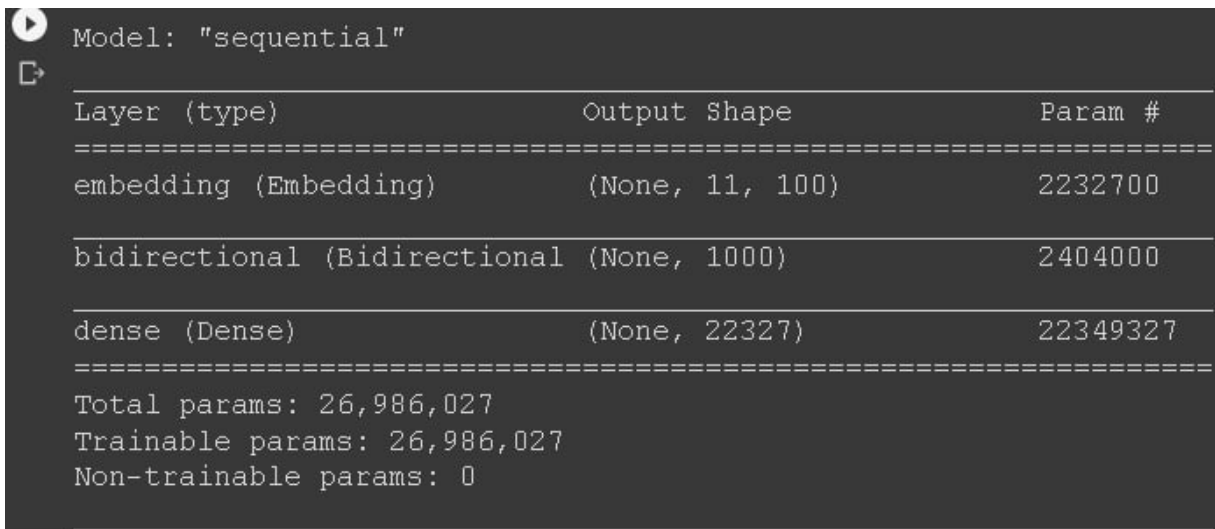
Hyper Parameters are set as shown

```
def BuildModel(PrintDetails=False):
    model = Sequential()
    model.add(Embedding(WordsNumber, 50, input_length=MaxSequenceLength-1))
    model.add(LSTM(100, return_sequences=True))
    model.add(Bidirectional(LSTM(100)))
    model.add(Dense(WordsNumber/2, activation='relu'))
    model.add(Dense(WordsNumber, activation='softmax'))
    adam = Adam(learning_rate=0.01)
    model.compile(loss='categorical_crossentropy', optimizer=adam, metrics=['accuracy'])
    #earlystop = EarlyStopping(monitor='val_loss', min_delta=0, patience=5, verbose=0, mode='auto')
    if(PrintDetails):
        print(model.summary())
        tf.keras.utils.plot_model(model, to_file='model.png', show_shapes=True, show_layer_names=True)
    return model
```

# Train And Validation

Elea Abo Madi :

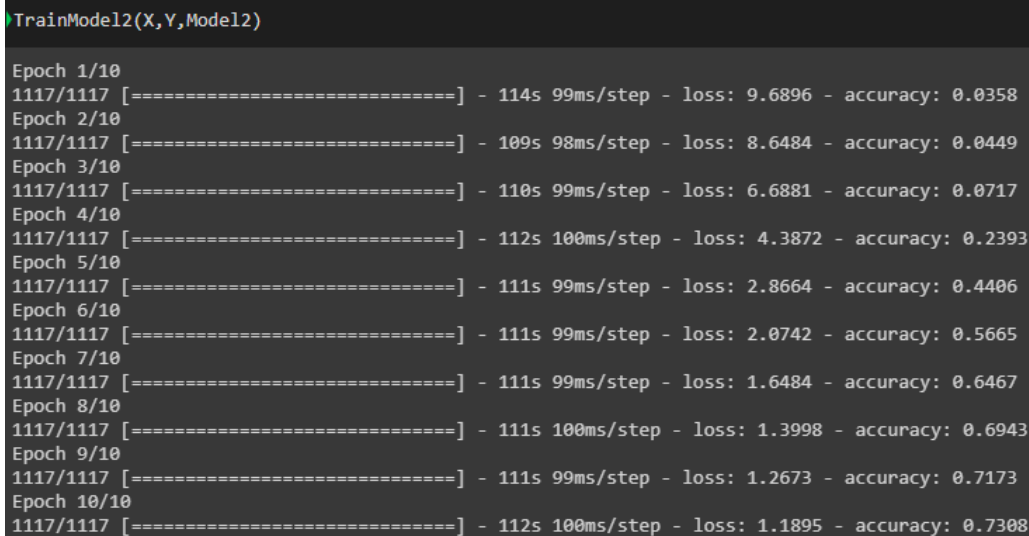
## 1. First Model



```
Model: "sequential"
Layer (type)                Output Shape              Param #
=====
embedding (Embedding)       (None, 11, 100)          2232700
bidirectional (Bidirectional (None, 1000)          2404000
dense (Dense)                (None, 22327)             22349327
=====
Total params: 26,986,027
Trainable params: 26,986,027
Non-trainable params: 0
```

The image shows a terminal window with a dark background. At the top, it says 'Model: "sequential"'. Below that is a table with three columns: 'Layer (type)', 'Output Shape', and 'Param #'. The table lists three layers: 'embedding (Embedding)' with output shape '(None, 11, 100)' and 2232700 parameters; 'bidirectional (Bidirectional)' with output shape '(None, 1000)' and 2404000 parameters; and 'dense (Dense)' with output shape '(None, 22327)' and 22349327 parameters. Below the table, it shows 'Total params: 26,986,027', 'Trainable params: 26,986,027', and 'Non-trainable params: 0'.

## Train with 10 epoch



```
TrainModel12(X,Y,Model12)

Epoch 1/10
1117/1117 [=====] - 114s 99ms/step - loss: 9.6896 - accuracy: 0.0358
Epoch 2/10
1117/1117 [=====] - 109s 98ms/step - loss: 8.6484 - accuracy: 0.0449
Epoch 3/10
1117/1117 [=====] - 110s 99ms/step - loss: 6.6881 - accuracy: 0.0717
Epoch 4/10
1117/1117 [=====] - 112s 100ms/step - loss: 4.3872 - accuracy: 0.2393
Epoch 5/10
1117/1117 [=====] - 111s 99ms/step - loss: 2.8664 - accuracy: 0.4406
Epoch 6/10
1117/1117 [=====] - 111s 99ms/step - loss: 2.0742 - accuracy: 0.5665
Epoch 7/10
1117/1117 [=====] - 111s 99ms/step - loss: 1.6484 - accuracy: 0.6467
Epoch 8/10
1117/1117 [=====] - 111s 100ms/step - loss: 1.3998 - accuracy: 0.6943
Epoch 9/10
1117/1117 [=====] - 111s 99ms/step - loss: 1.2673 - accuracy: 0.7173
Epoch 10/10
1117/1117 [=====] - 112s 100ms/step - loss: 1.1895 - accuracy: 0.7308
```

The image shows a terminal window with a dark background. It starts with 'TrainModel12(X,Y,Model12)'. Then it shows 10 epochs of training progress. Each line includes the epoch number, a progress bar (1117/1117 [=====]), time per step, loss, and accuracy. The accuracy increases from 0.0358 in epoch 1 to 0.7308 in epoch 10.

Accuracy : 0.73



Train time : 2 h

```
PredictNextOf(tokenizer,Model2,MaxSequenceLength,Sentence = "يا نفس هذا منزل الاحباب",PredectionLength=20)
```

/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/engine/sequential.py:455: UserWarning: warnings.warn("`model.predict\_classes()` is deprecated and will be removed in a future version. Please use `model.predict` instead.")

يا نفس هذا منزل الاحباب أُخِثَ دار الخُلد يا أُمَّ الْفَرى الأَسى الْكرى وَأَطْرَني العاري عَلَيكُمْ ما الْكرى ما فَعَلوا الْورى تَأْمَلُ الْورى صَبِيًا فِيهِ

```
PredictNextOf(tokenizer,Model2,MaxSequenceLength,Sentence = "لو قل",PredectionLength=40)
```

ib/python3.7/dist-packages/tensorflow/python/keras/engine/sequential.py:455: UserWarning: `model.predict\_classes()` is deprecated and will be removed in a future version. Please use `model.predict` instead.

arn("`model.predict\_classes()` is deprecated and will be removed in a future version. Please use `model.predict` instead.")

لو قال من ثَرى مَزَقُها بِيَدِ الثرى مُنْتَقِلًا وَأَنْ يَحْتَقِ وَأَنْ تَشْفَى فَلَإِيكَ سِتْرا وَيَعُدُّ الْورى لَمْ الْورى كُلُّ يَوْهَبُ الْورى لَكِنْ أَنَا فِيهِ صَنِيتُ جَاوِيَهُ نَشِيدُهُ نَشِيدُهُ خَلَقَتْ لَهَا أَهَذَا كَوَكَبُ الشَّمَلِ فِي الْخَبِ لَيْسَ الْورى تَأْمَلُ

```
PredictNextOf(tokenizer,Model2,MaxSequenceLength,Sentence = "بصر عن ذا اللب",PredectionLength=40)
```

al/lib/python3.7/dist-packages/tensorflow/python/keras/engine/sequential.py:455: UserWarning: `model.predict\_classes()` is deprecated and will be removed in a future version. Please use `model.predict` instead.

gs.warn("`model.predict\_classes()` is deprecated and will be removed in a future version. Please use `model.predict` instead.")

وَذَلِكَ ثَرَا إِنَّمَا التَّلَوُّ جَامِدًا الْقَضَا إِلِ إِنَّمَا صَوْرَتُهَا هُنَا كَالسِرِّ الْيَمِينُ فِيهِ الْورى لَا النَّاسِ النَّاسِ لَكِنْ لَمْ أَجِدْنِي تَسْتَقِي وَلَا الْورى مُوسَى هَذِي الْخَطُوبَا الشَّمَلُ الشَّمَلُ النَّاسِ لَكِنْ لَكِنْ تَخْلُقُ فِيهِ غَيْرُهُ وَاعْتِبَابُ

## 2. Second Model

Model: "sequential\_1"

Layer (type)	Output Shape	Param #
embedding_1 (Embedding)	(None, 11, 50)	1116350
lstm_1 (LSTM)	(None, 11, 100)	60400
bidirectional_1 (Bidirection	(None, 200)	160800
dense_1 (Dense)	(None, 11163)	2243763
dense_2 (Dense)	(None, 22327)	249258628
Total params: 252,839,941		
Trainable params: 252,839,941		
Non-trainable params: 0		
None		

TrainModel(X,Y,Model)

```
Epoch 1/10
1117/1117 [=====] - 2526s 2s/step - loss: 9.5575 - accuracy: 0.0379
Epoch 2/10
1117/1117 [=====] - 2495s 2s/step - loss: 8.9514 - accuracy: 0.0384
Epoch 3/10
1117/1117 [=====] - 2482s 2s/step - loss: 8.6518 - accuracy: 0.0381
Epoch 4/10
1117/1117 [=====] - 2480s 2s/step - loss: 8.4070 - accuracy: 0.0384
Epoch 5/10
1117/1117 [=====] - 2475s 2s/step - loss: 8.1877 - accuracy: 0.0397
Epoch 6/10
1117/1117 [=====] - 2478s 2s/step - loss: 7.9929 - accuracy: 0.0402
Epoch 7/10
1117/1117 [=====] - 2486s 2s/step - loss: 7.8068 - accuracy: 0.0411
Epoch 8/10
1117/1117 [=====] - 2484s 2s/step - loss: 7.6131 - accuracy: 0.0443
Epoch 9/10
790/1117 [=====>.....] - ETA: 12:05 - loss: 7.3531 - accuracy: 0.0485
```

uation

in another tab. [Show diff](#) r,Model,MaxSequenceLength,Sentence = "",PredectionLength=0):

4h 53m 33s completed at 2:10 PM

Google Colab Crushed after 5 hours in waiting .....

So the results will be delivered tomorrow at the session 😞

At the training we can see tat the accuracy is improving So by increasing the epoch the improvement will be grater .....

Accuracy :

Train time :

To Complete Text : ""