به نام خدا

دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

شبکه های عصبی و یادگیری عمیق

مینی‌پروژه سری دوم

نام و نام خانوادگی	شایان واصف – آرمان اکبری
شماره دانشجویی	810197456 - 810197603
تاریخ ارسال گزارش	دی‌ماه 1400

فهرست گزارش سوالات

سوال 1 – Music Generation

➢ Chopin:

در وهله اول ، مراحل زیر را پیش می‌بریم :

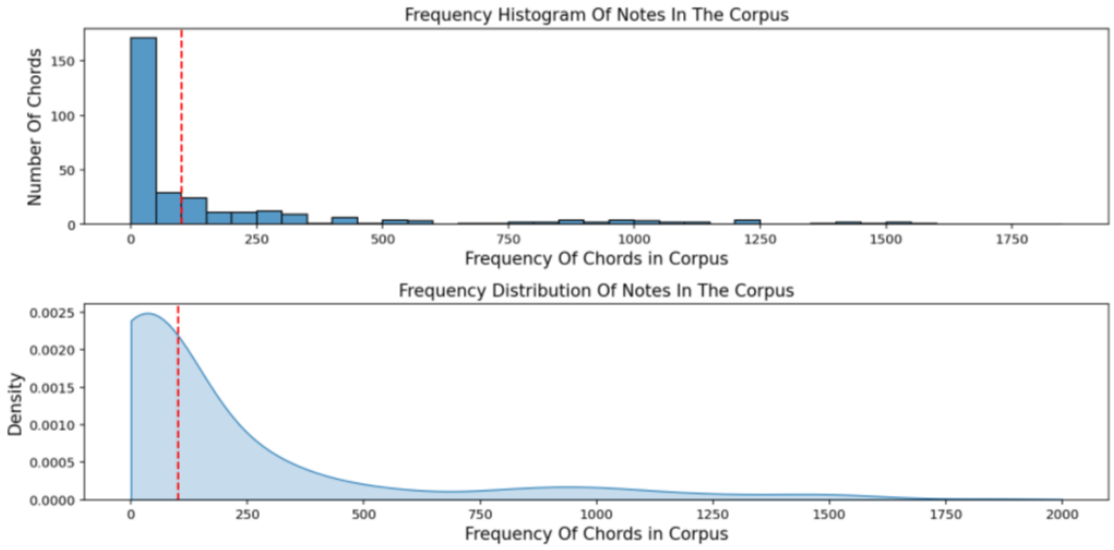
Data Exploration:

- Exploring the data Corpus
- Examine all the notes in the Corpus
- Simplifying our Corpus

برای مثال ، 50 نوت اول موجود در Corpus بدست آمده را در زیر آورده ایم :

['B-4', 'B-4', 'B-4', 'B-4', 'B-4', 'B-4', 'B-4', 'B-4', 'B-4', 'B-4', 'B-4', 'B-4', 'B-4',
'B-4', 'B-4', 'B-4', 'B-2', 'D5', '5.8.10', 'E-5', '2.5', '5.8.10', 'B-4', 'E-3', 'E-5',
'3.7.10', 'F5', '3.7', '3.7.10', 'B-4', 'D3', 'F5', '5.8.10', 'G5', '5.8', '5.8.10', '7.10.1',
'E-3', '7.10.1', '7.10.1', 'E3', '7.10.1', 'B-5', '1.7', 'F3', 'C6', '5.8', 'B-5', '8.0', 'G#2']

برای برسی نحوه توزیع آکورد(Chords) های بدست آمده ، به ترتیب Count-plot و Kde آنرا رسم می‌کنیم :



شکل 1-1 : Count-plot و Kde آکورد های بدست آمده

طبق دو نمودار بالا ، تقریبا تعداد زیادی از نوت ها کمتر از 100 بار در corpus تکرار شده اند ، بنابراین نوت هایی که کمتر از 100 بار در corpus تکرار شده اند را دور می‌ریزیم .

تعداد نوت های دارای این شرایط را با دستور زیر می‌خوانیم :

```
Total number of notes that occur less than 100 times: 200
```

شکل 1-2 : تعداد نوت های با تکرار کمتر از 200

طول جدید بدست آمده corpus با حذف تعداد نوت گفته شده به صورت زیر می‌باشد :

```
Length of Corpus after elemination the rare notes: 59854
```

شکل 1-3 : طول جدید corpus

در ادامه پردازش بر روی دادگان را با مراحل نشان داده شده در زیر انجام می‌دهیم

DATA PREPROCESSING

- Creating a dictionary
- Encoding and Splitting the corpus
- Assigning X and y
- Splitting Train and Seed datasets
- Creating a list of sorted unique characters

برای تعریف sequence ، طول هر کدام (window-size) را برابر 40 می‌گیریم و در قالب کد زیر شکل می‌دهیم :

Splitting the Corpus in equal length of strings and output target

```
[ ]  def input_data(seq,ws):  # ws is the window size
         out = []
         L = len(seq)
         for i in range(L-ws):
             window = seq[i:i+ws]
             label = seq[i+ws:i+ws+1]
             out.append((window,label))
         return out
```

شکل 1-4 : نحوه ایجاد Sequence

تعداد تمامی Sequence های بدست آمده برابر طول corpus حاصل از حذف 200 نوت نادر ، منهای طول هر sequence (40) می‌باشد :

```
Total number of sequences in the Corpus: 59814
```

شکل 1-5 : تعداد Sequence بدست آمده از corpus

در نهایت به تعریف مدل می پردازیم :

Model Building

- Initializing the Model
- Defining by adding layers
- Compiling the Model
- Training the Model

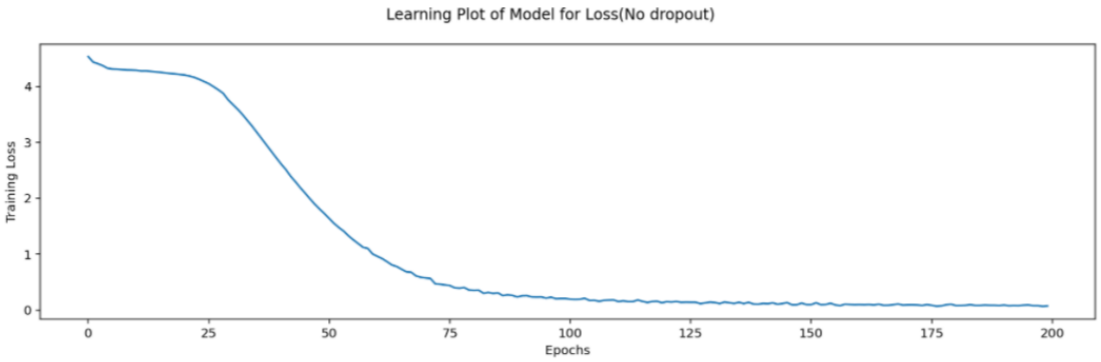
بدون dropout :

خلاصه مدل به صورت زیر می‌باشد :

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 40, 512)           1052672

 lstm_1 (LSTM)               (None, 256)               787456

 dense (Dense)               (None, 256)               65792

 dense_1 (Dense)             (None, 228)               58596

=================================================================
Total params: 1,964,516
Trainable params: 1,964,516
Non-trainable params: 0
_________________________________________________________________
```

شکل 1-6 : خلاصه مدل

با تعریف مدل ، به آموزش آن بر روی Sequence های بدست آمده می‌پردازیم . نمودار خطا بر حسب تعداد Epoch برای 200 epochs آورده شده است :



شکل 1-7 : نمودار خطا بر حسب Epoch

در نهایت ابتدا فایل midi. بدست آمده به Wav. تبدیل کرده و سپس حجم آنرا کاهش می‌دهیم . در نهایت با کمک دستور display. Audio. ، موزیک تولید شده را در notebook مربوط به آهنگساز Chopin ، ایجاد می‌کنیم:

```
#to play audio or corpus
IPython.display.Audio("../content/drive/MyDrive/Colab Notebooks/kalhor_shabake/MINI#2/Q1/Input_Output/Melody_Generated1.wav")
```

شکل 1-8 : آهنگ بدست آمده برای 100 نوت ابتدایی

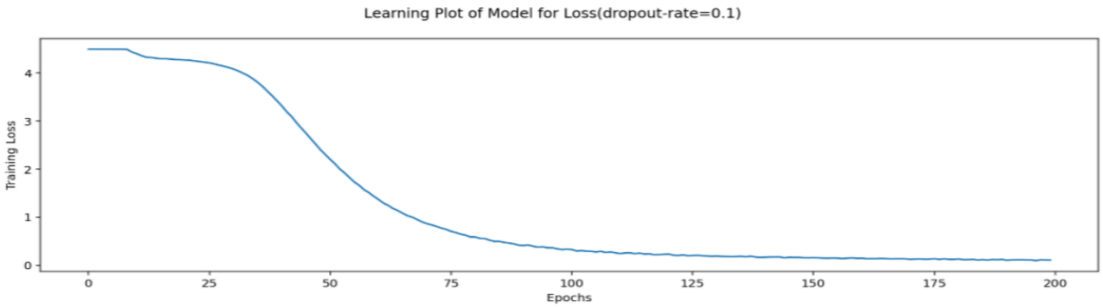
همراه dropout :

در این قسمت ، به مدل قسمت قبل دو لایه dropout با نرخ 0.1 ایجاد می‌کنیم :

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_2 (LSTM)               (None, 40, 512)           1052672

 dropout (Dropout)           (None, 40, 512)           0

 lstm_3 (LSTM)               (None, 256)               787456

 dense_2 (Dense)             (None, 256)               65792

 dropout_1 (Dropout)         (None, 256)               0

 dense_3 (Dense)             (None, 228)               58596

=================================================================
Total params: 1,964,516
Trainable params: 1,964,516
Non-trainable params: 0
_________________________________________________________________
```

شکل 1-6 : خلاصه مدل همراه Dropout

نمودار خطا بر حسب تعداد Epoch برای 200 epochs آورده شده است :



شکل 1-7 : نمودار خطا بر حسب Epoch

موزیک تولید شده را در notebook مربوط به نوازنده Chopin ، ایجاد می‌کنیم:

```
IPython.display.Audio("../content/drive/MyDrive/Colab Notebooks/kalhor_shabake/MINI#2/Q1/Input_Output/Melody_Generated2.wav")
```

0:48 / 0:48

با شنیدن هر دوی آهنگ های تولید شده ، بنظر می‌رسد که نحوه عوض شدن آکورد ها در هر دو یکی بوده ولی در حالت drop-out ، نوت ایجاد شده به گوش مخاطب ، آشناتر و منطقی تر بنظر می‌رسد.

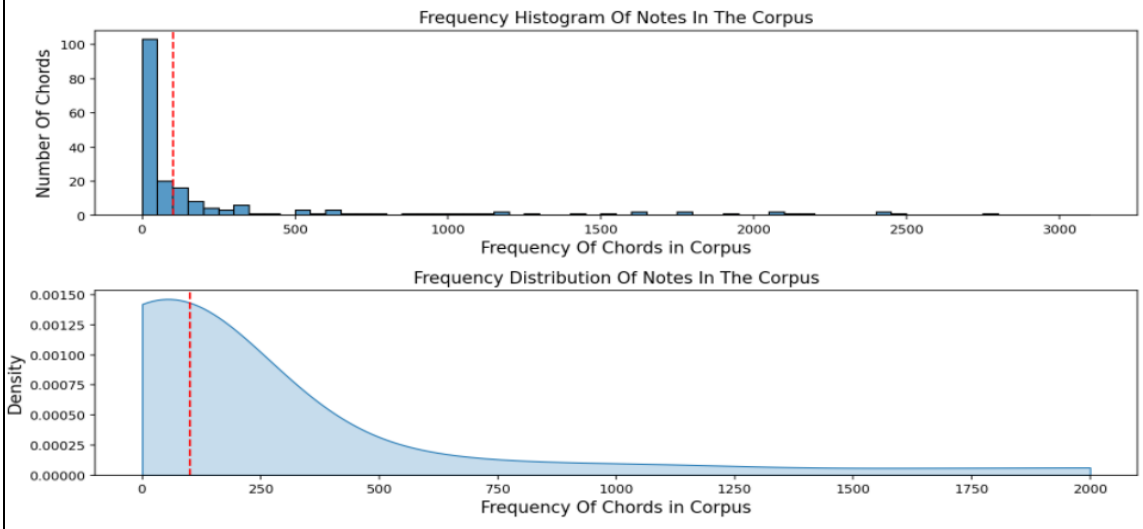
➢ Mozart:

تمامی مراحل ذکر شده در بالا را اینبار برای آهنگساز Mozart تکرار می‌کنیم :

50 نوت اول موجود در Corpus بدست آمده را در زیر آورده ایم :

['B4', 'A4', 'G#4', 'A4', 'C5', 'A3', '0.4', 'D5', '0.4', 'C5', 'B4', '0.4', 'C5', 'E5', 'A3', '0.4', 'F5', '0.4', 'E5', 'E-5', '0.4', 'E5', 'B5', 'A3', 'A5', 'G#5', '0.4', 'A5', 'B5', 'A3', 'A5', 'G#5', '0.4', 'A5', 'C6', 'A3', '0.4', 'A5', '0.4', 'C6', '0.4', '7.9', 'B5', 'E3', '6.9', '11.4', '4.7', '11.4', '6.9', '11.4']

برای برسی نحوه توزیع آکورد(Chords) های بدست آمده ، به ترتیب Count-plot و Kde آنرا رسم می‌کنیم :



شکل 1-1 : Count-plot و Kde آکورد های بدست آمده

همانطور که مشاهده می‌شود ، توزیع بدست آمده همانند قسمت پیشین می‌باشد ، بنابراین سیاست پیشین برای حذف نوت های کم تکرار انتخاب می‌کنیم :

تعداد نوت های دارای کمتر از 100 تکرار را با دستور زیر می‌خوانیم :

```
Total number of notes that occur less than 100 times: 123
```

شکل 1-2 : تعداد نوت های با تکرار کمتر از 200

طول جدید بدست آمده corpus با حذف تعداد نوت گفته شده به صورت زیر می‌باشد :

```
Length of Corpus after elemination the rare notes: 57247
```

شکل 1-3 : طول جدید corpus

برای تعریف sequence ، طول هر کدام (window-size) را برابر 40 می‌گیریم و در قالب کد قسمت قبل آنرا شکل می‌دهیم .

تعداد تمامی Sequence های بدست آمده برابر طول corpus حاصل از حذف 123 نوت نادر ، منهای طول هر sequence (40) می‌باشد :

```
Total number of sequences in the Corpus: 57207
```

شکل 1-5 : تعداد Sequence بدست آمده از corpus

در نهایت به تعریف مدل می پردازیم :

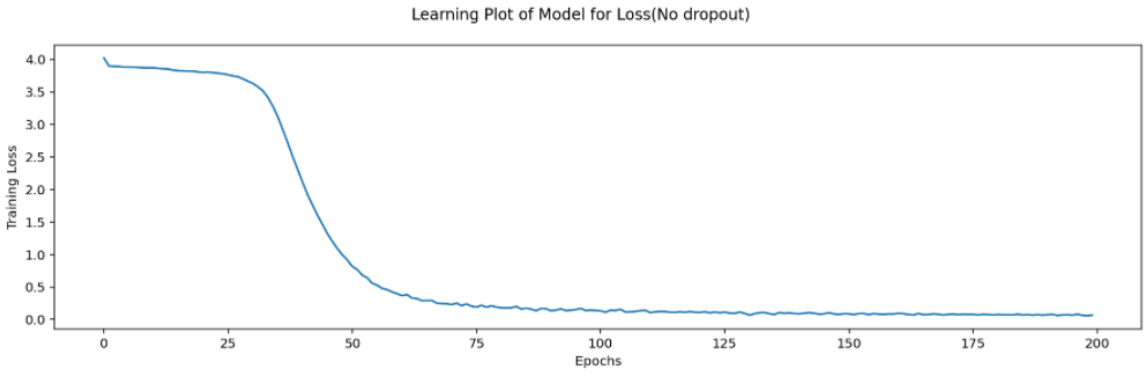
بدون drop-out :

خلاصه مدل به صورت زیر می‌باشد :

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 40, 512)           1052672

 lstm_1 (LSTM)               (None, 256)               787456

 dense (Dense)               (None, 256)               65792

 dense_1 (Dense)             (None, 140)               35980

=================================================================
Total params: 1,941,900
Trainable params: 1,941,900
Non-trainable params: 0
_________________________________________________________________
```

شکل 1-6 : خلاصه مدل بدون Dropout

با تعریف مدل ، به آموزش آن بر روی Sequence های بدست آمده می‌پردازیم . نمودار خطا بر حسب تعداد Epoch برای 200 epochs آورده شده است :



شکل 1-7 : نمودار خطا بر حسب Epoch

موزیک تولید شده را در notebook مربوط به نوازنده Chopin ، ایجاد می‌کنیم:

```
IPython.display.Audio("../content/drive/MyDrive/Input_Output/Melody_Generated3.wav")
```

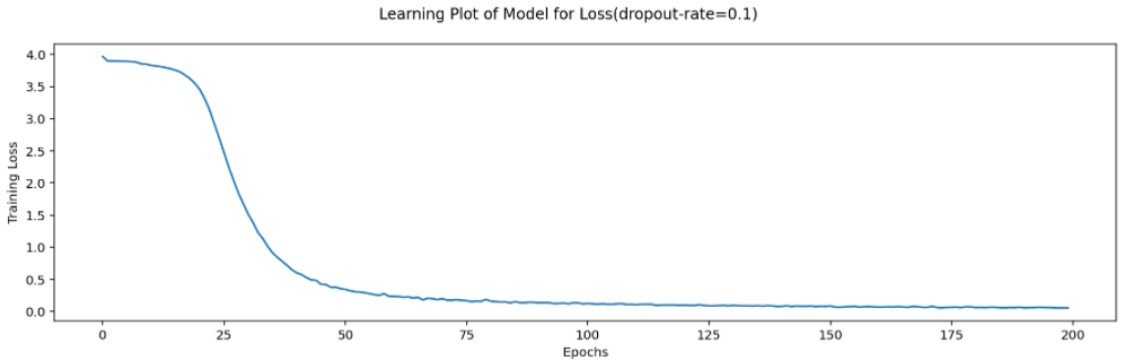
همراه dropout :

در این قسمت ، به مدل قسمت قبل دو لایه dropout با نرخ 0.1 ایجاد می‌کنیم :

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_2 (LSTM)               (None, 40, 512)           1052672

 dropout (Dropout)           (None, 40, 512)           0

 lstm_3 (LSTM)               (None, 256)               787456

 dense_2 (Dense)             (None, 256)               65792

 dropout_1 (Dropout)         (None, 256)               0

 dense_3 (Dense)             (None, 140)               35980

=================================================================
Total params: 1,941,900
Trainable params: 1,941,900
Non-trainable params: 0
_________________________________________________________________
```

شکل 1-6 : خلاصه مدل همراه Dropout

نمودار خطا بر حسب تعداد Epoch برای 200 epochs آورده شده است :



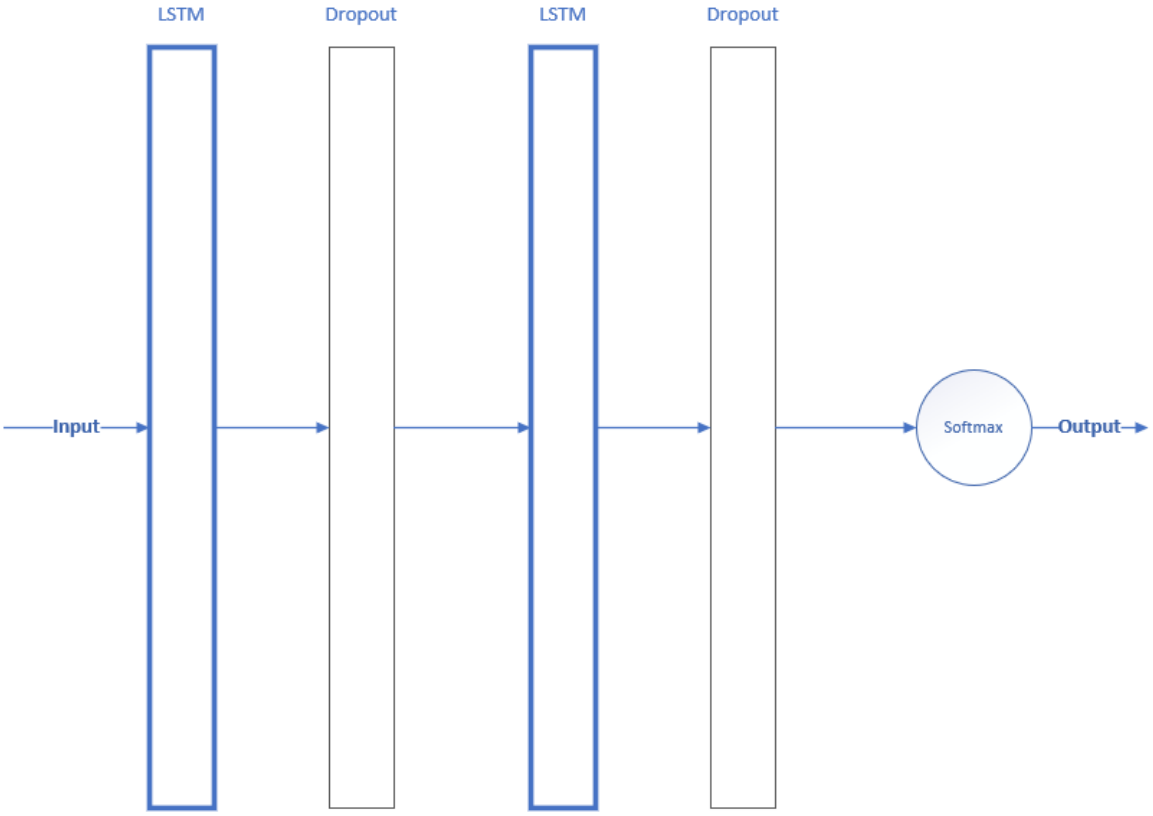
شکل 1-7 : نمودار خطا بر حسب Epoch

موزیک تولید شده را در notebook مربوط به نوازنده Chopin ، ایجاد می‌کنیم:

```
IPython.display.Audio("../content/drive/MyDrive/Input_Output/Melody_Generated4.wav")
```

سوال 3 – Lyric Generation

در این سوال با استفاده از شبکه‌های عصبی بازگشتی یا به اختصار RNN، به دنبال آموزش مدلی هستیم که با دریافت چند حرف ابتدایی، یک بیت شعر[1] تولید نماید. برای اینکار معماری زیر را برای مدل در نظر گرفته‌ایم:



شکل 3-1. معماری مدل سوال سه با استفاده از لایه‌های RNN و Dropout

به این صورت که ورودی در ابتدا وارد یک لایه LSTM با ابعاد مشخص شده، خروجی‌های آن از Dropout گذشته و وارد LSTM بعدی شده و در نهایت پس از گذشت مجدد از لایه Dropout، به تابع فعالساز Softmax رسیده و خروجی این تابع، خروجی کل مدل خواهد بود.

پس از طراحی مدل، به سراغ پیاده‌سازی آن روی مجموعه دادگان "Lyrics Dataset" می‌رویم. در ابتدا اطلاعات موجود را خوانده و قسمتی از آن را بررسی می‌کنیم:

[1] Lyric

	Artist Name	Song Name	Lyrics
0	Phoebe Bridgers	Motion Sickness	I hate you for what you did And I miss you li...
1	Phoebe Bridgers	Killer	Sometimes I think I'm a killer I scared you i...
2	Phoebe Bridgers	Georgia	Georgia, Georgia, I love your son And when he...
3	Phoebe Bridgers	Kyoto	Day off in Kyoto Got bored at the temple Look...
4	Phoebe Bridgers	Would You Rather	Playing "would you rather" When it comes to f...

شکل 3-2. قسمت ابتدایی مجموعه دادگان مورد بررسی

همان‌طور که دیده می‌شود این دادگان دارای 3 ستون از اطلاعات هستند که به ترتیب شامل نام خواننده آهنگ، نام آهنگ و متن اشعار آن ترانه می‌باشند. با توجه به اینکه بدنبال ساختن یک بیت شعر هستیم، بنابراین دو ستون اول، یعنی نام خواننده و آهنگ برای ما اهمیت و استفاده‌ای ندارند. در واقع دادگان آموزش‌ما مربوط به بخش Lyrics یا متن ترانه می‌باشد. برای اینکه این مجموعه حروف برای هدف ما مناسب باشند نیاز به انجام پیش‌پردازش‌هایی روی آنهاست تا مدل RNN ما بتواند با استفاده از آنها آموزش ببیند. برای انجام این پیش‌پردازش‌ها، به ترتیب مراحل زیر را طی می‌کنیم:

در ابتدا تمامی کلمه‌های ترانه‌های موجود در ستون Lyrics را داخل متغیری به نام set_of_words ذخیره می‌نماییم. در ادامه با خروجی گرفتن تعداد حروف منحصر بفرد، می‌بینیم که 72 حرف وجود دارد که با توجه به تعداد حروف الفبای انگلیسی و اعداد 1 تا 10، همچنان عدد بالاییست. در نتیجه متوجه می‌شویم که تعداد از حروف موجود در متن ترانه‌های دیتاست ما دارای حروف خاص هستند که ما علاقه‌ای به حضور آن‌ها در شعر خود نداریم. بنابراین با پیدا کردن این حروف، آن‌ها را از مجموعه حروف خود یا همان set_of_words حذف می‌نماییم.

در ادامه برای اینکه سیستم توانایی درک حروف را داشته باشد نیاز به یک سیستم Encoding داریم تا حروف را به اعداد نگاشت دهد. برای اینکار از دیکشنری استفاده کرده و مجموعه حروف را که ترتیب قرارگیری آن را می‌دانیم، به فضای اعداد نگاشت می‌دهیم. در ادامه برای اینکه بتوانیم این مجموعه دادگان نسبتا بزرگ را به عنوان خروجی به شبکه بدهیم، آن را بخش بخش می‌کنیم.

در مرحله آخر از پیش‌پردازش ورودی آموزش را برابر این بخش‌های بدست آمده قرار داده و آن را نرمالایز می‌کنیم و خروجی آموزش را به صورت one-hot تعریف می‌نماییم.

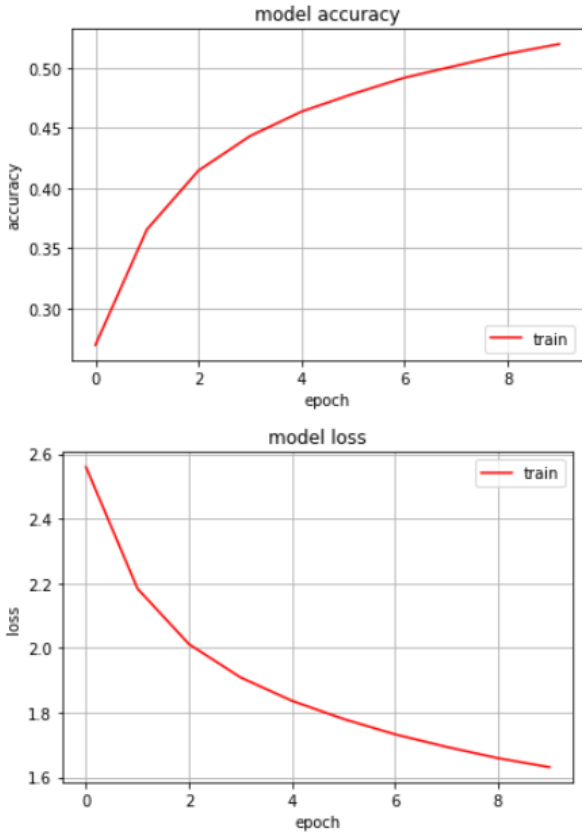
اکنون پس از انجام اعمال مربوط به پیش‌پردازش مدل خود را که معماری آن در شکل 3-1 موجود است پیاده‌سازی می‌کنیم. گزارش ساخت مدل به صورت زیر می‌باشد:

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 lstm_2 (LSTM)                (None, 40, 256)           264192

 dropout_2 (Dropout)          (None, 40, 256)           0

 lstm_3 (LSTM)                (None, 256)               525312

 dropout_3 (Dropout)          (None, 256)               0

 dense_1 (Dense)              (None, 51)                13107

=================================================================
Total params: 802,611
Trainable params: 802,611
Non-trainable params: 0
_________________________________________________________________
```

شکل 3-3. خلاصه گزارش مدل ساخته شده برای سوال 3

الف)

در این بخش مدل آموزش داده شده را از لحاظ دقت و خطا بررسی خواهیم کرد. پس از کامپایل مدل به ازای تابع هزینه "Categorical Crossentropy" بهینه‌ساز "Adam" نتایج زیر حاصل شده‌اند:



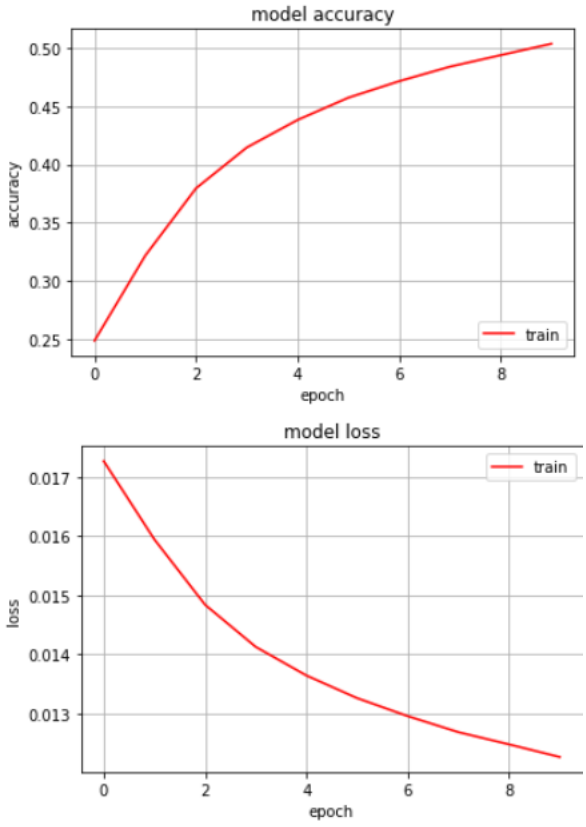
شکل 3-4. نمودار تغییرات دقت و خطا مدل در مدت آموزش

پیش‌پردازش‌های انجام شده بر روی داده‌ها کمک شایانی به عملکرد آن می‌کنند. در ابتدا نگاشت و تبدیل حروف به یک صفحه اعداد به طور کلی امکان آموزش داده‌ها را فراهم می‌کند. با توجه به ماهیت مسئله که تک خروجی categorical دارد، اعمال one-hot encoding و همچنین با توجه به اینکه تمام حروف و کلمات دارای ارزش یکسانی هستند، نرمالیزیشن اطلاعات ورودی دقت سیستم در به حافظه سپاری و یادگیری داده‌ها را بالا می‌برد.

ب)

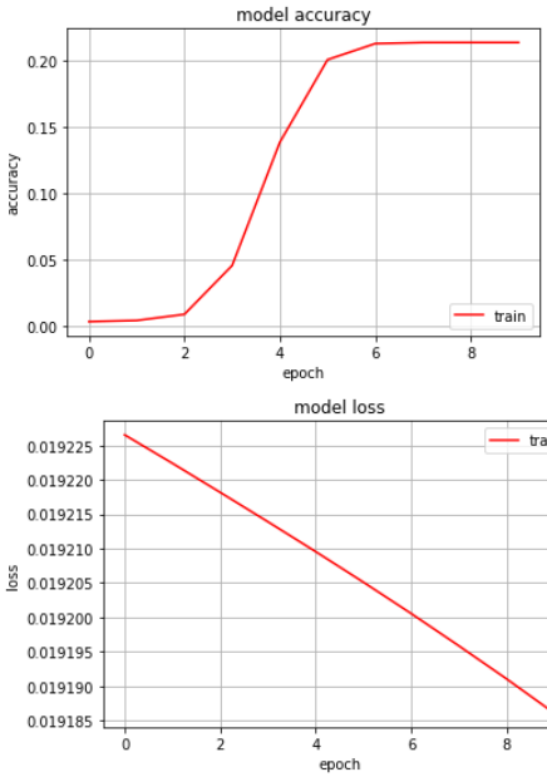
برای این بخش از تابع هزینه‌های MAE، MSE، Categorical Crossentropy و توابع بهینه‌ساز Adams،Adadelta و SGD استفاده کرده‌ایم که نتایج هر یک از آنها در ادامه آمده است:

بهینه‌ساز adam – تابع هزینه MSE:



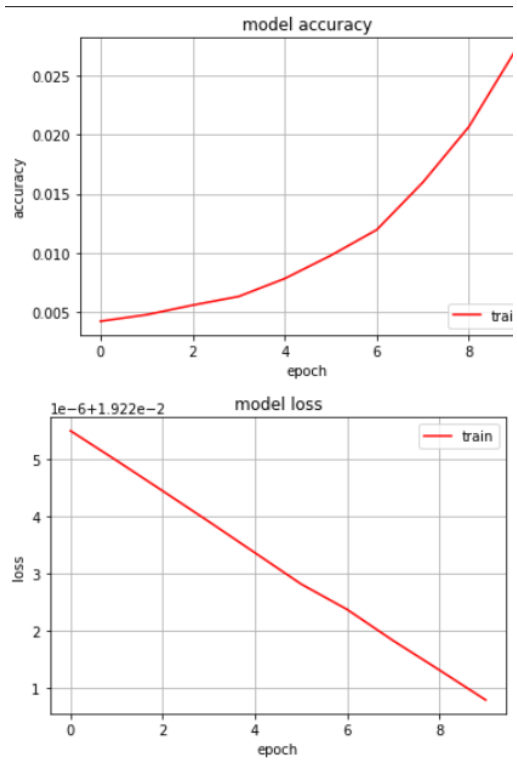
شکل 3-5. نمودار تغییرات خطا و دقت مدل شبکه RNN برای MSE-adam

بهینه‌ساز sgd – تابع هزینه MSE:



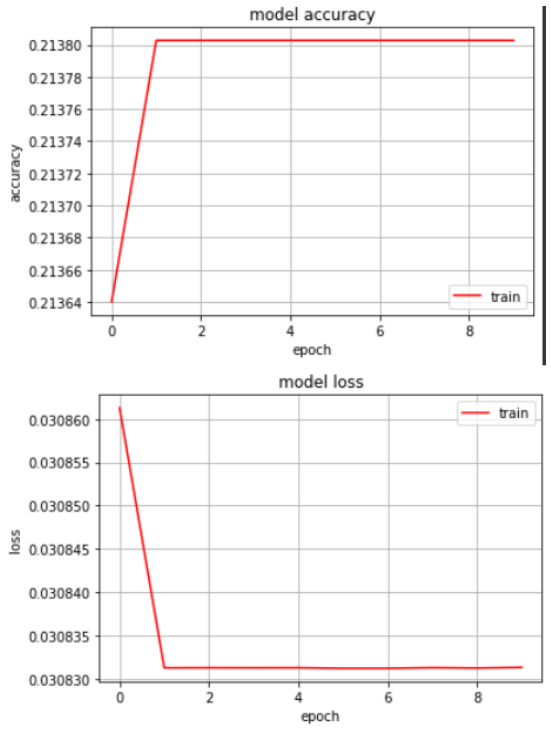
شکل 3-6. نمودار تغییرات خطا و دقت مدل شبکه RNN برای MSE - sgd

بهینه‌ساز adadelta – تابع هزینه MSE:



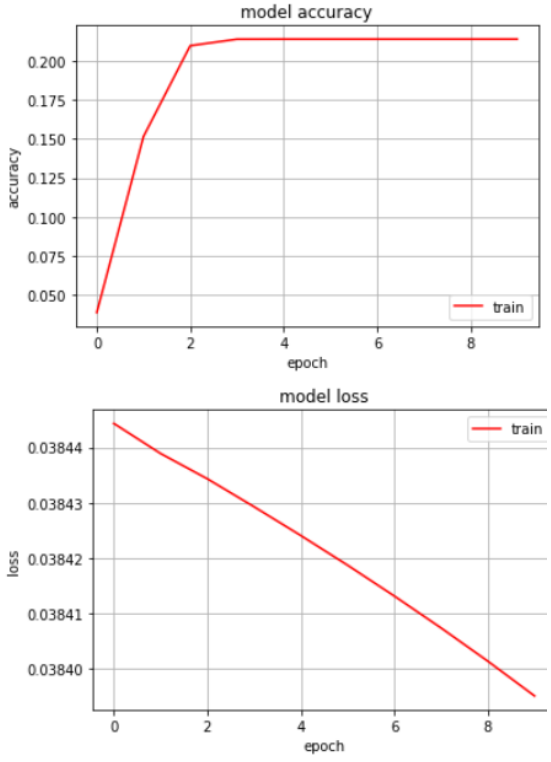
شکل 3-7. نمودار تغییرات خطا و دقت مدل شبکه RNN برای MSE-adadelta

بهینه‌ساز adam – تابع هزینه MAE:



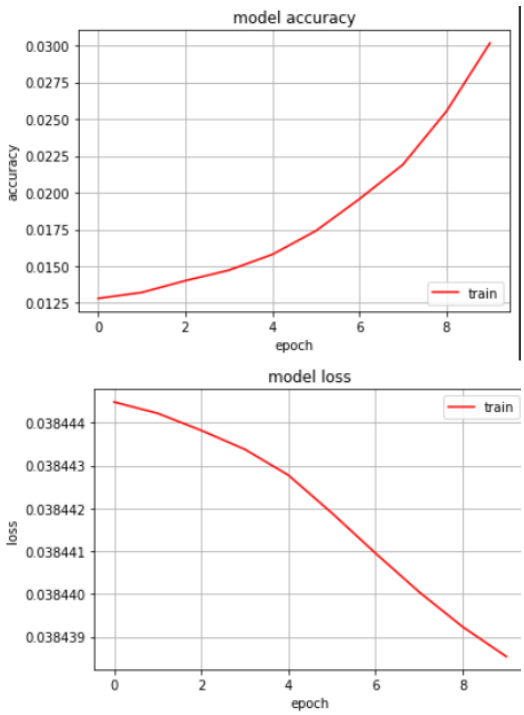
شکل 3-8. نمودار تغییرات خطا و دقت مدل شبکه RNN برای MAE-adam

بهینه‌ساز sgd – تابع هزینه MAE:



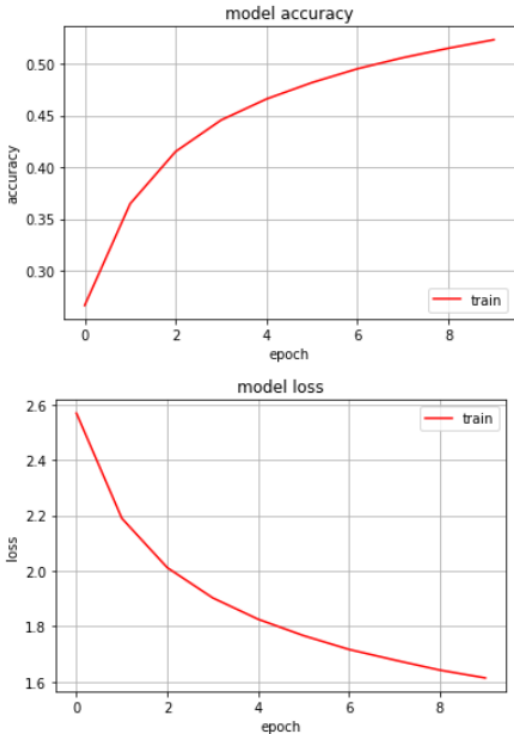
شکل 3-9. نمودار تغییرات خطا و دقت مدل شبکه RNN برای MAE - sgd

بهینه‌ساز adadelta – تابع هزینه MAE:



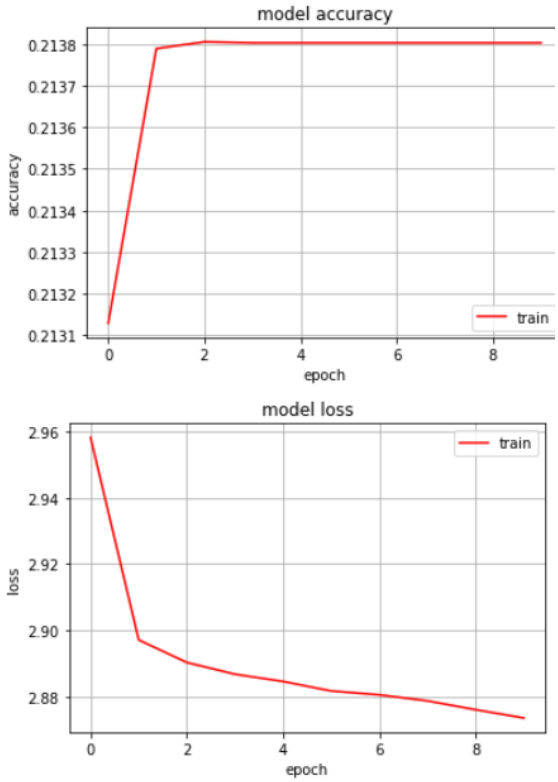
شکل 3-10. نمودار تغییرات خطا و دقت مدل شبکه RNN برای MAE - adadelta

بهینه‌ساز adam – تابع هزینه Categorical Crossentropy:



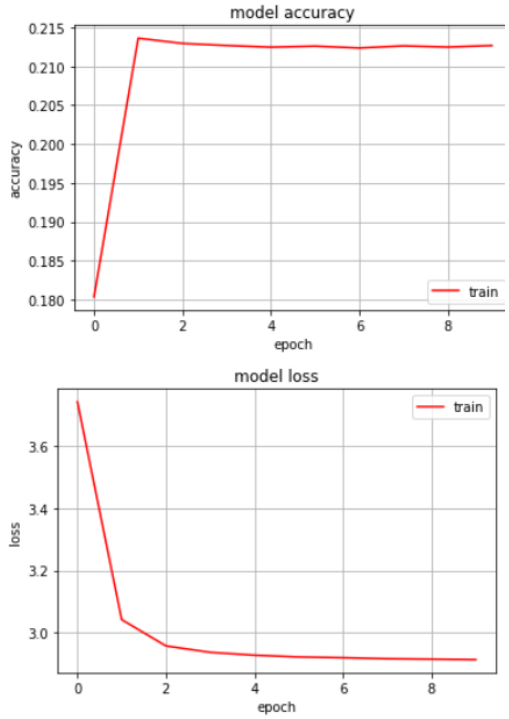
شکل 3-11. نمودار تغییرات خطا و دقت مدل شبکه RNN برای Categorical Crossentropy - adam

بهینه‌ساز sgd – تابع هزینه Categorical Crossentropy:



شکل 3-12. نمودار تغییرات خطا و دقت مدل شبکه RNN برای Categorical Crossentropy - sgd

بهینه‌ساز adadelta – تابع هزینه Categorical Crossentropy:

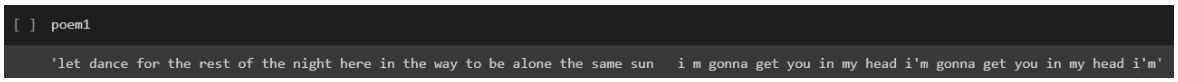


شکل 3-13. نمودار تغییرات خطا و دقت مدل شبکه RNN برای Categorical Crossentropy - adadelta

با توجه به نمودارهای دقت و خطای بدست آمده برای 3 تابع هزینه و 3 بهینه‌ساز مختلف ما، بهترین ترکیب‌های adam-Categorical Crossentropy و MSE-adam هستند که دقتی حدود برابر 50% دارند. همینطور اگر چه برای مدل‌هایی با تابع هزینه MAE پایینترین مقدار هزینه را داریم اما از طرفی دقت پایین‌تری دریافت خواهیم کرد که با توجه به ماهیت مسئله، دقت بالا اهمیت زیادی دارد.

پ)

در زیر بیت کوتاه تولید شده توسط مدل برگزیده خود را می‌بینیم:



شکل 3-14. شعر تولید شده توسط مدل برای سوال سه

که به ازای ورودی:

```
let dance for the rest of the night here
```

با توجه به اینکه یک بیت شعر به عنوان خروجی خواسته شده است، خروجی زیر را برای ما ساخته است:

```
let dance for the rest of the night here in the way to be alone the
same sun  i m gonna get you in my head i'm gonna get you in my head
```

تفاوتی که حروف و کلمات ساخته شده ابتدایی نسبت به کلمات و حروف متاخر دارند این است که هر چه جلوتر می‌رویم حروف و کلمات تکراری بیشتری می‌بینیم که می‌تواند ریشه در حافظه دار بودن شبکه داشته باشد.

ت)

راه‌ها و روش‌های متفاوتی برای کاهش هزینه محاسباتی وجود دارد. برای مثال مانند کاری که ما کرده‌ایم، یکی از راه‌ها کم کردن تعداد ایپاک‌ها است. در ابتدا تعداد ایپاک‌های مورد استفاده 100 تا بود که با بررسی انجام شده، علاوه بر زمانگیری و هزینه محاسباتی بسیار بالا، پس از ایپاک حدودا دهم، تغییر مثبت چندانی در هزینه و دقت سیستم مشاهده نمی‌شود و حتی گاها در ایپاک‌های بالاتر کاهش دقت و افزایش هزینه هم داشتیم که راه مطلوبی نمی‌باشد.

همچنین با محدود تر کردن حروف مورد استفاده می‌توان مجددا هزینه محاسباتی را کاهش داد به این صورت که برای مثال تنها حروف انگلیسی کوچک را نگه داشته و باقی حروف را حذف کنیم.

همچنین اضافه کردن لایه Dropout بین لایه‌ها باعث افزایش دقت مدل می‌شود.

ث)

کارکرد حافظه در مدل ما به این صورت است که با توجه به اینکه تعدادی حروف به شبکه داده می‌شود که مفهومی برای شبکه ندارد، اینکه هنگام بازسازی کلمات و ساختن شعر حروف به صورت معناداری کنار یکدیگر قرار بگیرند از کاربردهای حافظه است. همچنین در سطح بالاتر اینکه کلمات کنار هم دارای نزدیکی معنایی باشند تا جملات نسبتا معناداری خلق شود از کاربردهای حافظه است.