

مقدمه :

در این پروژه که شامل سه بخش است که در دو قسمت اول با مدل **feed forward** ، شبکه عصبی آموزش دیده و در قسمت سوم با مدل **RNN** در این تمرین تقریباً اکثر قسمت های پروژه شبیه به هم هستند فقط قسمت های مدل و کامپایلش مقداری متفاوت هستند و همچنین به دلیل زیاد بودن دیتا آموزشی و نداشتن سیستم قوی به پیشنهاد مهندس یزدانی مقدار دیتا کاهش داده شده تا بشه مدل را آموزش داد لذا خروجی عددی که در تمرین مشاهده میشود مقدار درستی از ارزیابی کل پروژه نیست ولی کاهش **loss** و **perplexity** در مدل های مختلف تمرین و فرقشون در خروجی کاملاً قابل مشاهده و قابل تحلیل هستند.

خلاصه کار :

در اکثر قسمت های این تمرین و بخش هاش ابتداء دیتا تست و آموزشی لود شده اند و بعد کلمات اضافه یا **stop words** ها از دیتا ها حذف شده اند بعد کل دیتا ها چه تست و چه آموزشی **Tokenize** شده اند تا لغات جدا شوند و مقدار عددی به این لغات اختصاص داده شود بعد با استفاده از **n-gram** و جدا کردن کلمات در **n** گرام های هدف یک **sequence** از دیتا تست و آموزشی ایجاد شده است بعد برای پر کردن جاهای که با **n** گرام پر نشده اند باتابع **pad_sequences** آن قسمت ها پر شده اند و در قسمت بعد گرام اخر به عنوان **y** در نظر گرفته شده است مثلاً اگر ۱ و ۲ و ۳ باشد مقدار ۳ به عنوان **y** در نظر گرفته میشود و دو مقدار دیگر به عنوان ورودی در نظر گرفته می شوند در قسمت بعد این **y** ها به ماتریکس کلاس های باینری تبدیل شده اند تا بشود آنها را در فرایند آموزش استفاده کرد در ادامه برای بخش هایی که نیاز به لایه قبل آموزش دیده است مدل **glove** لود شده است و بعد مدل مربوط به هر بخش ایجاد شده که در قسمت بعد گزارش اشاره می شود بعد در ادامه مدل با **SGD** که مدل بهینه شده **GD** هست بهینه شده و **loss** با **cross entropy** هزینه برداری و بررسی شده است در قسمت ارزیابی از تابعی هم به عنوان **perplexity** استفاده شده که طبق مفهوم ریاضیش پیاده سازی شده و در قسمت آموزش هر بخش از **batch_size=25** و **epochs=100** استفاده کرده است تا فرایند تحلیل بخش های مختلف قابل بررسی باشد بعد فرایند آموزش ، برای تحلیل از چارت مقادیر ارزیابی استفاده شده و بعد این عمل ، مدل با مقادیر تستی ارزیابی شده است

توضیح خط به خط کد - قسمت های تکراری:

در ابتدای قسمت هایی که در کل پروژه استفاده شده و تکراری هستند توضیح داده شده اند بعد در ادامه هر تمرین جدا قسمت های مهم مربوط به تمرین ها توضیح داده شده اند در ابتدای چون در **colab** پروژه ها نوشته شده به **drive** گوگل متصل می شویم و به پوشه فایل های دیتا میرویم



```
from google.colab import drive  
drive.mount('/content/drive')  
%cd /content/drive/MyDrive/NLP/Project3/  
%ls
```

بعد در ادامه فایل های تکست یا دیتا ها آموزشی و تست را با تابع `load_doc` لود میکنیم همونجری که در قسمت خلاصه گفته شد دیتا آموزشی به مقداری که فضای رم اجازه میداد کاهش داده شده و دیتای جدید با عنوان فایل `train_low` استفاده شده

```
# load doc into memory  
def load_doc(filename):  
    # open the file as read only  
    file = open(filename, 'r')  
    # read all text  
    text = file.read()  
    # close the file  
    file.close()  
    return text  
  
train = load_doc('dataset/train_low.txt')  
test = load_doc('dataset/test.txt')
```

در ادامه `stop words` های دیتا ها آموزشی و تست با تابع `remove_stop_words` حذف شدند.

```
def remove_stop_words(value):
    try:
        nltk.data.find('tokenizers/punkt')
    except LookupError:
        nltk.download('punkt')

    stop_words = set(stopwords.words('english'))

    word_tokens = word_tokenize(value)

    filtered_sentence = [word for word in word_tokens if word not in stopwords.words('english')]

    filtered_sentence = []

    for w in word_tokens:
        if w not in stop_words:
            filtered_sentence.append(w)

    return TreebankWordDetokenizer().detokenize(filtered_sentence).replace('\\', '')
```



```
import nltk
nltk.download('stopwords')
train=remove_stop_words(train)
test=remove_stop_words(test)
```

در ادامه دیتا های آموزشی و تست Tokeniz شدند



```
tokenizer_train = Tokenizer()
tokenizer_train.fit_on_texts([train])
encoded_train = tokenizer_train.texts_to_sequences([train])[0]

tokenizer_test = Tokenizer()
tokenizer_test.fit_on_texts([test])
encoded_test = tokenizer_test.texts_to_sequences([test])[0]
```

و سایز کلمات موجود در دیتا ها استخراج شدند



```
vocab_size_train = len(tokenizer_train.word_index) + 1
print('Vocabulary Size: %d' % vocab_size_train)

vocab_size_test = len(tokenizer_test.word_index) + 1
print('Vocabulary Size: %d' % vocab_size_test)
```

در ادامه n-gram ها جدا شدند در زیر یک 5-gram را مشاهده میکنید!



```
#five gram

sequences_train = list()
for i in range(4, len(encoded_train)):
    sequence_train = encoded_train[i-3:i+2]
    sequences_train.append(sequence_train)
print('Total Sequences train: %d' % len(sequences_train))

sequences_test = list()
for i in range(4, len(encoded_test)):
    sequence_test = encoded_test[i-3:i+2]
    sequences_test.append(sequence_test)
print('Total Sequences test: %d' % len(sequences_test))
```

که در قسمت

encoded_train[i-3:i+2]

میشود gram ها را تنظیم کرد و سایزشون را مشخص کرد در این قسمت برای هر دو دیتا این موضوع جداسازی شده است در یک لیست از sequence ها تنظیم شده اند
در ادامه باید قسمت هایی که کم جدا شده اند برای مثال اگر یک 5-gram باشد و یک sequence فقط ۳ مقدار داشته باشه قسمت های دیگه با مقدار ۰ پر میشود تا یک 5-gram بشود (pre یعنی از اول مقادیر)



```
max_length_train = max([len(seq) for seq in sequences_train])
sequences_train = pad_sequences(sequences_train, maxlen=max_length_train, padding='pre')
print('Max Sequences train Length: %d' % max_length_train)

max_length_test = max([len(seq) for seq in sequences_test])
sequences_test = pad_sequences(sequences_test, maxlen=max_length_test, padding='pre')
print('Max Sequences test Length: %d' % max_length_test)
```

در ادامه x و y جدا سازی میشوند برای مثال برای gram-5 به تعداد ۴ مقدار به عنوان x و ۱ عدد باقی مانده y میشود برای هر دو دیتا آموزشی و تستی این عمل انجام میشود



```
sequences_train = np.array(sequences_train)
X_train, y_train = sequences_train[:, :-1], sequences_train[:, -1]

sequences_test = np.array(sequences_test)
X_test, y_test = sequences_test[:, :-1], sequences_test[:, -1]
```

بعد این ارایه y ها را باید به سایز کلمات قسمت آموزشی تبدیل می کنیم تا بعد ارایه تنظیم شود و فرایند آموزش به درستی انجام بشود



```
y_train = to_categorical(y_train, num_classes=vocab_size_train)
y_test = to_categorical(y_test, num_classes=vocab_size_train)
```

در ادامه قسمت ساخت مدل هست که به دلیل تفاوت هاش با تمرین های دیگه در قسمت دیگه برای هر تمرین توضیح مدل داده میشود
برای تحلیل این مدل ازتابع perplexity استفاده میشود



```
def perplexity(y_true, y_pred):
    cross_entropy = losses.categorical_crossentropy(y_true, y_pred)
    perplexity = backend.pow(2.0, cross_entropy)
    return perplexity
```

و با مقادیر که در قسمت خلاصه گفته شد مدل آموزش میبینید



```
metrics = model.fit(X_train, y_train, epochs=100, batch_size=25)
```

بعد این عمل با کتابخانه pandas loss و perplexity مقادیر را میسنجیم



```
result = pd.DataFrame.from_dict(metrics.history)
result.plot.line(secondary_y=["loss", "val_loss"])
```

خروجی این مدل ها در قسمت بعدی در هر تمرین قابل مشاهده است بعد در ادامه مدل با دیتاهاى تستی سنجدیده میشود



```
print(model.evaluate(X_test,y_test))
```

توضیح خط به خط کد - تمرین ها:

بخش اول:

الف) در این قسمت طبق تمرین عمل شد در ابتدا context ورودی ۴ در نظر گرفته شده است که میشود 5 gram

```
#five gram

sequences_train = list()
for i in range(4, len(encoded_train)):
    sequence_train = encoded_train[i-3:i+2]
    sequences_train.append(sequence_train)
print('Total Sequences train: %d' % len(sequences_train))

sequences_test = list()
for i in range(4, len(encoded_test)):
    sequence_test = encoded_test[i-3:i+2]
    sequences_test.append(sequence_test)
print('Total Sequences test: %d' % len(sequences_test))
```

در ادامه مدل glove لود شد و بعد شبکه ایجاد شد

```
# Prepare Glove File
def readGloveFile(gloveFile):
    with open(gloveFile, 'r') as f:
        wordToGlove = {} # map from a token (word) to a Glove embedding vector
        wordToIndex = {} # map from a token to an index
        indexToWord = {} # map from an index to a token

    for line in f:
        record = line.strip().split()
        token = record[0] # take the token (word) from the text line
        wordToGlove[token] = np.array(record[1:], dtype=np.float64)

    tokens = sorted(wordToGlove.keys())
    for idx, tok in enumerate(tokens):
        kerasIdx = idx + 1 # 0 is reserved for masking in Keras (see above)
        wordToIndex[tok] = kerasIdx # associate an index to a token (word)
        indexToWord[kerasIdx] = tok

    return wordToIndex, indexToWord, wordToGlove
```



```
# Create Pretrained Keras Embedding Layer
def createPretrainedEmbeddingLayer(wordToGlove, wordToIndex, isTrainable):
    vocabLen = len(wordToIndex) + 1 # adding 1 to account for masking
    embDim = next(iter(wordToGlove.values())).shape[0] # works with any glove dimensions (e.g. 50)

    embeddingMatrix = np.zeros((vocabLen, embDim)) # initialize with zeros
    for word, index in wordToIndex.items():
        embeddingMatrix[index, :] = wordToGlove[word] # create embedding: word index to Glove word embedding

    embeddingLayer = Embedding(vocabLen, embDim, weights=[embeddingMatrix], input_length=max_length_train-1,
                               trainable=isTrainable)
    return embeddingLayer

# usage
wordToIndex, indexToWord, wordToGlove = readGloveFile("glove.6B.50d.txt")
pretrainedEmbeddingLayer = createPretrainedEmbeddingLayer(wordToGlove, wordToIndex, False)
```

این قسمت لود فایل glove در بخش اول در تمرین های دیگر یکسان است.
در ادامه مدل با فایل glove و به عنوان یک لایه مدل ایجاد میشود و طبق شبکه گفته شده در تمرین ساخته میشود البته دو Flatten برای فلت کردن دیتا ها و یک Dropout برای حذف یک سری از نورون ها برای بهتر کردن شبکه در فرایند ترین در این شبکه استفاده شده است.



```
model = Sequential()
model.add(pretrainedEmbeddingLayer)
model.add(Flatten())
model.add(Dense(35, activation='sigmoid'))
model.add(Dropout(0.1))
model.add(Flatten())
model.add(Dense(vocab_size_train, activation='softmax'))

gd=tf.keras.optimizers.SGD(
    learning_rate=0.02,
    name='SGD'
)

model.compile(loss='categorical_crossentropy', optimizer=gd, metrics=[perplexity])

print(model.summary())
```

و SGD هم با optimizer استفاده شده که یک مدل بهینه شده gradient descent است و loss هم با crossentropy مورد ارزیابی انجام شده و metrics هم از perplexity موردن استفاده قرار گرفته در این تمرین از ۳۵ لایه مخفی استفاده شده است و نرخ یادگیری هم بر روی ۰.۰۲ است و خروجی این تمرین با توجه به دیتا کم به شکل زیر است

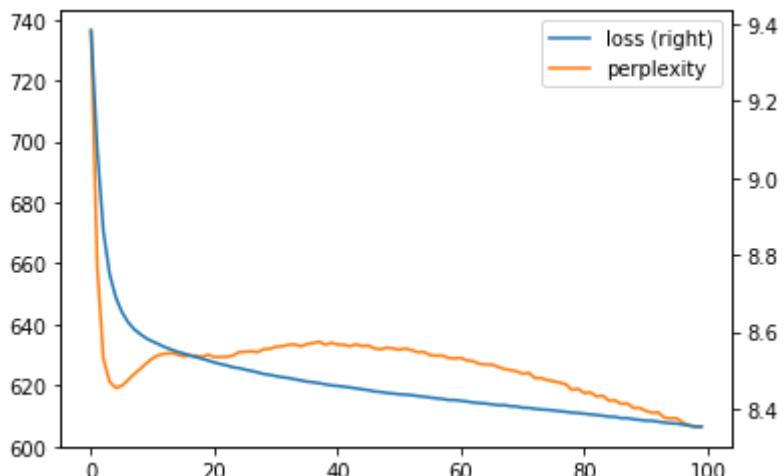


```

Epoch 1/100
2282/2282 [=====] - 22s 9ms/step - loss: 9.4856 - perplexity: 758.3750
Epoch 2/100
2282/2282 [=====] - 23s 10ms/step - loss: 9.1402 - perplexity: 672.2408
Epoch 3/100
2282/2282 [=====] - 22s 10ms/step - loss: 8.8951 - perplexity: 629.2902
Epoch 4/100
2282/2282 [=====] - 26s 11ms/step - loss: 8.7446 - perplexity: 614.8703
Epoch 5/100
2282/2282 [=====] - 20s 9ms/step - loss: 8.6865 - perplexity: 614.9957
Epoch 6/100
2282/2282 [=====] - 21s 9ms/step - loss: 8.6610 - perplexity: 617.3013
Epoch 7/100
2282/2282 [=====] - 20s 9ms/step - loss: 8.6252 - perplexity: 619.0501
.
.
.
Epoch 97/100
2282/2282 [=====] - 20s 9ms/step - loss: 8.3527 - perplexity: 604.8307
Epoch 98/100
2282/2282 [=====] - 20s 9ms/step - loss: 8.3579 - perplexity: 607.4549
Epoch 99/100
2282/2282 [=====] - 20s 9ms/step - loss: 8.3515 - perplexity: 606.2839
Epoch 100/100
2282/2282 [=====] - 20s 9ms/step - loss: 8.3435 - perplexity: 604.8104

```

نمودار خروجی



همینجور که مشاهده میکنید loss به سمت پایین میاید ولی perplexity بعد یک نوسان نه خیلی زیاد باز به سمت پایین میاید و ارزیابی قسمت تست به شکل زیر است



```

386/386 [=====] - 2s 6ms/step - loss: 8.6445 - perplexity: 559.4490
[8.644495010375977, 559.448974609375]

```

د) قسمت context 2

در این قسمت که با context ۲ ورودی است به معنی gram-3 است که ابتدا gram ها را تنظیم میکنیم

```

#Three gram

sequences_train = list()
for i in range(2, len(encoded_train)):
    sequence_train = encoded_train[i-2:i+1]
    sequences_train.append(sequence_train)
print('Total Sequences train: %d' % len(sequences_train))

sequences_test = list()
for i in range(2, len(encoded_test)):
    sequence_test = encoded_test[i-2:i+1]
    sequences_test.append(sequence_test)
print('Total Sequences test: %d' % len(sequences_test))

```

و بعد در قسمت ساخت مدل بعد لود لایه glove مدل را مثل مثال تمرین الف بخش اول ایجاد میکنیم

```

model = Sequential()
model.add(pretrainedEmbeddingLayer)
model.add(Flatten())
model.add(Dense(35, activation='sigmoid'))
model.add(Dropout(0.1))
model.add(Flatten())
model.add(Dense(vocab_size_train, activation='softmax'))

gd=tf.keras.optimizers.SGD(
    learning_rate=0.02,
    name='SGD'
)

model.compile(loss='categorical_crossentropy', optimizer=gd, metrics=[perplexity])

print(model.summary())

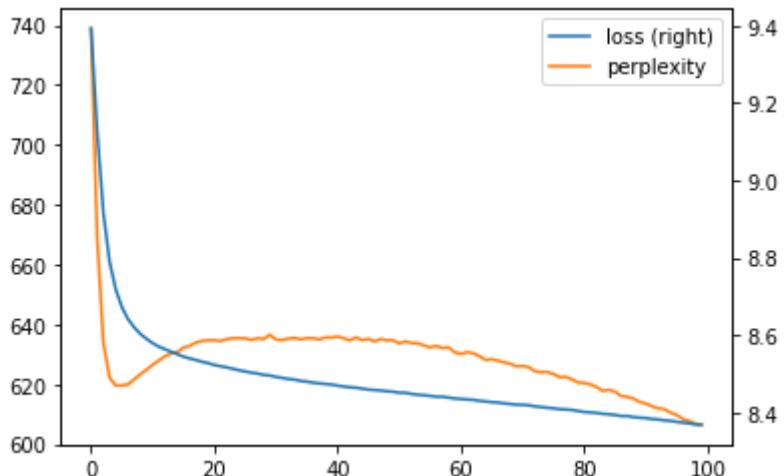
```

و بعد فرایند آموزش خروجی به شکل زیر است



```
Epoch 1/100  
2282/2282 [=====] - 24s 10ms/step - loss: 9.4906 - perplexity: 758.7431  
Epoch 2/100  
2282/2282 [=====] - 24s 10ms/step - loss: 9.1909 - perplexity: 683.0336  
Epoch 3/100  
2282/2282 [=====] - 24s 11ms/step - loss: 8.9577 - perplexity: 637.5202  
Epoch 4/100  
2282/2282 [=====] - 25s 11ms/step - loss: 8.8187 - perplexity: 623.1413  
Epoch 5/100  
2282/2282 [=====] - 26s 11ms/step - loss: 8.7255 - perplexity: 618.2575  
. .  
Epoch 98/100  
2282/2282 [=====] - 28s 12ms/step - loss: 8.3575 - perplexity: 602.4513  
Epoch 99/100  
2282/2282 [=====] - 27s 12ms/step - loss: 8.3733 - perplexity: 608.8522  
Epoch 100/100  
2282/2282 [=====] - 29s 12ms/step - loss: 8.3574 - perplexity: 602.9591
```

و نمودار خروجی



و همین جور که مشاهده میکنید تقریبا مثل تمرین الف است ولی نوسانش کمتر است و ارزیابی قسمت تست که نسبت به تمرین الف بهتر شده است



```
386/386 [=====] - 3s 8ms/step - loss: 8.6328 - perplexity: 556.2695  
[8.632795333862305, 556.26953125]
```

د) قسمت 3 context

در این قسمت مدل و بقیه قسمت هاش شبیه به context 2 با این تفاوت که 4 gram است و قسمت جداساز gram ها به شکل زیر است

```
#four gram

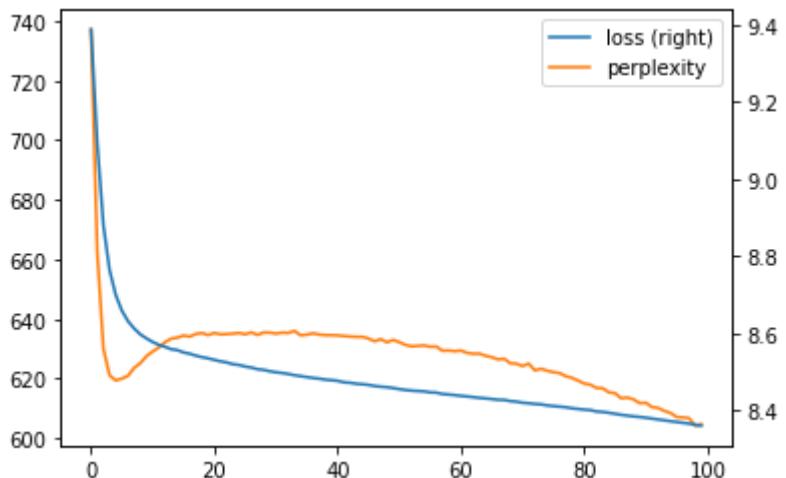
sequences_train = list()
for i in range(3, len(encoded_train)):
    sequence_train = encoded_train[i-2:i+2]
    sequences_train.append(sequence_train)
print('Total Sequences train: %d' % len(sequences_train))

sequences_test = list()
for i in range(3, len(encoded_test)):
    sequence_test = encoded_test[i-2:i+2]
    sequences_test.append(sequence_test)
print('Total Sequences test: %d' % len(sequences_test))
```

و خروجی

```
Epoch 1/100
2282/2282 [=====] - 8s 3ms/step - loss: 9.4891 - perplexity: 758.2478
Epoch 2/100
2282/2282 [=====] - 6s 3ms/step - loss: 9.1566 - perplexity: 676.6059
Epoch 3/100
2282/2282 [=====] - 6s 3ms/step - loss: 8.9326 - perplexity: 634.2456
Epoch 4/100
2282/2282 [=====] - 6s 3ms/step - loss: 8.7867 - perplexity: 621.4102
Epoch 5/100
2282/2282 [=====] - 6s 3ms/step - loss: 8.7094 - perplexity: 617.1010
.
.
.
Epoch 96/100
2282/2282 [=====] - 6s 3ms/step - loss: 8.3674 - perplexity: 605.9908
Epoch 97/100
2282/2282 [=====] - 6s 3ms/step - loss: 8.3687 - perplexity: 606.8677
Epoch 98/100
2282/2282 [=====] - 6s 3ms/step - loss: 8.3483 - perplexity: 599.0531
Epoch 99/100
2282/2282 [=====] - 6s 3ms/step - loss: 8.3678 - perplexity: 607.1292
Epoch 100/100
2282/2282 [=====] - 6s 3ms/step - loss: 8.3618 - perplexity: 603.7863
```

و نمودار



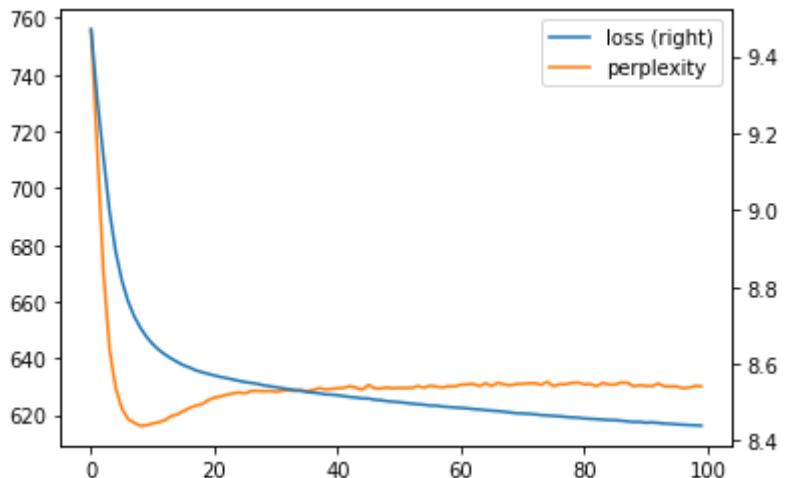
هست که نسبت به ۲ Context یک مقدار بدتر شده نوسانش نسبت به الف کمتره ولی نوسان بالا تر از ۲ Context است و در قسمت ارزیابی دیتای تست هم بهتر از سوال الف است و بدتر از تمرین ب است

```
● ● ●
386/386 [=====] - 1s 3ms/step - loss: 8.6362 - perplexity: 557.2966
[8.636173248291016, 557.296630859375]
```

۵) در این قسمت با فرض Context ۴ انجام شده در بخش اول مدل با نرخ یادگیری ۰.۰۱ انجام شده که خروجی به شکل زیر است

```
● ● ●
Epoch 1/100
2282/2282 [=====] - 9s 3ms/step - loss: 9.5483 - perplexity: 769.9697
Epoch 2/100
2282/2282 [=====] - 7s 3ms/step - loss: 9.3269 - perplexity: 724.8901
Epoch 3/100
2282/2282 [=====] - 7s 3ms/step - loss: 9.1730 - perplexity: 679.0164
Epoch 4/100
2282/2282 [=====] - 7s 3ms/step - loss: 9.0258 - perplexity: 646.9346
Epoch 5/100
2282/2282 [=====] - 7s 3ms/step - loss: 8.9092 - perplexity: 629.8713
Epoch 6/100
2282/2282 [=====] - 7s 3ms/step - loss: 8.8281 - perplexity: 621.2082
.
.
.
Epoch 95/100
2282/2282 [=====] - 9s 4ms/step - loss: 8.4376 - perplexity: 627.5877
Epoch 96/100
2282/2282 [=====] - 8s 3ms/step - loss: 8.4360 - perplexity: 628.5787
Epoch 97/100
2282/2282 [=====] - 7s 3ms/step - loss: 8.4368 - perplexity: 626.0225
Epoch 98/100
2282/2282 [=====] - 7s 3ms/step - loss: 8.4425 - perplexity: 632.2542
Epoch 99/100
2282/2282 [=====] - 7s 3ms/step - loss: 8.4300 - perplexity: 623.4292
Epoch 100/100
2282/2282 [=====] - 7s 3ms/step - loss: 8.4471 - perplexity: 630.0646
```

که در نمودار



که آنچه در نمودار مشخص است نوسان perplexity خیلی کم نسبت به تمرین ها قبل است و در ابتداء نوسان پایین تر از loss است ولی در ادامه نوسان بالا میرود ولی loss مثل تمرین های قبل روند نزولی دارد

در قسمت ارزیابی دیتای تست خروجی تقریباً مثل تمرین های قبل است

```
● ● ●
-----evaluate-----
386/386 [=====] - 1s 3ms/step - loss: 8.6315 - perplexity: 558.8427
[8.631499290466309, 558.8427124023438]
```

در نرخ یادگیری 0.03 خروجی به شکل زیر است

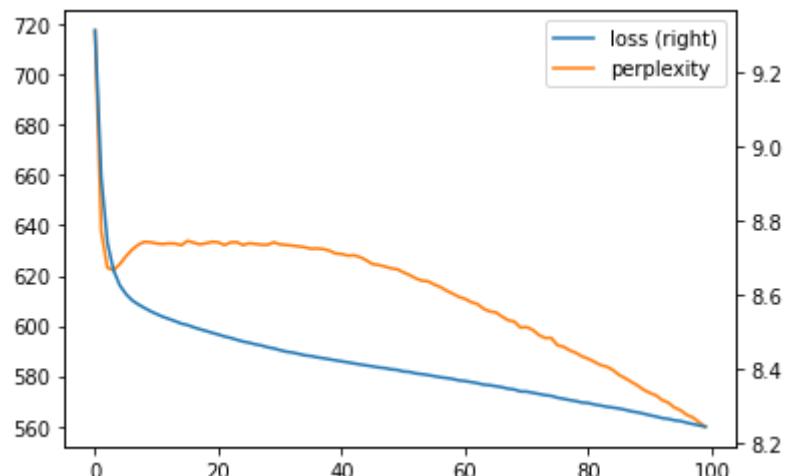
```

● ● ●

Epoch 1/100
2282/2282 [=====] - 8s 3ms/step - loss: 9.4429 - perplexity: 748.4941
Epoch 2/100
2282/2282 [=====] - 7s 3ms/step - loss: 8.9737 - perplexity: 642.1187
Epoch 3/100
2282/2282 [=====] - 7s 3ms/step - loss: 8.7495 - perplexity: 618.7335
Epoch 4/100
2282/2282 [=====] - 7s 3ms/step - loss: 8.6639 - perplexity: 618.5576
Epoch 5/100
2282/2282 [=====] - 7s 3ms/step - loss: 8.6249 - perplexity: 620.7380
Epoch 6/100
2282/2282 [=====] - 7s 3ms/step - loss: 8.6060 - perplexity: 624.8040
Epoch 7/100
2282/2282 [=====] - 7s 3ms/step - loss: 8.5946 - perplexity: 628.7011
.
.
.
Epoch 94/100
2282/2282 [=====] - 7s 3ms/step - loss: 8.2679 - perplexity: 567.1425
Epoch 95/100
2282/2282 [=====] - 7s 3ms/step - loss: 8.2653 - perplexity: 566.3267
Epoch 96/100
2282/2282 [=====] - 7s 3ms/step - loss: 8.2455 - perplexity: 563.0526
Epoch 97/100
2282/2282 [=====] - 7s 3ms/step - loss: 8.2608 - perplexity: 562.1390
Epoch 98/100
2282/2282 [=====] - 8s 3ms/step - loss: 8.2430 - perplexity: 561.4379
Epoch 99/100
2282/2282 [=====] - 7s 3ms/step - loss: 8.2405 - perplexity: 558.7074
Epoch 100/100
2282/2282 [=====] - 7s 3ms/step - loss: 8.2515 - perplexity: 561.6334

```

که در نمودار خروجی



در این نمودار مشخص است که این نرخ بسیار با خروجی های تمرین های دیگر تفاوت دارد و نوسان خیلی زودتر از بقیه تمرین ها شروع شده و تقریبا به سمت نزول هدایت شده loss هم کماکان به شکل نزولی و مثل بقیه است



```
-----evaluate-----
386/386 [=====] - 1s 3ms/step - loss: 8.6681 - perplexity: 570.4092
[8.66810417175293, 570.4092407226562]
```

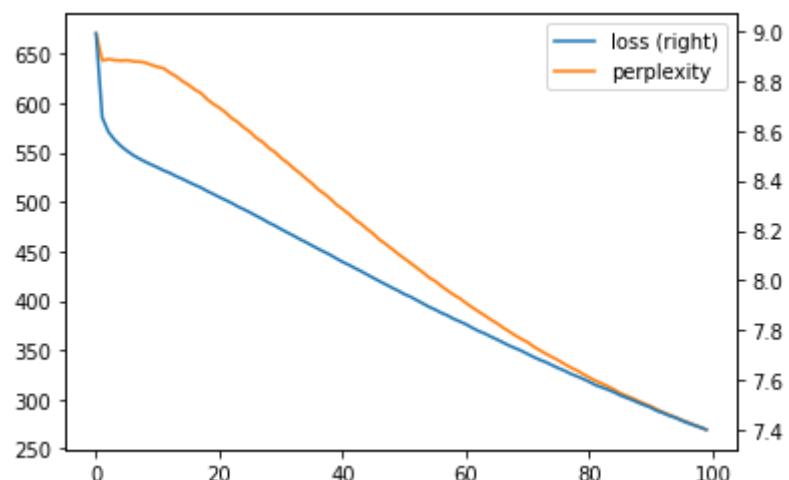
در ارزیابی و در مقایسه با نرخ یادگیری قبلی خروجی perplexity بهتری از خود نشان میدهد ولی loss بیشتری نسبت به قبلی دارد!

در نرخ یادگیری 0.1 خروجی به شکل زیر است



```
Epoch 1/100
2282/2282 [=====] - 8s 3ms/step - loss: 9.2226 - perplexity: 704.7536
Epoch 2/100
2282/2282 [=====] - 7s 3ms/step - loss: 8.6640 - perplexity: 632.5827
Epoch 3/100
2282/2282 [=====] - 7s 3ms/step - loss: 8.5817 - perplexity: 632.8902
Epoch 4/100
2282/2282 [=====] - 7s 3ms/step - loss: 8.5555 - perplexity: 632.2454
Epoch 5/100
2282/2282 [=====] - 7s 3ms/step - loss: 8.5219 - perplexity: 632.6564
.
.
.
Epoch 96/100
2282/2282 [=====] - 7s 3ms/step - loss: 7.4221 - perplexity: 276.2366
Epoch 97/100
2282/2282 [=====] - 7s 3ms/step - loss: 7.4167 - perplexity: 274.9374
Epoch 98/100
2282/2282 [=====] - 7s 3ms/step - loss: 7.4099 - perplexity: 273.7755
Epoch 99/100
2282/2282 [=====] - 7s 3ms/step - loss: 7.4015 - perplexity: 270.4181
Epoch 100/100
2282/2282 [=====] - 7s 3ms/step - loss: 7.3811 - perplexity: 265.7801
```

و نمودار



همینجا که قابل مشاهده است تفاوت بسیار زیاد تر از بقیه تمرین ها است و perplexity و loss خیلی زودتر از بقیه شبیب پیدا کردن نکته جالب در این قسمت این است که perplexity بسیار پایین آمده ولی loss خیلی بالاتر! در ارزیابی مقادیر تستی نتیجه بسیار بد تر از بقیه است!



```
-----evaluate-----
386/386 [=====] - 1s 3ms/step - loss: 8.9400 - perplexity: 762.4026
[8.940037727355957, 762.4026489257812]
```

ز) در این تمرین تمرکز بر روی لایه ها است نرخ یادگیری بر روی **0.02** تنظیم شده است و لایه مخفی با مقادیر تمرین تست شده اند در قسمت اول با لایه مخفی **50** ، مدل تست شده است



```
model = Sequential()
model.add(pretrainedEmbeddingLayer)
model.add(Flatten())
model.add(Dense(50, activation='sigmoid'))
model.add(Dropout(0.1))
model.add(Flatten())
model.add(Dense(vocab_size_train, activation='softmax'))

gd=tf.keras.optimizers.SGD(
    learning_rate=0.02,
    name='SGD'
)

model.compile(loss='categorical_crossentropy', optimizer=gd, metrics=[perplexity])
```

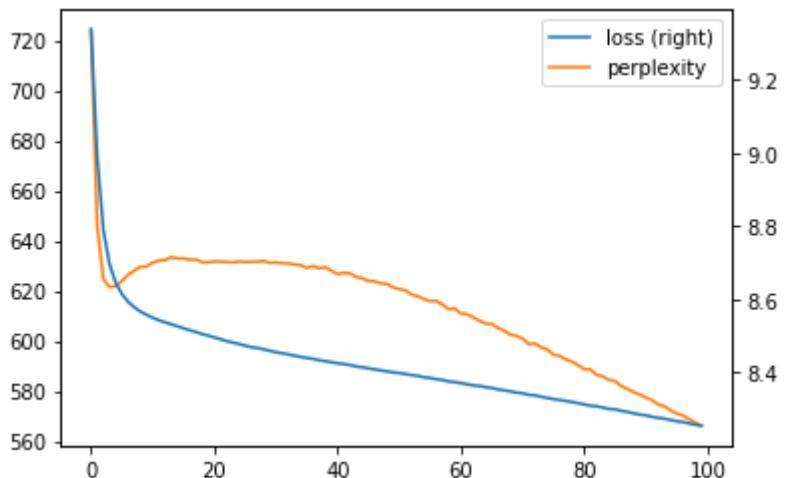
که خروجی به شکل زیر است



```
Epoch 1/100
2282/2282 [=====] - 29s 12ms/step - loss: 9.4518 - perplexity: 750.6510
Epoch 2/100
2282/2282 [=====] - 28s 12ms/step - loss: 9.0629 - perplexity: 654.7058
Epoch 3/100
2282/2282 [=====] - 29s 13ms/step - loss: 8.8114 - perplexity: 623.5755
Epoch 4/100
2282/2282 [=====] - 28s 12ms/step - loss: 8.7068 - perplexity: 619.3848
.
.
.

Epoch 95/100
2282/2282 [=====] - 29s 13ms/step - loss: 8.2757 - perplexity: 573.9829
Epoch 96/100
2282/2282 [=====] - 30s 13ms/step - loss: 8.2619 - perplexity: 570.0066
Epoch 97/100
2282/2282 [=====] - 30s 13ms/step - loss: 8.2640 - perplexity: 566.9009
Epoch 98/100
2282/2282 [=====] - 31s 13ms/step - loss: 8.2494 - perplexity: 566.7756
Epoch 99/100
2282/2282 [=====] - 30s 13ms/step - loss: 8.2538 - perplexity: 564.6380
Epoch 100/100
2282/2282 [=====] - 31s 13ms/step - loss: 8.2549 - perplexity: 565.2275
```

و نمودار به شکل زیر است



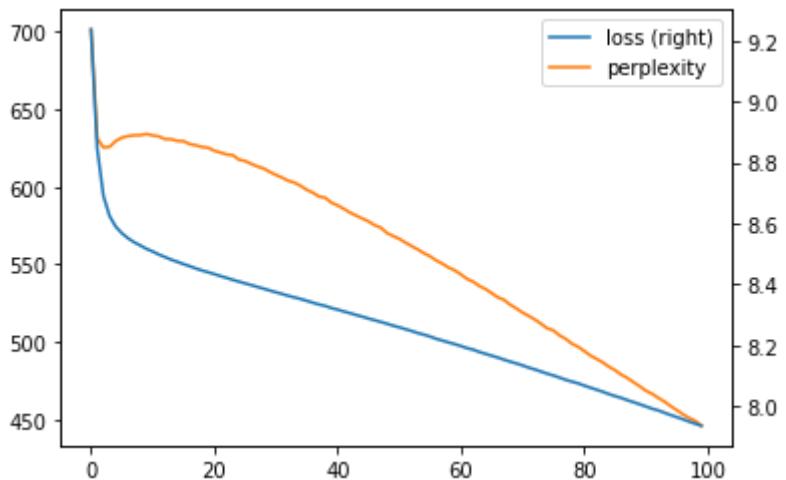
همانطور که قابل مشاهده است این تمرین هم تقریبا به شکل تمرین های اول شب ملاجمی دارد و perplexity با یک نوسان به شیب پایین نزول میکند ! در قسمت ارزیابی تفاوت زیاد نسب به بخش های اول قابل مشاهده نیست و تقریبا به همان شکل تمارین اول است

```
● ● ●
-----evaluate-----
386/386 [=====] - 3s 7ms/step - loss: 8.6655 - perplexity: 567.8088
[8.665534019470215, 567.8087768554688]
```

در لایه مخفی ۱۰۰ تایی خروجی به شکل زیر است

```
● ● ●
Epoch 1/100
2282/2282 [=====] - 41s 18ms/step - loss: 9.3793 - perplexity: 734.9800
Epoch 2/100
2282/2282 [=====] - 41s 18ms/step - loss: 8.9087 - perplexity: 633.9142
Epoch 3/100
2282/2282 [=====] - 41s 18ms/step - loss: 8.6988 - perplexity: 620.5102
Epoch 4/100
2282/2282 [=====] - 41s 18ms/step - loss: 8.6380 - perplexity: 622.8795
Epoch 5/100
2282/2282 [=====] - 42s 18ms/step - loss: 8.5851 - perplexity: 623.4719
Epoch 6/100
2282/2282 [=====] - 42s 18ms/step - loss: 8.5654 - perplexity: 625.3350
.
.
.
Epoch 94/100
2282/2282 [=====] - 42s 18ms/step - loss: 7.9817 - perplexity: 458.9256
Epoch 95/100
2282/2282 [=====] - 43s 19ms/step - loss: 7.9648 - perplexity: 456.4293
Epoch 96/100
2282/2282 [=====] - 43s 19ms/step - loss: 7.9665 - perplexity: 456.2294
Epoch 97/100
2282/2282 [=====] - 41s 18ms/step - loss: 7.9642 - perplexity: 455.2181
Epoch 98/100
2282/2282 [=====] - 41s 18ms/step - loss: 7.9469 - perplexity: 451.2411
Epoch 99/100
2282/2282 [=====] - 42s 18ms/step - loss: 7.9200 - perplexity: 442.9793
Epoch 100/100
2282/2282 [=====] - 41s 18ms/step - loss: 7.9312 - perplexity: 444.7544
```

و نمودار



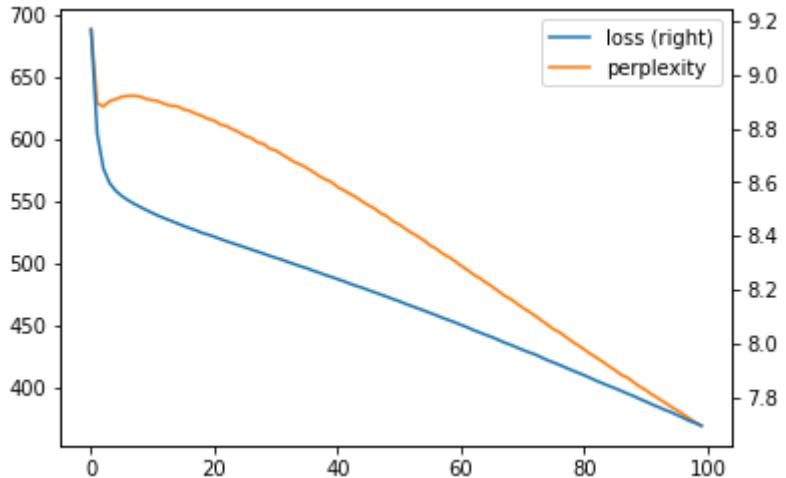
به شکل زیر است که مثل تمرین نرخ یادگیری 0.01 در ابتدا شبیه زیاد پیدا کرده و بعد مقادیر نزول کردن و در قسمت ارزیابی هم نسبت به تمرین قبل خروجی بدی داده است

```
● ● ●
-----evaluate-----
386/386 [=====] - 3s 8ms/step - loss: 8.7122 - perplexity: 591.7875
[8.712240219116211, 591.7875366210938]
```

در لایه مخفی ۱۵۰ تایی خروجی به شکل زیر است

```
● ● ●
Epoch 1/100
2282/2282 [=====] - 56s 24ms/step - loss: 9.3382 - perplexity: 725.0545
Epoch 2/100
2282/2282 [=====] - 56s 24ms/step - loss: 8.8318 - perplexity: 627.6342
Epoch 3/100
2282/2282 [=====] - 55s 24ms/step - loss: 8.6700 - perplexity: 622.5616
Epoch 4/100
2282/2282 [=====] - 56s 24ms/step - loss: 8.6034 - perplexity: 624.8517
Epoch 5/100
2282/2282 [=====] - 56s 24ms/step - loss: 8.5698 - perplexity: 626.9414
Epoch 6/100
2282/2282 [=====] - 55s 24ms/step - loss: 8.5430 - perplexity: 627.5300
.
.
.
Epoch 94/100
2282/2282 [=====] - 55s 24ms/step - loss: 7.7386 - perplexity: 383.4438
Epoch 95/100
2282/2282 [=====] - 54s 24ms/step - loss: 7.7386 - perplexity: 382.1866
Epoch 96/100
2282/2282 [=====] - 54s 24ms/step - loss: 7.7246 - perplexity: 378.7454
Epoch 97/100
2282/2282 [=====] - 55s 24ms/step - loss: 7.7091 - perplexity: 375.3159
Epoch 98/100
2282/2282 [=====] - 55s 24ms/step - loss: 7.6968 - perplexity: 372.9526
Epoch 99/100
2282/2282 [=====] - 54s 24ms/step - loss: 7.6891 - perplexity: 368.4361
Epoch 100/100
2282/2282 [=====] - 54s 24ms/step - loss: 7.6969 - perplexity: 368.9561
```

و نمودار به شکل



که یک شب تندی در ابتدا تجربه کرده و روند نزول پیش گرفته است در قسمت ارزیابی loss نسب به بقیه خروجی خوبی نگرفته

```
● ● ●
-----evaluate-----
386/386 [=====] - 4s 11ms/step - loss: 8.7611 - perplexity: 616.1201
[8.76106071472168, 616.1201171875]
```

در این تمرین به نظر میرسه هر چه لایه ها زیاد تر میشن خروجی تست بدتری داریم ولی در فرایند ترین بر عکس خروجی بهتری دارن.

بخش دوم:

در این بخش بجای استفاده از لایه glove از لایه Embedding ۵۰ بعدی استفاده شده است

```
● ● ●

model = Sequential()
model.add(Embedding(vocab_size_train, 50, name="embedding", input_length=max_length_train-1))
model.add(Flatten())
model.add(Dense(35, activation='sigmoid'))
model.add(Dropout(0.1))
model.add(Flatten())
model.add(Dense(vocab_size_train, activation='softmax'))

gd=tf.keras.optimizers.SGD(
    learning_rate=0.02,
    name='SGD'
)

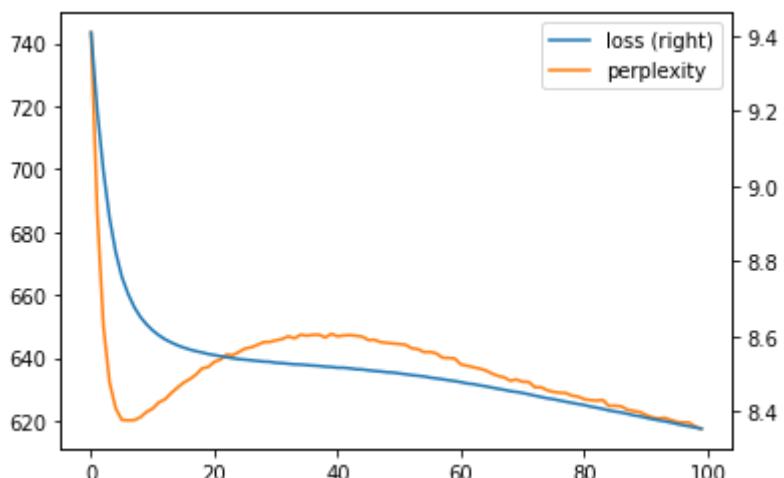
model.compile(loss='categorical_crossentropy', optimizer=gd, metrics=[perplexity])
```

که خروجی به شکل زیر است



```
Epoch 1/100
2282/2282 [=====] - 9s 3ms/step - loss: 9.5020 - perplexity: 761.5888
Epoch 2/100
2282/2282 [=====] - 7s 3ms/step - loss: 9.2321 - perplexity: 696.6583
Epoch 3/100
2282/2282 [=====] - 7s 3ms/step - loss: 9.0745 - perplexity: 656.2572
Epoch 4/100
2282/2282 [=====] - 7s 3ms/step - loss: 8.9477 - perplexity: 635.4925
.
.
.
Epoch 96/100
2282/2282 [=====] - 7s 3ms/step - loss: 8.3656 - perplexity: 618.6832
Epoch 97/100
2282/2282 [=====] - 7s 3ms/step - loss: 8.3648 - perplexity: 619.2824
Epoch 98/100
2282/2282 [=====] - 7s 3ms/step - loss: 8.3452 - perplexity: 616.3122
Epoch 99/100
2282/2282 [=====] - 7s 3ms/step - loss: 8.3487 - perplexity: 619.5077
Epoch 100/100
2282/2282 [=====] - 7s 3ms/step - loss: 8.3334 - perplexity: 611.0101
```

نمودار و



همانطور که قابل مشاهده است در ابتدا هر دو مقدار با شیب خوبی به پایین نزول کردن و لیکن perplexity یک نوسان کم داشت که سریع برگشت و خودشو به loss رساند هم با همان شیب مثل تمرین های قبل به پایین نزول کرد در قسمت ارزیابی مقادیر تستی خروجی متوسط داشت و مثل بقیه تمرین ها بود



```
386/386 [=====] - 1s 3ms/step - loss: 8.6329 - perplexity: 560.2018
[8.632889747619629, 560.2017822265625]
```

: بخش سوم

در این بخش شبکه به شبکه RNN تبدیل شده است و با مقادیر بخش اول تمرین تست و ارزیابی شده است

الف) در این قسمت ۴ context قرار دادیم و نورون های لایه مخفی ۳۵ نورون در نظر گرفتیم و نرخ یادگیری را ۰.۰۲ در نظر گرفتیم

```
● ● ●

model = Sequential()
model.add(pretrainedEmbeddingLayer)
model.add(LSTM(35))
model.add(Dropout(0.5))
model.add(Dense(vocab_size_train, activation='softmax'))

gd=tf.keras.optimizers.SGD(
    learning_rate=0.02,
    name='SGD'
)

model.compile(loss='categorical_crossentropy', optimizer=gd, metrics=[perplexity])
```

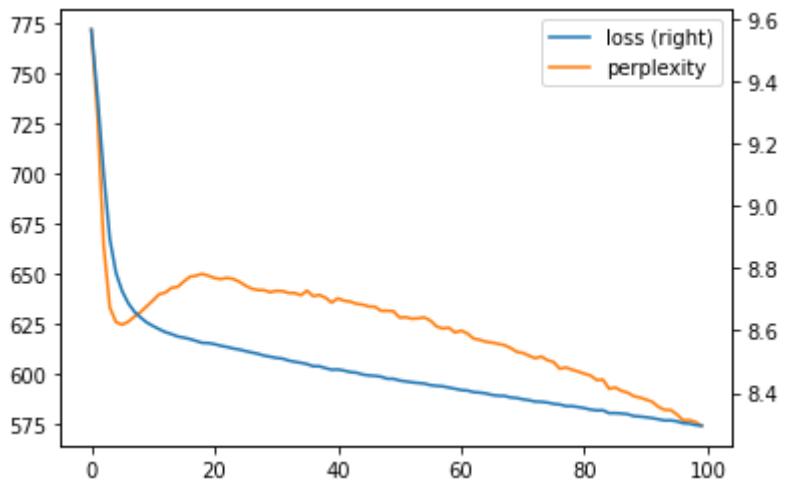
در قسمت لایه های مخفی یک لایه LSTM با مقدار ۳۵ گذاشتیم و شبکه ما با این لایه ایجاد شده است

در خروجی این تمرین

```
● ● ●

Epoch 1/100
2282/2282 [=====] - 15s 4ms/step - loss: 9.6053 - perplexity: 783.2290
Epoch 2/100
2282/2282 [=====] - 8s 4ms/step - loss: 9.3854 - perplexity: 739.6144
Epoch 3/100
2282/2282 [=====] - 8s 4ms/step - loss: 9.1687 - perplexity: 677.3666
Epoch 4/100
2282/2282 [=====] - 8s 4ms/step - loss: 8.9292 - perplexity: 634.4647
Epoch 5/100
2282/2282 [=====] - 8s 4ms/step - loss: 8.7902 - perplexity: 623.8430
.
.
.
Epoch 96/100
2282/2282 [=====] - 8s 4ms/step - loss: 8.2964 - perplexity: 577.6936
Epoch 97/100
2282/2282 [=====] - 8s 4ms/step - loss: 8.3040 - perplexity: 574.0648
Epoch 98/100
2282/2282 [=====] - 8s 4ms/step - loss: 8.2835 - perplexity: 574.2846
Epoch 99/100
2282/2282 [=====] - 8s 4ms/step - loss: 8.2895 - perplexity: 572.6993
Epoch 100/100
2282/2282 [=====] - 8s 4ms/step - loss: 8.2812 - perplexity: 570.8740
```

مشاهده میکنید و نمودار به شکل زیر است



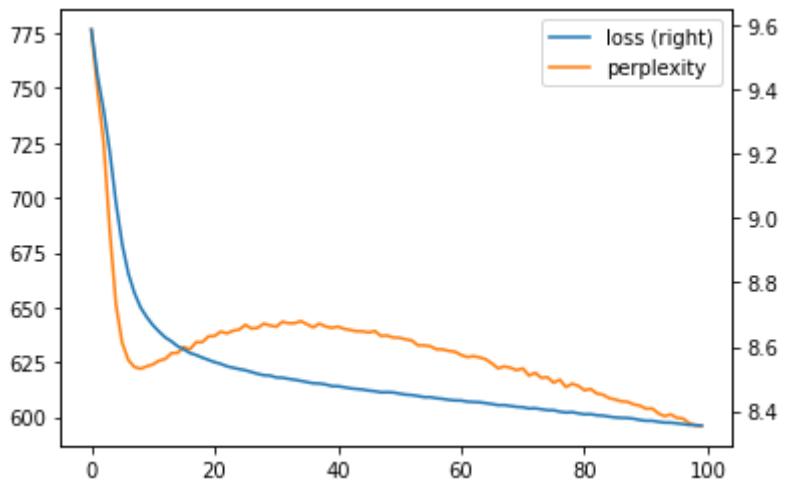
که آنچه قابل مشاهده است یک نوسان زیاد در Epoch های دوربر ۲۰ مشاهده میشود که بعد این نوسان کاهش پیدا کرده در قسمت ارزیابی خروجی نسبتاً خوبی دیده میشود

```
● ● ●
print(model.evaluate(X_test,y_test))
386/386 [=====] - 2s 3ms/step - loss: 8.6751 - perplexity: 567.1617
[8.675086975097656, 567.1617431640625]
```

د) در این قسمت context 2 در نظر گرفته شده و با نرخ یادگیری 0.02 مدل آموزش دیده است که خروجی به شکل زیر است

```
● ● ●
Epoch 1/100
2282/2282 [=====] - 16s 4ms/step - loss: 9.6129 - perplexity: 785.4277
Epoch 2/100
2282/2282 [=====] - 9s 4ms/step - loss: 9.4825 - perplexity: 753.0087
Epoch 3/100
2282/2282 [=====] - 9s 4ms/step - loss: 9.3643 - perplexity: 734.0666
Epoch 4/100
2282/2282 [=====] - 9s 4ms/step - loss: 9.2480 - perplexity: 694.7899
.
.
.
Epoch 97/100
2282/2282 [=====] - 9s 4ms/step - loss: 8.3512 - perplexity: 597.8938
Epoch 98/100
2282/2282 [=====] - 9s 4ms/step - loss: 8.3557 - perplexity: 593.9278
Epoch 99/100
2282/2282 [=====] - 9s 4ms/step - loss: 8.3504 - perplexity: 593.1225
Epoch 100/100
2282/2282 [=====] - 9s 4ms/step - loss: 8.3428 - perplexity: 592.4467
```

و نمودار



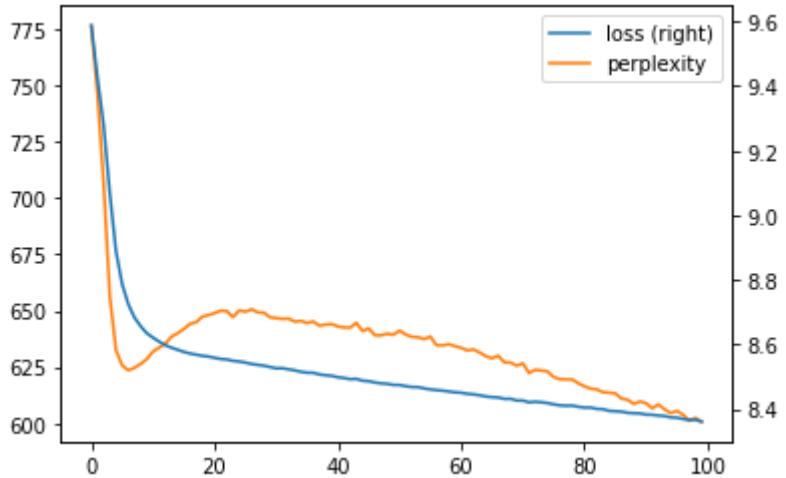
همینطور که مشاهده می شود نوسان تقریبا کمی موجود است در ارزیابی خروجی خوبی مشاهده میشود نسبت به تمرین الف

```
● ● ●
386/386 [=====] - 2s 3ms/step - loss: 8.6477 - perplexity: 557.8184
[8.647721290588379, 557.8184204101562]
```

در شکل زیر است

```
● ● ●
Epoch 1/100
2282/2282 [=====] - 18s 4ms/step - loss: 9.6136 - perplexity: 785.7071
Epoch 2/100
2282/2282 [=====] - 10s 4ms/step - loss: 9.4639 - perplexity: 750.2536
Epoch 3/100
2282/2282 [=====] - 10s 4ms/step - loss: 9.3179 - perplexity: 719.3484
Epoch 4/100
2282/2282 [=====] - 10s 4ms/step - loss: 9.1140 - perplexity: 664.3815
.
.
.
Epoch 96/100
2282/2282 [=====] - 10s 5ms/step - loss: 8.3562 - perplexity: 600.2907
Epoch 97/100
2282/2282 [=====] - 11s 5ms/step - loss: 8.3655 - perplexity: 598.9091
Epoch 98/100
2282/2282 [=====] - 11s 5ms/step - loss: 8.3595 - perplexity: 599.6828
Epoch 99/100
2282/2282 [=====] - 10s 5ms/step - loss: 8.3579 - perplexity: 598.8699
Epoch 100/100
2282/2282 [=====] - 10s 5ms/step - loss: 8.3592 - perplexity: 599.5196
```

و نمودار



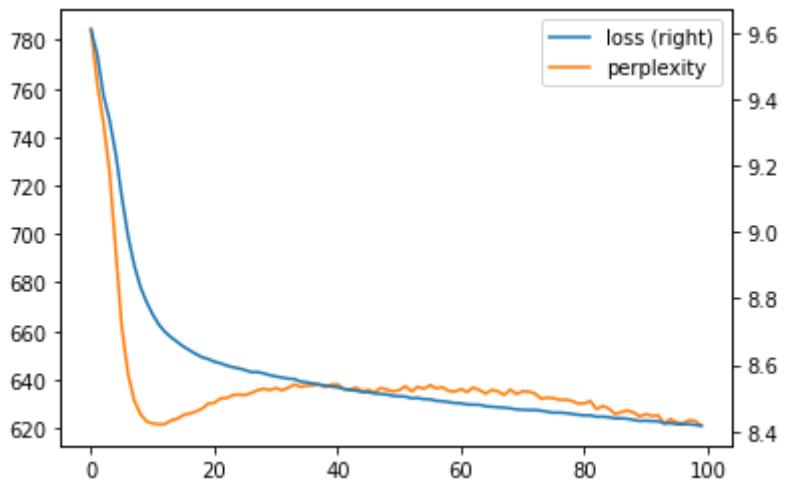
در این تمرین نوسان خوبی وجود دارد و در ارزیابی خروجی بهتری نسبت به دو تمرین قبل مشاهده میشود.

```
● ● ●
386/386 [=====] - 2s 3ms/step - loss: 8.6400 - perplexity: 556.1956
[8.639974594116211, 556.1956176757812]
```

۵) در این قسمت مدل با نرخ یادگیری های مختلف بخش اول تست شده است
در نرخ یادگیری 0.01 خروجی به شکل زیر است

```
● ● ●
Epoch 1/100
2282/2282 [=====] - 44s 18ms/step - loss: 9.6238 - perplexity: 789.7085
Epoch 2/100
2282/2282 [=====] - 39s 17ms/step - loss: 9.5609 - perplexity: 767.6199
Epoch 3/100
2282/2282 [=====] - 40s 17ms/step - loss: 9.4411 - perplexity: 748.3632
Epoch 4/100
2282/2282 [=====] - 43s 19ms/step - loss: 9.3554 - perplexity: 731.9715
Epoch 5/100
2282/2282 [=====] - 41s 18ms/step - loss: 9.2714 - perplexity: 701.8260
.
.
.
Epoch 96/100
2282/2282 [=====] - 38s 17ms/step - loss: 8.4344 - perplexity: 626.6828
Epoch 97/100
2282/2282 [=====] - 38s 17ms/step - loss: 8.4368 - perplexity: 627.8666
Epoch 98/100
2282/2282 [=====] - 37s 16ms/step - loss: 8.4217 - perplexity: 623.8033
Epoch 99/100
2282/2282 [=====] - 39s 17ms/step - loss: 8.4185 - perplexity: 621.5299
Epoch 100/100
2282/2282 [=====] - 38s 17ms/step - loss: 8.4134 - perplexity: 617.8134
```

و نمودار



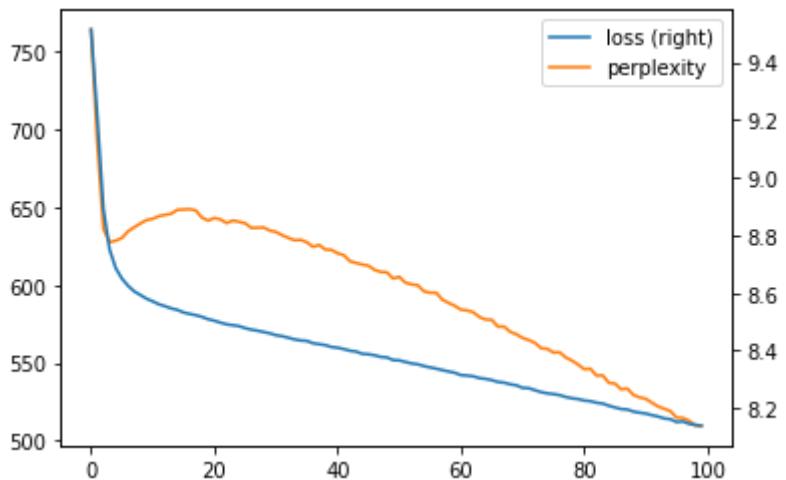
همینجوری که مشاهده میکنید یک سری نوسان های کم در این نمودار و خروجی قابل مشاهده است
در Epoch های بین ۶۰ تا ۱۰۰ نوسان های ریز زیاد است ولی رو به کاهش است
در ارزیابی دیتای تست خروجی متوسطی مثل بقیه تمرین ها مشاهده میشود

```
● ● ●
386/386 [=====] - 5s 11ms/step - loss: 8.6492 - perplexity: 560.7870
[8.649179458618164, 560.7869873046875]
```

در نرخ یادگیری 0.03 خروجی به شکل زیر است

```
● ● ●
Epoch 1/100
2282/2282 [=====] - 42s 17ms/step - loss: 9.5852 - perplexity: 778.5539
Epoch 2/100
2282/2282 [=====] - 40s 17ms/step - loss: 9.3032 - perplexity: 715.5202
Epoch 3/100
2282/2282 [=====] - 41s 18ms/step - loss: 8.9546 - perplexity: 640.3489
Epoch 4/100
2282/2282 [=====] - 42s 18ms/step - loss: 8.7683 - perplexity: 624.7229
.
.
.
Epoch 96/100
2282/2282 [=====] - 37s 16ms/step - loss: 8.1348 - perplexity: 511.9456
Epoch 97/100
2282/2282 [=====] - 36s 16ms/step - loss: 8.1493 - perplexity: 512.6515
Epoch 98/100
2282/2282 [=====] - 35s 15ms/step - loss: 8.1373 - perplexity: 511.8324
Epoch 99/100
2282/2282 [=====] - 35s 15ms/step - loss: 8.1380 - perplexity: 506.5578
Epoch 100/100
2282/2282 [=====] - 37s 16ms/step - loss: 8.1224 - perplexity: 505.1826
```

و نمودار



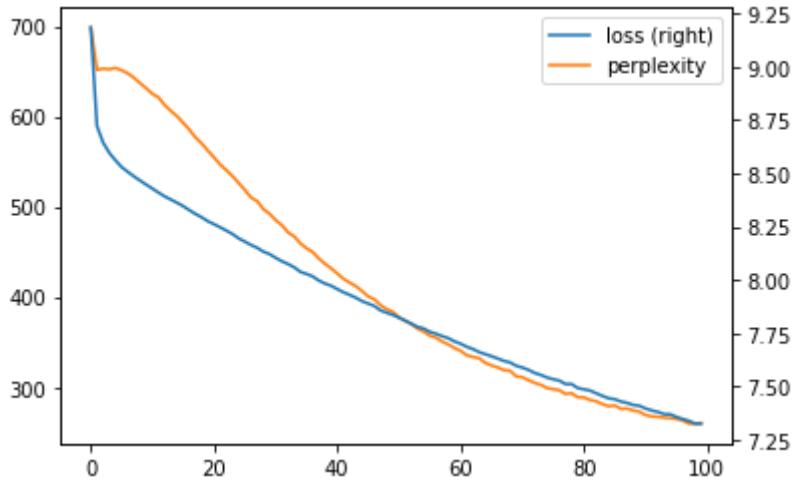
این نرخ یادگیری و نمودارش تفاوت زیادی با نرخ قبلی دارد شیبیش بعد یک نوسان سریع به پایین آمده در ارزیابی خروجی نسبت به سوال قبل بدتر شده مقادیر زیاد تر شده!

```
● ● ●
-----evaluate-----
386/386 [=====] - 5s 10ms/step - loss: 8.7340 - perplexity: 602.2679
[8.734025955200195, 602.2678833007812]
```

در نرخ یادگیری 0.1 خروجی به شکل زیر است

```
● ● ●
Epoch 1/100
2282/2282 [=====] - 41s 17ms/step - loss: 9.4132 - perplexity: 741.9571
Epoch 2/100
2282/2282 [=====] - 37s 16ms/step - loss: 8.7405 - perplexity: 637.2114
Epoch 3/100
2282/2282 [=====] - 36s 16ms/step - loss: 8.6348 - perplexity: 641.9884
Epoch 4/100
2282/2282 [=====] - 36s 16ms/step - loss: 8.5838 - perplexity: 642.4123
Epoch 5/100
2282/2282 [=====] - 34s 15ms/step - loss: 8.5592 - perplexity: 643.7546
Epoch 6/100
2282/2282 [=====] - 33s 15ms/step - loss: 8.5095 - perplexity: 639.8062
.
.
.
Epoch 96/100
2282/2282 [=====] - 41s 18ms/step - loss: 7.3359 - perplexity: 260.1209
Epoch 97/100
2282/2282 [=====] - 34s 15ms/step - loss: 7.3367 - perplexity: 259.9016
Epoch 98/100
2282/2282 [=====] - 34s 15ms/step - loss: 7.3274 - perplexity: 256.5746
Epoch 99/100
2282/2282 [=====] - 33s 14ms/step - loss: 7.3156 - perplexity: 258.4492
Epoch 100/100
2282/2282 [=====] - 36s 16ms/step - loss: 7.3068 - perplexity: 255.5455
```

و نمودار



این خروجی مقادیر را با یک شبکه زیاد نسبت به دو تمرین قبل نشان میدهد
در ارزیابی مقادیر بسیار بد شده اند نسبت به دو تمرین قبل

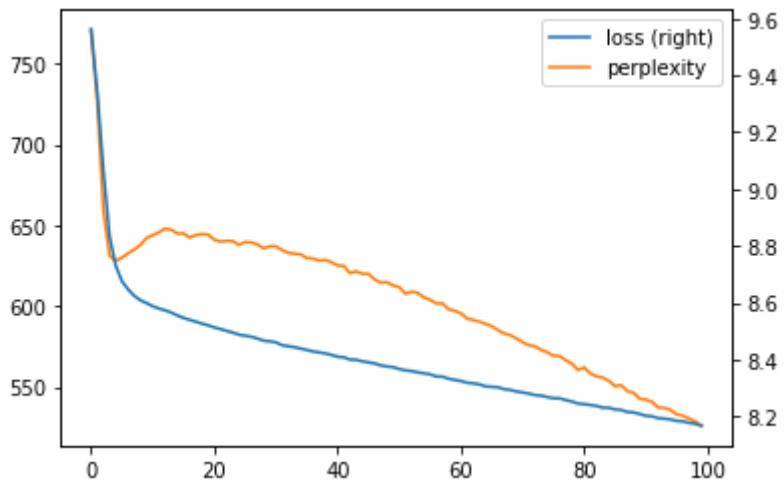
```
● ● ●
-----evaluate-----
386/386 [=====] - 4s 8ms/step - loss: 9.0279 - perplexity: 962.4387
[9.027881622314453, 962.4386596679688]
```

هر چه نرخ بالاتر میشود خروجی مقادیر در تست بدتر میشود !

ز) در این قسمت شبکه با ۵۰ لایه مخفی و نرخ یادگیری ۰.۰۲ تست شده است
خروجی به شکل زیر است

```
● ● ●
Epoch 1/100
2282/2282 [=====] - 14s 5ms/step - loss: 9.6057 - perplexity: 783.4779
Epoch 2/100
2282/2282 [=====] - 11s 5ms/step - loss: 9.3916 - perplexity: 741.2758
Epoch 3/100
2282/2282 [=====] - 11s 5ms/step - loss: 9.1449 - perplexity: 672.2089
Epoch 4/100
2282/2282 [=====] - 11s 5ms/step - loss: 8.8635 - perplexity: 632.6994
Epoch 5/100
2282/2282 [=====] - 11s 5ms/step - loss: 8.7414 - perplexity: 623.9763
.
.
.
Epoch 94/100
2282/2282 [=====] - 11s 5ms/step - loss: 8.1907 - perplexity: 534.8130
Epoch 95/100
2282/2282 [=====] - 11s 5ms/step - loss: 8.1799 - perplexity: 531.8100
Epoch 96/100
2282/2282 [=====] - 11s 5ms/step - loss: 8.1747 - perplexity: 526.8899
Epoch 97/100
2282/2282 [=====] - 12s 5ms/step - loss: 8.1908 - perplexity: 531.7630
Epoch 98/100
2282/2282 [=====] - 11s 5ms/step - loss: 8.1676 - perplexity: 526.6509
Epoch 99/100
2282/2282 [=====] - 11s 5ms/step - loss: 8.1572 - perplexity: 524.3673
Epoch 100/100
2282/2282 [=====] - 11s 5ms/step - loss: 8.1579 - perplexity: 523.7294
```

و نمودار



که در این مورد **loss** به شکل خوبی نزول کرده ولی **perplexity** بعد یک نوسان تقریباً زیاد نزول کرده

در ارزیابی دیتای تست خروجی تقریباً خوبی بسیت آمده است

```
● ● ●  
-----evaluate-----  
386/386 [=====] - 2s 4ms/step - loss: 8.7125 - perplexity: 590.3358  
[8.712544441223145, 590.3358154296875]
```

در ۱۰۰ لایه مخفی خروجی به شکل زیر است

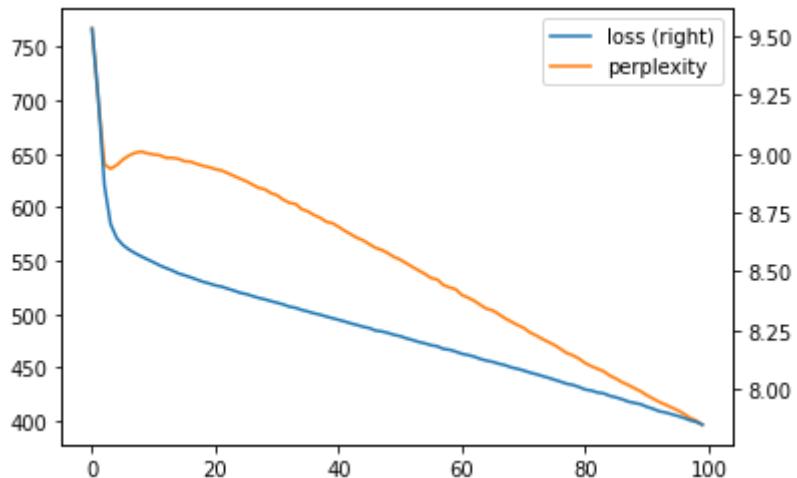
```

● ● ●

Epoch 1/100
2282/2282 [=====] - 13s 5ms/step - loss: 9.5942 - perplexity: 780.6418
Epoch 2/100
2282/2282 [=====] - 11s 5ms/step - loss: 9.3254 - perplexity: 723.9355
Epoch 3/100
2282/2282 [=====] - 11s 5ms/step - loss: 8.9251 - perplexity: 641.3890
Epoch 4/100
2282/2282 [=====] - 11s 5ms/step - loss: 8.7127 - perplexity: 628.7847
Epoch 5/100
2282/2282 [=====] - 11s 5ms/step - loss: 8.6462 - perplexity: 630.8429
Epoch 6/100
2282/2282 [=====] - 11s 5ms/step - loss: 8.6123 - perplexity: 639.9426
Epoch 7/100
2282/2282 [=====] - 11s 5ms/step - loss: 8.5916 - perplexity: 642.4192
Epoch 8/100
2282/2282 [=====] - 11s 5ms/step - loss: 8.5711 - perplexity: 644.6283
Epoch 9/100
2282/2282 [=====] - 11s 5ms/step - loss: 8.5531 - perplexity: 646.0143
.
.
.
Epoch 94/100
2282/2282 [=====] - 11s 5ms/step - loss: 7.8997 - perplexity: 413.2817
Epoch 95/100
2282/2282 [=====] - 11s 5ms/step - loss: 7.8918 - perplexity: 411.5627
Epoch 96/100
2282/2282 [=====] - 11s 5ms/step - loss: 7.8704 - perplexity: 406.7458
Epoch 97/100
2282/2282 [=====] - 11s 5ms/step - loss: 7.8643 - perplexity: 404.4717
Epoch 98/100
2282/2282 [=====] - 11s 5ms/step - loss: 7.8586 - perplexity: 400.8810
Epoch 99/100
2282/2282 [=====] - 11s 5ms/step - loss: 7.8524 - perplexity: 398.4087
Epoch 100/100
2282/2282 [=====] - 11s 5ms/step - loss: 7.8413 - perplexity: 393.9825

```

نمودار



که همینطور که مشاهده می شود یک شب بسیار نرمی دارد در Epoch های اول perplexity یک نوسان داشت و بعد نزولی شد در ارزیابی خروجی به شکل زیر است که نسبت به تمرین قبل ، یک مقدار بدتر شده است



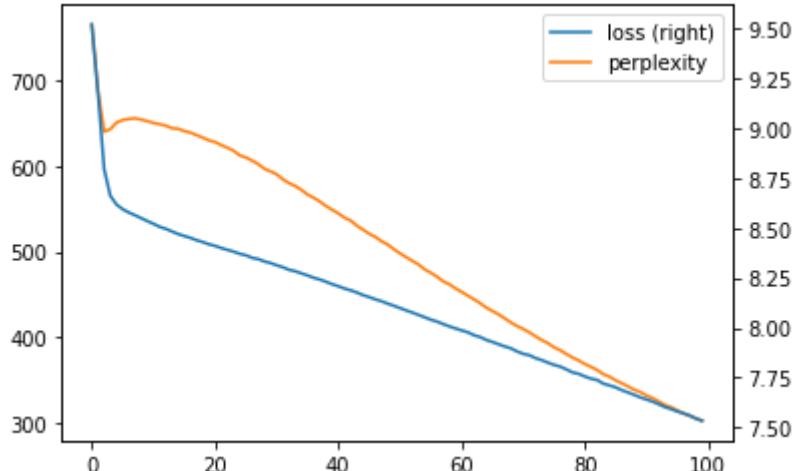
```
386/386 [=====] - 2s 4ms/step - loss: 8.7876 - perplexity: 650.9092
[8.787636756896973, 650.9092407226562]
```

در ۱۵۰ لایه مخفی خروجی به شکل زیر است



```
Epoch 1/100
2282/2282 [=====] - 14s 5ms/step - loss: 9.5919 - perplexity: 780.4068
Epoch 2/100
2282/2282 [=====] - 12s 5ms/step - loss: 9.2811 - perplexity: 712.3192
Epoch 3/100
2282/2282 [=====] - 11s 5ms/step - loss: 8.8539 - perplexity: 637.7193
.
.
.
Epoch 95/100
2282/2282 [=====] - 11s 5ms/step - loss: 7.5731 - perplexity: 314.4768
Epoch 96/100
2282/2282 [=====] - 11s 5ms/step - loss: 7.5722 - perplexity: 312.5231
Epoch 97/100
2282/2282 [=====] - 11s 5ms/step - loss: 7.5722 - perplexity: 309.7796
Epoch 98/100
2282/2282 [=====] - 11s 5ms/step - loss: 7.5442 - perplexity: 306.4101
Epoch 99/100
2282/2282 [=====] - 11s 5ms/step - loss: 7.5292 - perplexity: 302.1248
Epoch 100/100
2282/2282 [=====] - 11s 5ms/step - loss: 7.5403 - perplexity: 302.4000
```

و نمودار به شکل زیر است



که یک شیب زودتری نسبت به دو تمرین قبل تجربه کرده و در ارزیابی دیتای تست خروجی بدتری دارد



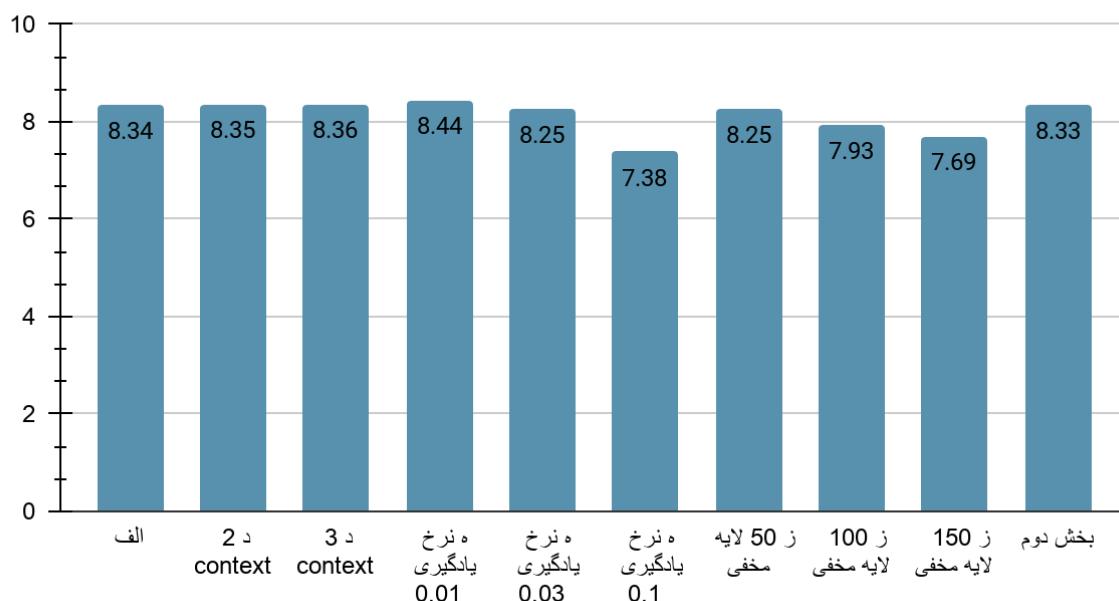
```
-----evaluate-----
386/386 [=====] - 2s 4ms/step - loss: 8.8779 - perplexity: 722.3127
[8.877877235412598, 722.3126831054688]
```

هر چه لایه مخفی بیشتر میشود در ارزیابی دیتای تست خروجی به مراتب بدتری مشاهده میشود !

تحلیل خروجی ها :

در ادامه خروجی تمرین ها را به شکل نموداری مشاهده میکنید این خروجی ها را با مقادیر کم در دیتای آموزشی بدست امده ولی قابل تحلیل هستند و تفاوت ها در تمرین قابل مشاهده است

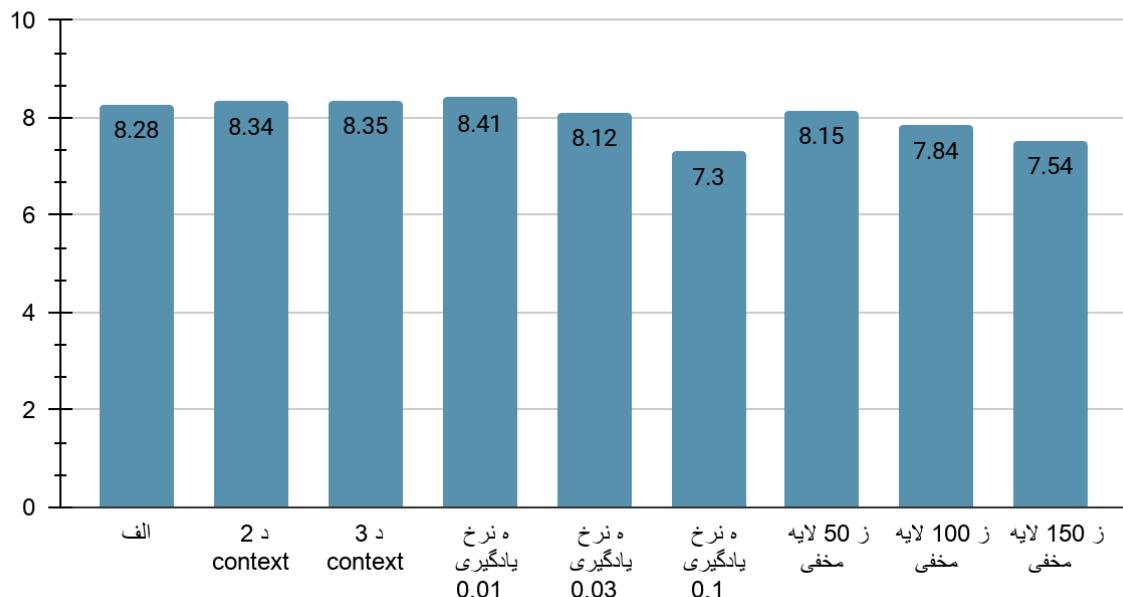
تحلیل خروجی loss بخش اول و دوم



در loss بخش اول ، کم ترین مقدار نرخ یادگیری 0.1 بود که نشان میدهد loss با افزایش نرخ یادگیری تغییر میکند و بیشترین مقدار هم مربوط به نرخ یادگیری 0.01 است که نشان می دهد کم بودن نرخ یادگیری می تواند چقدر در loss تاثیر بگذارد ! نرخ یادگیری مسئله مهمی است که باید دقیق برایش تصمیم گرفت و وقت گذاشت !

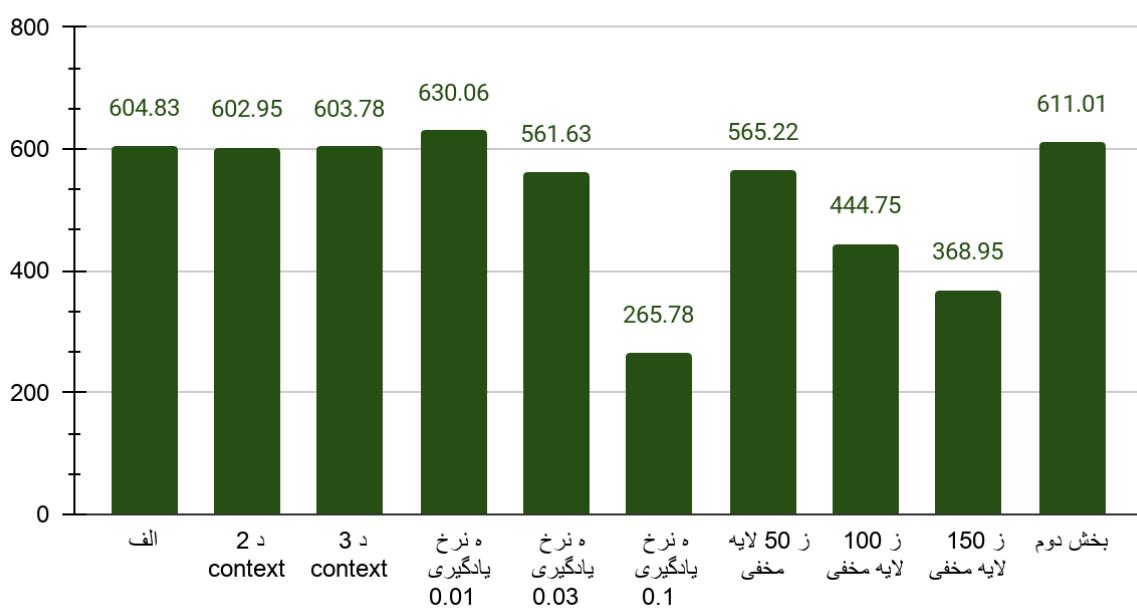
قسمت بخش دوم هم loss بالایی نسبتا ، نسبت به بقیه دارد که طبیعی است بخاطر نداشتن بردار های از پیش آموزش دیده شده !

تحلیل خروجی loss بخش سوم



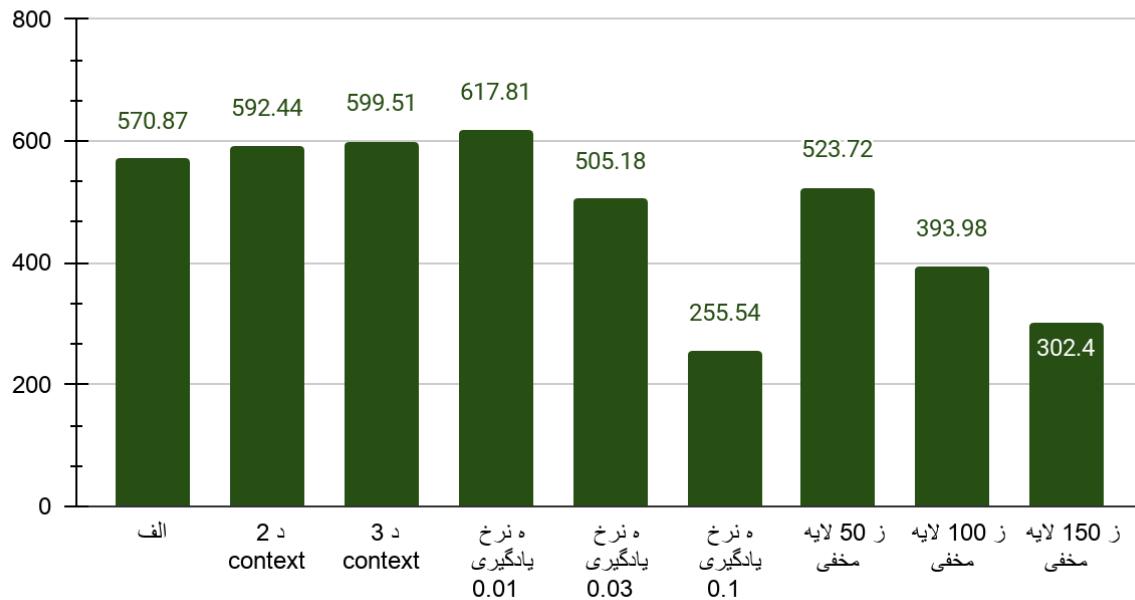
هر تمرینی که در بخش اول موجود است خروجی مقدار loss آن در شبکه های RNN کمتر است در بخش سوم کمترین مقدار loss مربوط به 0.1 است و بیشترین هم مربوط به نرخ یادگیری 0.01 است و همان نکته که در بالا گفته شد را تایید میکند همین جور در لایه ها مخفی زیاد بودن لایه باعث کاهش loss میشود و کم بودن context هم باعث کم شدن loss میشود البته تاثیر زیادی ندارد ولی بهتر است !

تحلیل خروجی perplexity بخش اول و دوم



در perplexity هم نرخ یادگیری مهم است و بهترین مقدار مربوط به این موضوع است در بخش دوم هم perplexity زیادی دارد و نشان دهنده این است که دیتا از قبل اموزش داده شده چقدر مهم است و میتواند در خروجی تاثیر بگذارد! در لایه مخفی هم هر چه لایه بیشتر باشد context کمتری داریم و در perplexity ها هم به همین شکل context کم تر perplexity بهتر

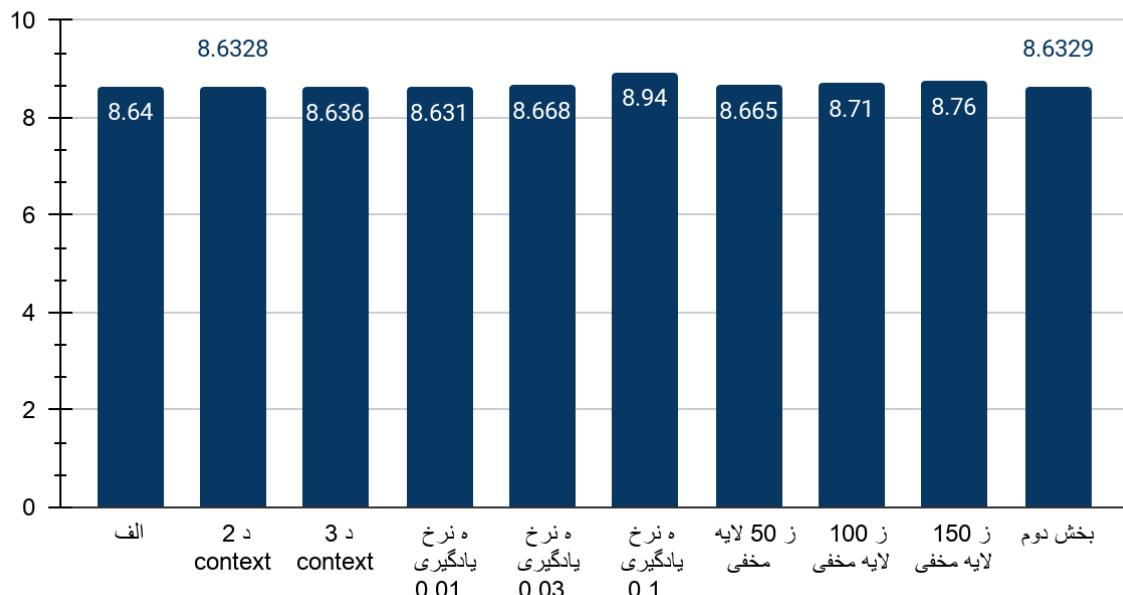
تحلیل خروجی perplexity بخش سوم



هر تمرینی که در بخش اول موجود است خروجی مقدار perplexity آن در شبکه های RNN کم تر است و کمترین قسمت هم مربوط به نرخ یادگیری 0.01 است بیشترین هم مربوط به نرخ یادگیری 0.01 در لایه های مخفی لایه بیشتر perplexity بهتر است و جالب در این است که در RNN در قسمت context ۴ مقدار ۰.۰۱ برای تمرین alf است مقدار بهتری نسب به ! 2,3 context

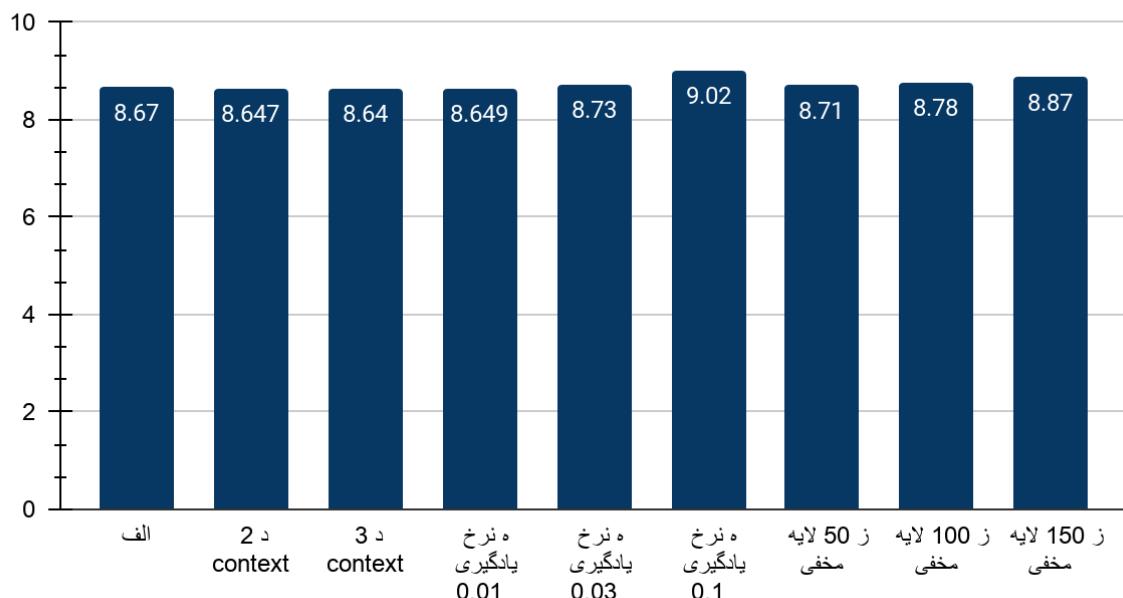
در قسمت بعد به تحلیل ارزیابی دیتا نیست بر روی مدل اشاره می شود

تحلیل loss مقادیر تستی بخش اول و دوم



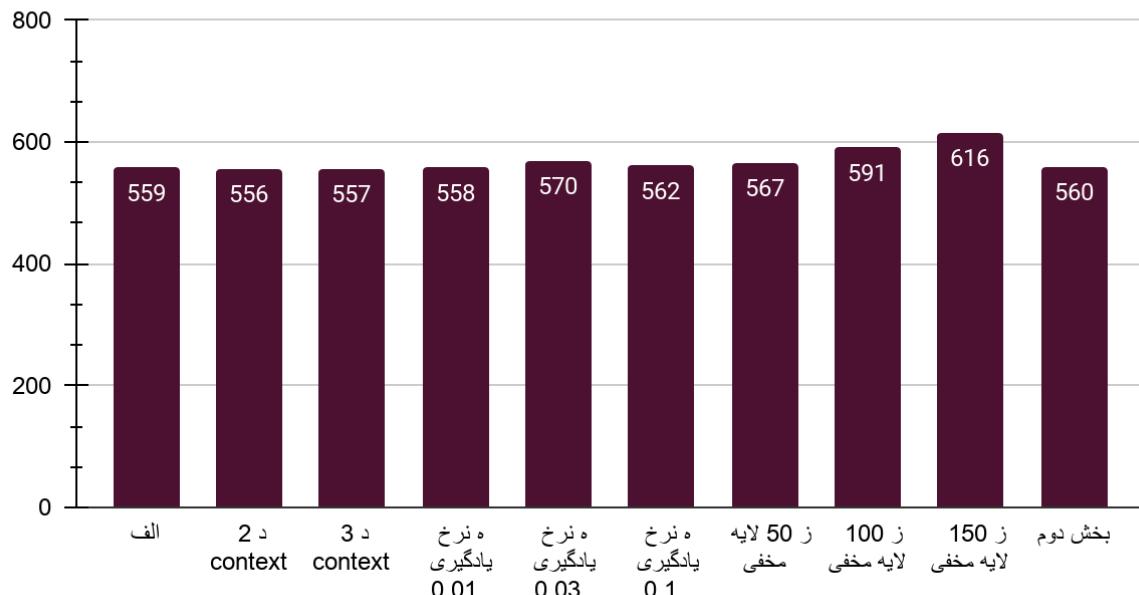
همان طور که در loss بخش اول و دوم دیده می شود بهترین نتیجه برای نرخ یادگیری 0.01 است که در قسمت آموزش مدل بد ترین قسمت بود و در این قسمت بهترین و جالب در این است که بد ترین مقدار برای نرخ 0.1 است که در قسمت آموزش مدل بهترین بود و در لایه مخفی هم لایه ای که کمتر مقدار دارد بهترین نتیجه داده آنچه برداشت می شود از این است که در قسمت ارزیابی دقیقا نتایج بر عکس شده اند ! در بخش دوم هم یک loss نسبتا بدی گرفته !

تحلیل loss مقادیر تستی بخش سوم



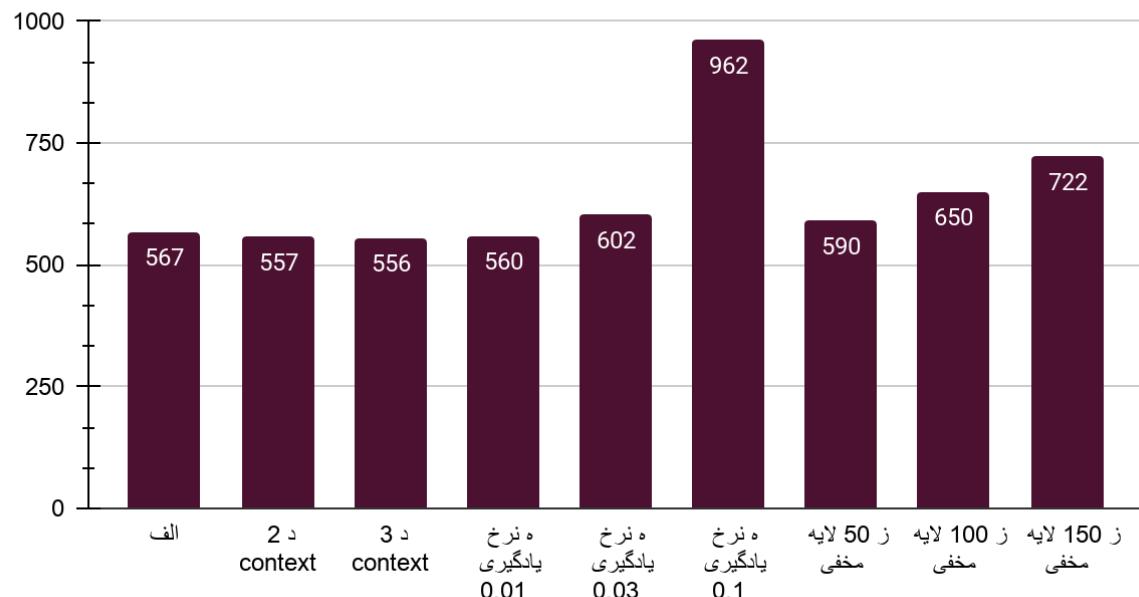
در مدل های RNN هم نتیجه Loss بدتر شده نسبت به بخش اول و بدترین هم برای نرخ یادگیری 0.1 است و بهترین هم برای تمرین ورودی 3 context است!

تحلیل perplexity مقادیر تستی بخش اول و دوم



در قسمت perplexity نتیجه جالبی از خود داشته و مقدار بهتری نسب به قسمت آموزش مدل داشته بهترین قسمت هم برای 2 بوده و بدترین هم 150 لایه مخفی در قسمت نرخ هم بهترین برای نرخ 0.01 بوده و بدترینش 0.03 است جالب در این است که در بخش دوم یک نتیجه متوسط رو به پایینی گرفته

تحلیل perplexity مقادیر تستی بخش سوم



در مدل های RNN هم قضیه یک فرق کرده و نرخ یادگیری 0.1 که در قسمت تمرین اول متوسط بود در این قسمت مقدار زیادی داشته و بدترین قسمت شده بهترین قسمت هم برای 3 context بوده است

نتیجه:

نتیجه که از این تمرین قابل برداشت است این است که نرخ یادگیری بسیار مهم است و در انتخابش باید دقیق و مدل های RNN به مراتب نتیجه بهتری دارن و داشتن دیتای از پیش آموزش داده شده نتیجه بهتری برای ما به ارمغان می آورد

با تشکر