به نام خدا

دانشگاه تهران

پردیس دانشکده‌های فنی

دانشکده برق و کامپیوتر

پـزدازش زبـان هـای طبـیـعـی

CA3

فاطمه سلیقه

۸۱۰۱۹۸۳۰۶

فروردین ماه ۱۳۹۹

سوال ۱

الف و ب)

ابتدا از روی فایل داده های آموزش خوانده سپس پیش پردازش های لا زم را انجام داده و در نهایت متن آموزش را به صورت لیستی از کلمات که پشت سر هم آمده اند در می آوریم .

سپس از اول لیست شروع می کنیم و ۴ کلمه ای که پشت سر هم امده اند را در یک لیست قرار می دهیم و کلمه پنجم را به عنوان label (خروجی مورد نظر) در نظر می گیریم . نمونه ای از خروجی تاکنون:

sequences	labels
['many are the hours', 'are the hours in', 'the hours in which', 'hours in which i', 'in which i have', 'which i have unk',	['in', 'which', 'i', 'have', 'unk', 'upon',

سپس با استفاده از tokenizer دو لیست بالا را به صورت عددی در ما اوریم . از آنجایی که خروجی شبکه یک آرایه با اندازه vocabulary است بنابراین لازم است تا خروجی را به صورت بردار one_hot با اندازه vocabulary در بیاوریم . از آنجایی که اندازه vocabulary برابر 62155 و تعداد داده ها برابر 819575 است . بنابراین خروجی با ابعاد 62155*819575 می شود . این اندازه بسیار بزرگی است و با خطای حافظه مواجه بودم .(در google colab باز هم با مشکل مواجه شد) . برای حل مشکل vocabulary را محدود کرده و ۲۰۰۰ کلمه پرتکرار را انتخاب نمودم و به جای بقیه کلمات unk گذاشتم .

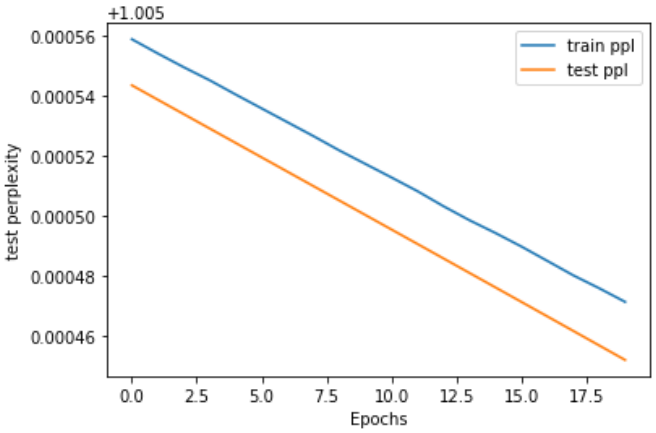
بنابراین در نهایت داده ها به صورت زیر در آمدند :

```
array([[  96,   31,    2,  607],      array([[[0, 0, 0, ..., 0, 0, 0],
       [  31,    2,  607,    5],             [0, 0, 0, ..., 0, 0, 0],
       [   2,  607,    5,   30],             [0, 0, 0, ..., 0, 0, 0],
       ...,                                  ...,
       [1932,   14,   17,    8],             [0, 1, 0, ..., 0, 0, 0],
       [  14,   17,    8,   45],             [0, 0, 0, ..., 0, 0, 0],
       [  17,    8,   45,   37]])            [0, 0, 0, ..., 0, 0, 0]])
```

حال برای طراحی شبکه ، لایه اول یک Embedding است که اندازه وزن های آن را با استفاده از GloVe بدست می آوریم . سپس یک لایه پنهان به ۳۵ نورون قرار داده و از تابع tanh برای activation استفاده نموده و در نهایت لایه خروجی با اندازه ارایه های به طول اندازه vocabulary و softmax می باشد . برای به روز رسانی پارامترها هم از روش stochastic gradient descent استفاده شده است و تابع هزینه هم binary_crossentropy می باشد . learning rate را هم در ابتدا برابر 0.02 قرار داده شده است .

ج)

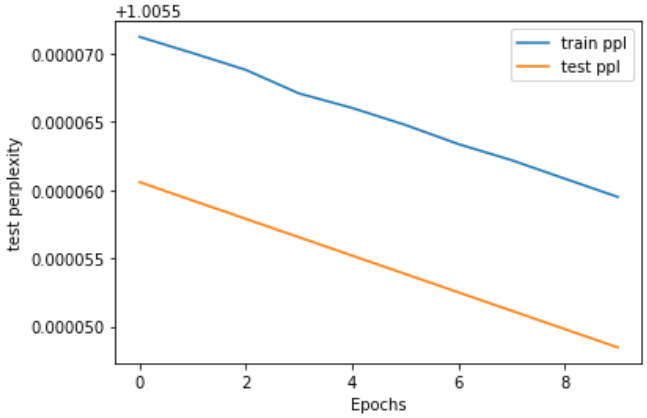
```
loss =  0.004251258724443608 acc =  0.9995003938827065
```



د)

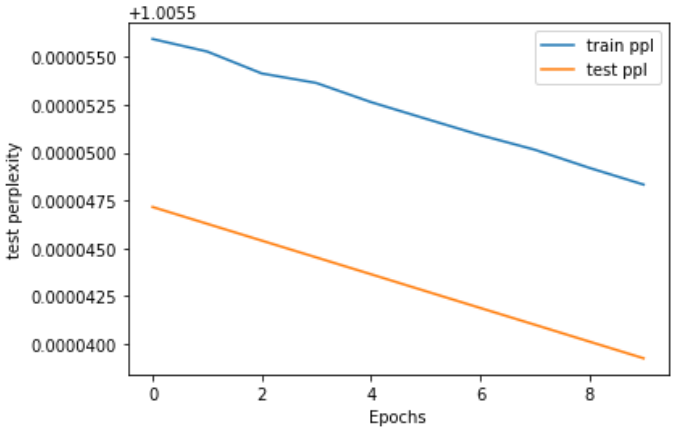
Context = 3

```
loss =  0.0055331340027785746 acc =  0.9993342756859924
```



Context = 2

```
loss =  0.00552396107219541 acc =  0.9993342757012065
```



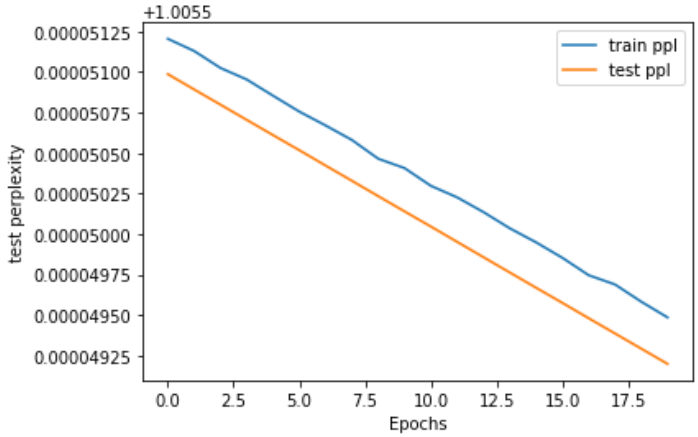
با توجه به نتایج به دست آمده زمانی که تعداد context برابر ۴ است بهترین حالت است و سپس زمانی که برابر ۲ است و پس از آن زمانی که برابر ۳ است .

به صورت منطقی هر چه تاریخچه بیشتر باشد یعنی تعداد کلمات context بیشتر باشد نتیجه باید بهتر باشد . نتایج به دست آمده نشان میدهد که تعداد context برابر ۴ باشد نتیجه بهتر است و تفاوت آن با بقیه بیشتر است . اما باید در حالت context برابر ۳ از ۲ بهتر می بود اما بسیار نزدیک هم شدند . شاید به دلیل خود داده های تست باشد .

ه)

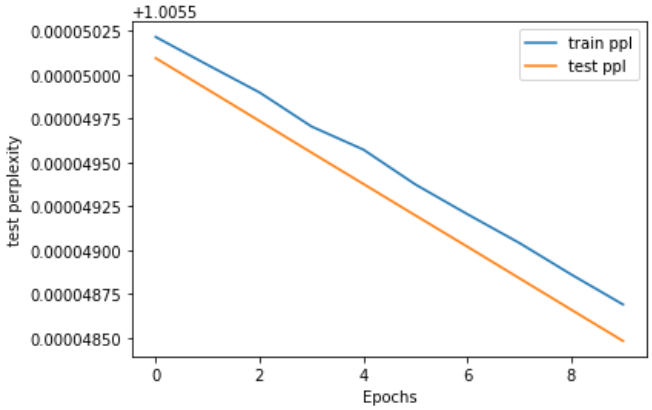
Learning rate = 0.01

```
loss =  0.005533857861893853 acc =  0.9993342757072898
```



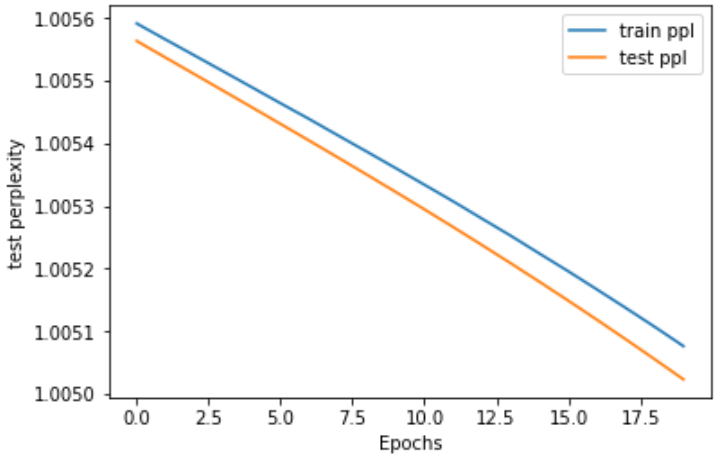
Learning rate = 0.03

```
loss =  0.005533146970997169 acc =  0.9993342757072898
```



Learning rate = 0.1

```
loss =  0.005010515466336049 acc =  0.9993342757072898
```

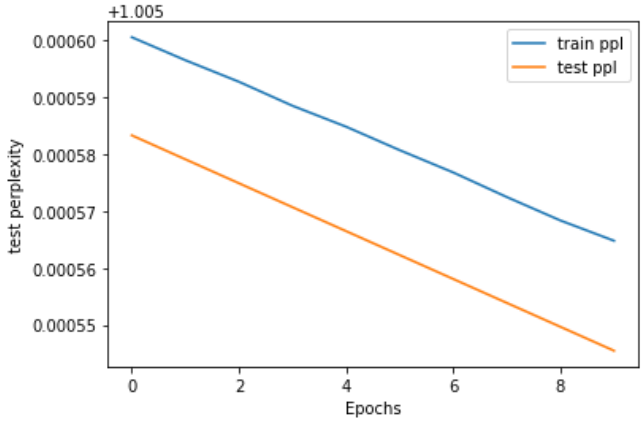


با توجه به نتایج به دست آمده از تست می بینیم که lrبرابر 0.02 از همه بهتر و سپس 0.1 و پس از آن 0.03 و 0.01 تقریبا به یک اندازه هستند .

ز)

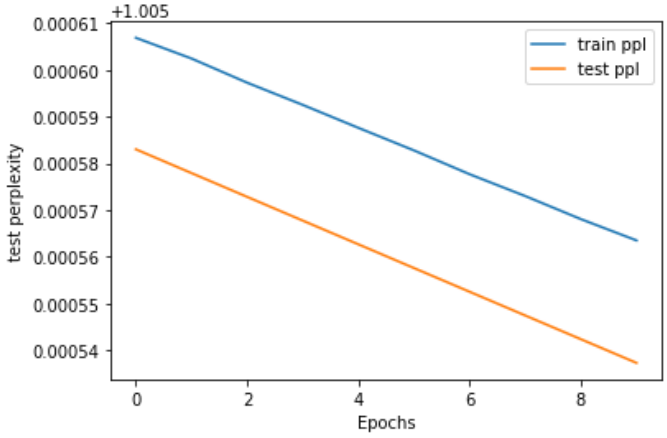
تعداد نورون ۵۰ :

```
loss =  0.005530127469384436 acc =  0.9993342757072898
```



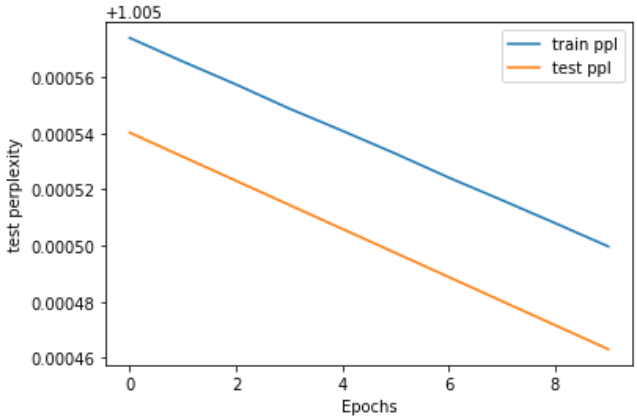
تعداد نورون ۱۰۰ :

```
loss =  0.005522077924094266 acc =  0.9993342757072898
```



تعداد نورون ۱۵۰:

```
loss =  0.005448177159402117 acc =  0.9993342757072898
```

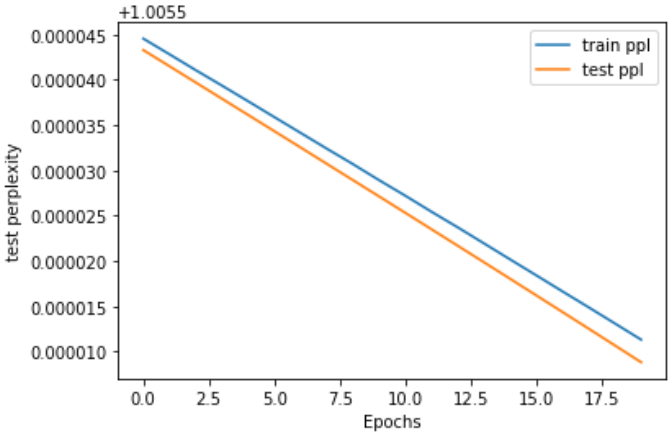


با توجه به نتایج به دست آمده با افزایش تعداد نورون ها مقدار loss و perplexity کاهش می یابد در نتیجه بهتر عمل می کند . همان گونه که در شبکه های دیگر هم افزایش تعداد نورون ها به بهتر شدن نتیجه کمک می کند .

سوال ۲

برای این قسمت شبکه مورد نظر پیاده سازی شد و نمودار زیر از ppl به دست آمده :

```
loss =  0.005493701212362967 acc =  0.9993342757072898
```



همان طور که از نتایج به دست آمده می بینیم در قسمت دوم شبکه بهتر عمل می کند یعنی perplexity بهتر است زیرا بردارهای لایه Embedding براساس داده خودمون آموزش داده می شود و از وزن های آماده استفاده نشده است .