سوال ۱
الف

ساختار کلی فایل‌های آپلود شده به شکل زیر است:

`Q1.py`

این فایل شامل کدهایی است که برای بخش الف تا ز سوال ۱ زده شده است و در آن پارامترهای مختلف و تاثیر آن‌ها بررسی شدند.

`constants.py`

شامل مقادیر ثابت مثل آدرس فایل ورودی و فایل Glove است.

`Q2.py`

در این فایل شبکه عصبی با مقادیر مورد نظر سوال ۲ صدا شده است.

`data_reader.py`

این فایل شامل توابعی است که مسئول خواندن داده‌ها و فایل‌ها هستند.

`neural_network.py`

شبکه عصبی مورد نظر در این فایل پیاده سازی شده است.

`pre_processing.py`

در این فایل توابعی پیاده سازی شده‌اند که ورودی را گرفته و آن را tokenize کرده و تمامی کلمات را به حروف کوچک تبدیل کرده و کلمات اضافه را نیز حذف می‌کنند.

`main.py`

این فایل فایل اجرایی است که در جای مناسب توابع پیاده سازی شده در فایل‌های دیگر را صدا می‌کند.

همان طور که بالاتر اشاره مدل یادگیری در فایل neural_network.py پیاده سازی شده است. برای پیاده سازی مدل خواسته شده از کتابخانه‌ی keras کمک می‌گیریم. شکل زیر ساختار کلی این مدل را نمایش می‌دهد:

```
model = Sequential([
    Dense(self.number_of_context_words*self.projection_size, input_shape=(self.number_of_context_words*self.projection_size,)),
    Activation('linear'),
    Dense(self.number_of_hidden_layer_neurons),
    Activation('sigmoid'),
    Dense(self.vocab_size),
    Activation('softmax')
])
print(model.summary())
sgd = optimizers.SGD(lr=self.learning_rate, decay=1e-6, momentum=0.9, nesterov=True)
model.compile(optimizer=sgd,
      loss='categorical_crossentropy',
      metrics=['accuracy'])
cp = calculate_perplexity(self.train_data_X, self.train_data_Y)
history = model.fit(self.train_data_X, self.train_data_Y, epochs=number_of_epochs, batch_size=256, validation_data=(self.test_data_X, self.test_da
self.plot_accuracy_changes(history)
cp.plot_per()
```

برای این که بتوانیم perplexity را محاسبه کنیم یک تابع Callback می‌نویسیم که در پایان هر batch (on batch end) مقدار perplexity را محاسبه کند. در انتها نمودار دقت و perplexity را رسم می‌کنیم. (این تابع مخصوص در کلاس calculate_perplexity تعریف شده که در شکل بالا در متغیر cp ذخیره شده است.)

هر یک از حالات خواسته شده در صورت سوال برای ۲۰ ایپاک آموزش داده می‌شوند.

به دلیل مشکل Memory Error و عدم دسترسی به سیستمی که حافظه‌ی بهتری داشته باشد مجبور شدم آموزش را تنها روی بخشی از داده‌های فراهم شده انجام دهم.

ب

ساختار کلی شبکه مشابه ساختار رسم شده در صورت سوال است یعنی همان طور که در شکل زیر دیده می‌شود سایز لایه‌ی ورودی برابر با تعداد کلمات context است که طول هر یک نیز برابر با اندازه/بزرگی embedding آن است پس سایز لایه‌ی ورودی ضرب این دو عدد است. پس از آن سایز سایر لایه‌ها نیز برابر با سایز نشان داده شده در شکل است.

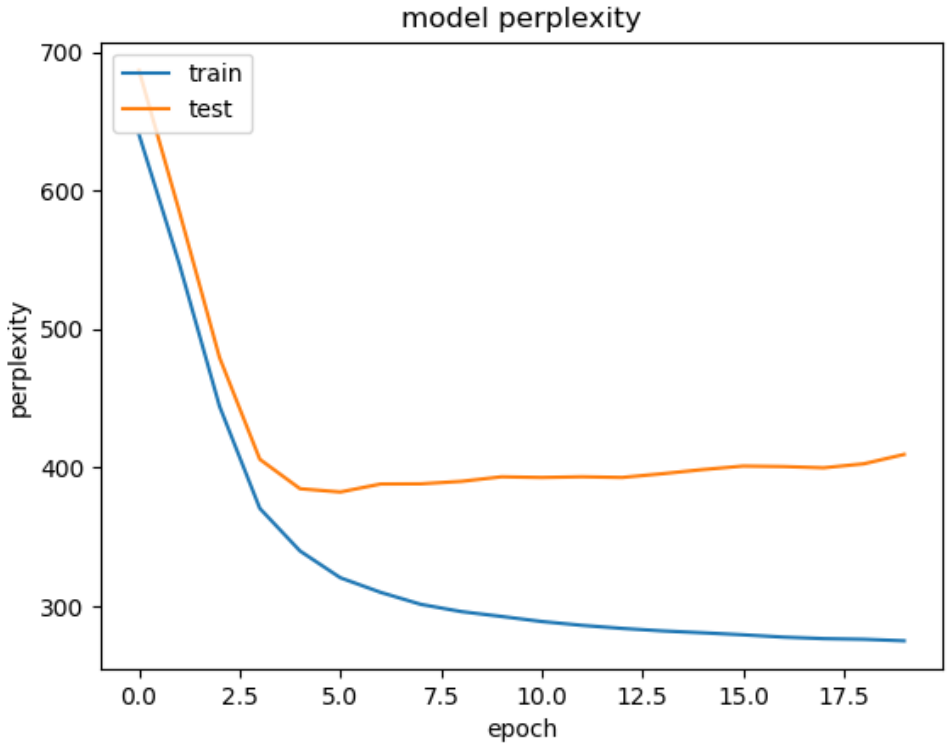
```
model = Sequential([
    Dense(self.number_of_context_words*self.projection_size, input_shape=(self.number_of_context_words*self.projection_size,)),
    Activation('linear'),
    Dense(self.number_of_hidden_layer_neurons),
    Activation('sigmoid'),
    Dense(self.vocab_size),
    Activation('softmax')
])
```

مقدار دهی اولیه‌ی پارامترها به این شکل است که learning rate را در تابع SGD که optimizer ما است تنظیم می‌کنیم. پارامترهای دیگری نیز نظیر تابع loss را متناسب با راهنمایی سوال تنظیم می‌کنیم. از آنجا که از keras برای این پیاده سازی کمک گرفته شده است به روز رسانی وزن‌ها در همین کتابخانه پیاده سازی شده است و دیگر نیازی نیست این به روز رسانی را ما به صورت دستی انجام دهیم. شرط پایان نیز با توجه به تعداد ایپاک مشخص می‌شود. یعنی هر زمان که به تعداد ایپاک بیان شده آموزش تکرار شد یادگیری تمام می‌شود.

ج

نمودار perplexity به شکل زیر است:

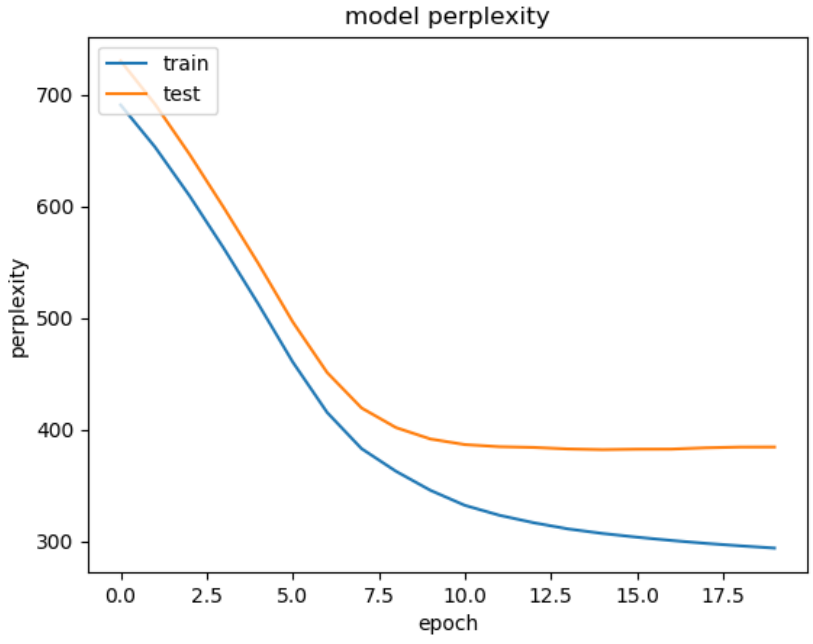
می‌توان دید که روند کلی کاهشی است و به طور کلی هم عملکرد روی داده‌های آموزش بهتر از داده‌های تست است که منطقی و قابل پیش‌بینی است. می‌توان دید که در جایی فاصله‌ی دو نمودار از هم بیشتر می‌شود، این جایی است که مدل شروع به overfit شدن کرده است و احتمالا نقطه‌ای است که باید آموزش را متوقف کنیم.



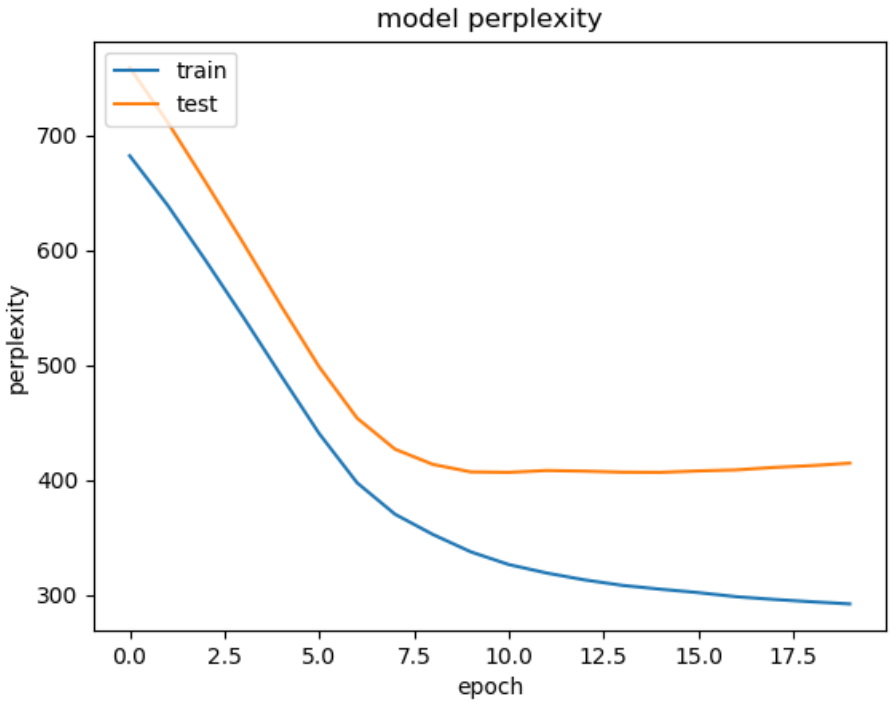
د

نمودارهای مختلف بر اساس اندازه‌ی context word به شکل زیر هستند. می‌توان دید که با کاهش تعداد context word ها عملکرد مدل بدتر می‌شود. اگر به context wordها به چشم دانش مدل نگاه کنیم منطقی است که با کم کردن دانش مدل و میزان اطلاعاتی که دارد عملکرد ضعیف‌تری را مشاهده کنیم.

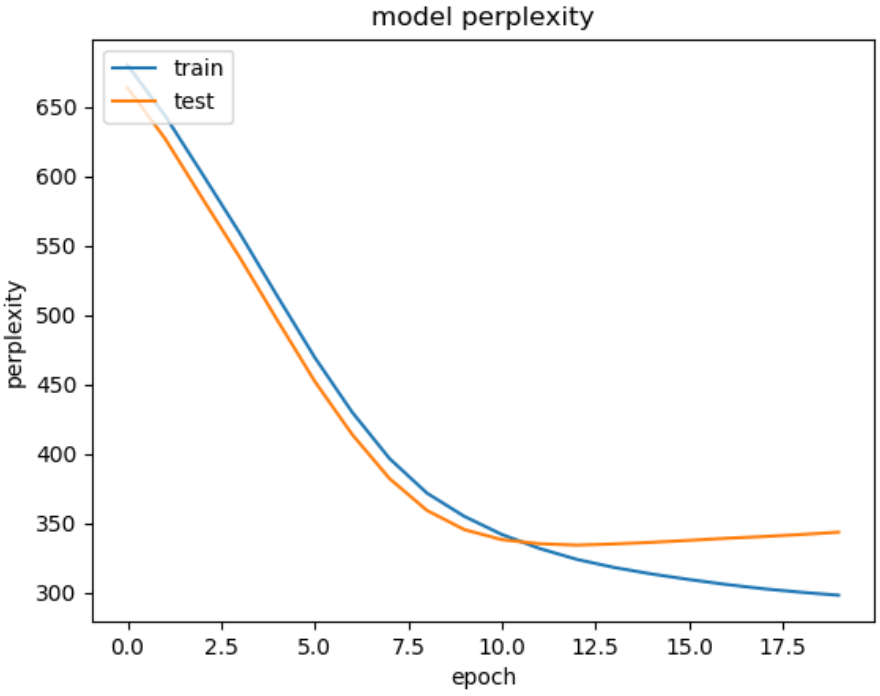
number of context words = 4



number of context words = 3



number of context words = 2

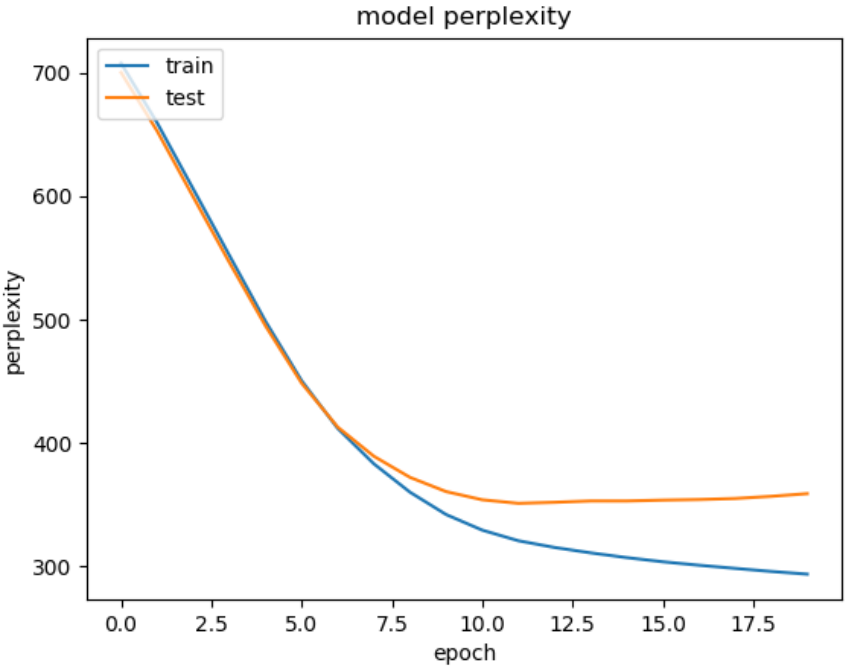


۵

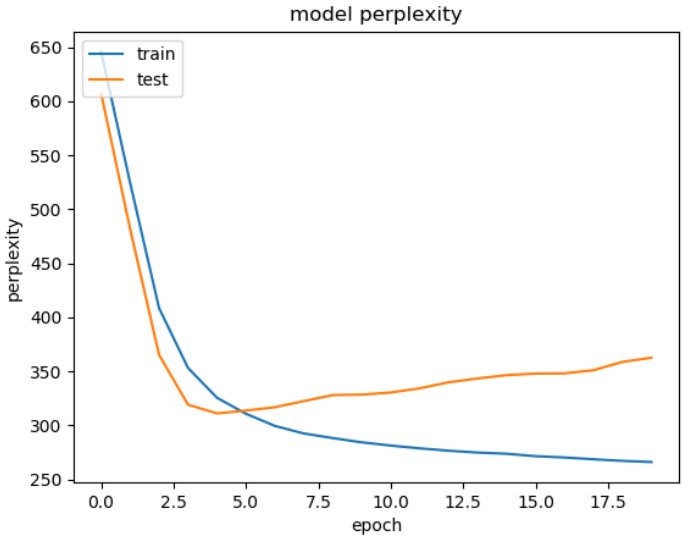
نمودارها با مقادیر learning rate مختلف به شکل زیر است:

با توجه به این نمودارها می‌توان دید که با افزایش نرخ یادگیری ابتدا دقت روی آموزش و تست بهبود پیدا می‌کند و سرعت همگرایی به بهترین جواب نیز بهتر می‌شود اما وقتی نرخ یادگیری زیادی بزرگ می‌شود باعث می‌شود که نتایج روی داده‌های آموزش بهتر از حالت قبل شود ولی اختلاف میان آموزش و تست را زیادتر می‌کند.

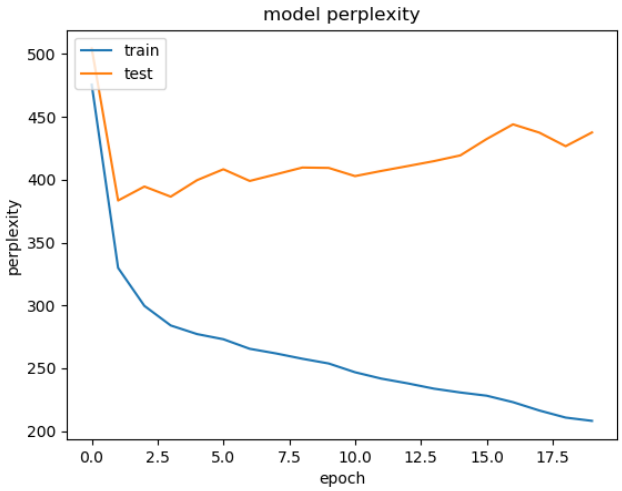
learning rate = 0.01



learning rate = 0.03



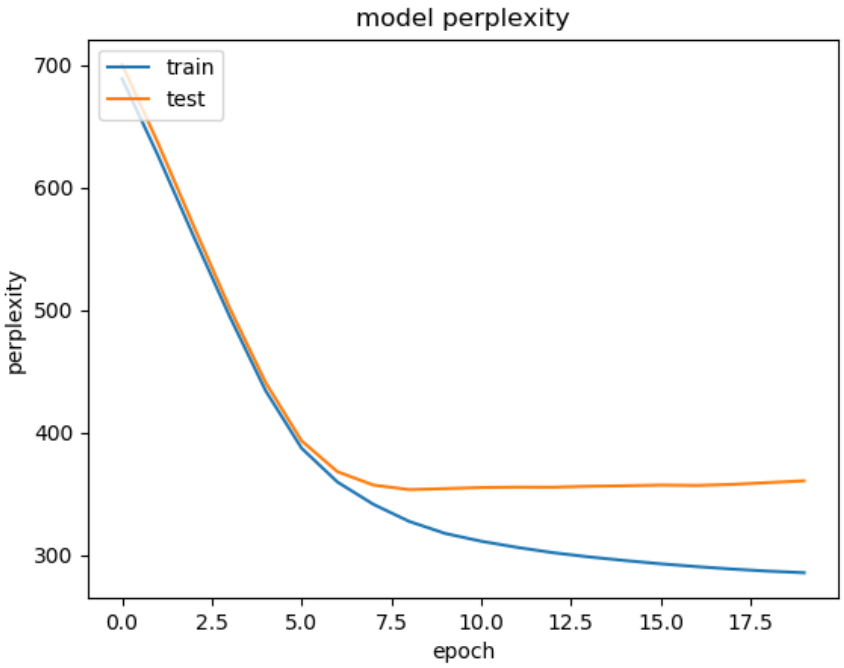
learning rate = 0.1



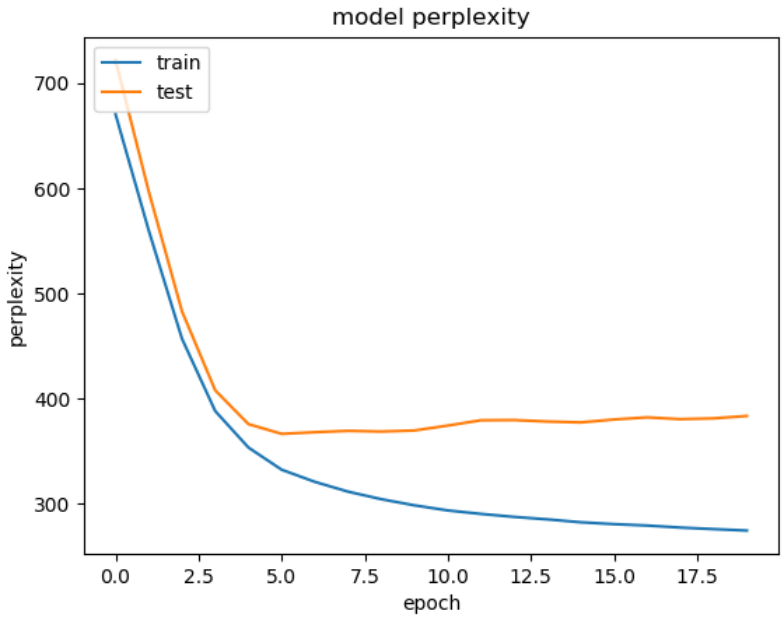
ز

نمودارها با مقادیر مختلف تعداد نرون‌های لایه میانی به شکل زیر است:

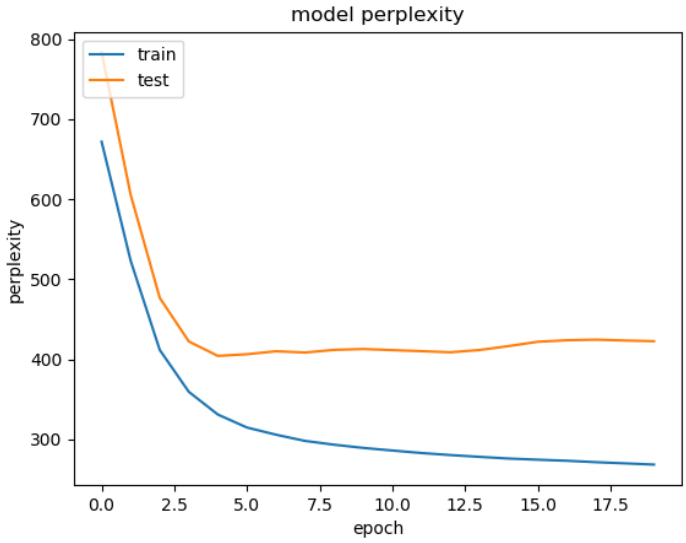
hidden layer nodes = 50



hidden layer nodes = 100



hidden layer nodes = 150



می‌توان دید که در تعداد اپیاک برابر با بالا رفتن تعداد نرون‌های لایه میانی چون پارامترها بیشتر می‌شود مدل عملکرد بهتری ندارد چون به اپیاک های بیشتری برای آموزش نیاز دارد و اختلاف آموزش و تست نیز زیاد می‌شود ولی انتظار داریم با بالارفتن اپیاک‌ها تعمیم پذیری مدل با نرون‌های لایه میانی بیشتر بیشتر باشد.

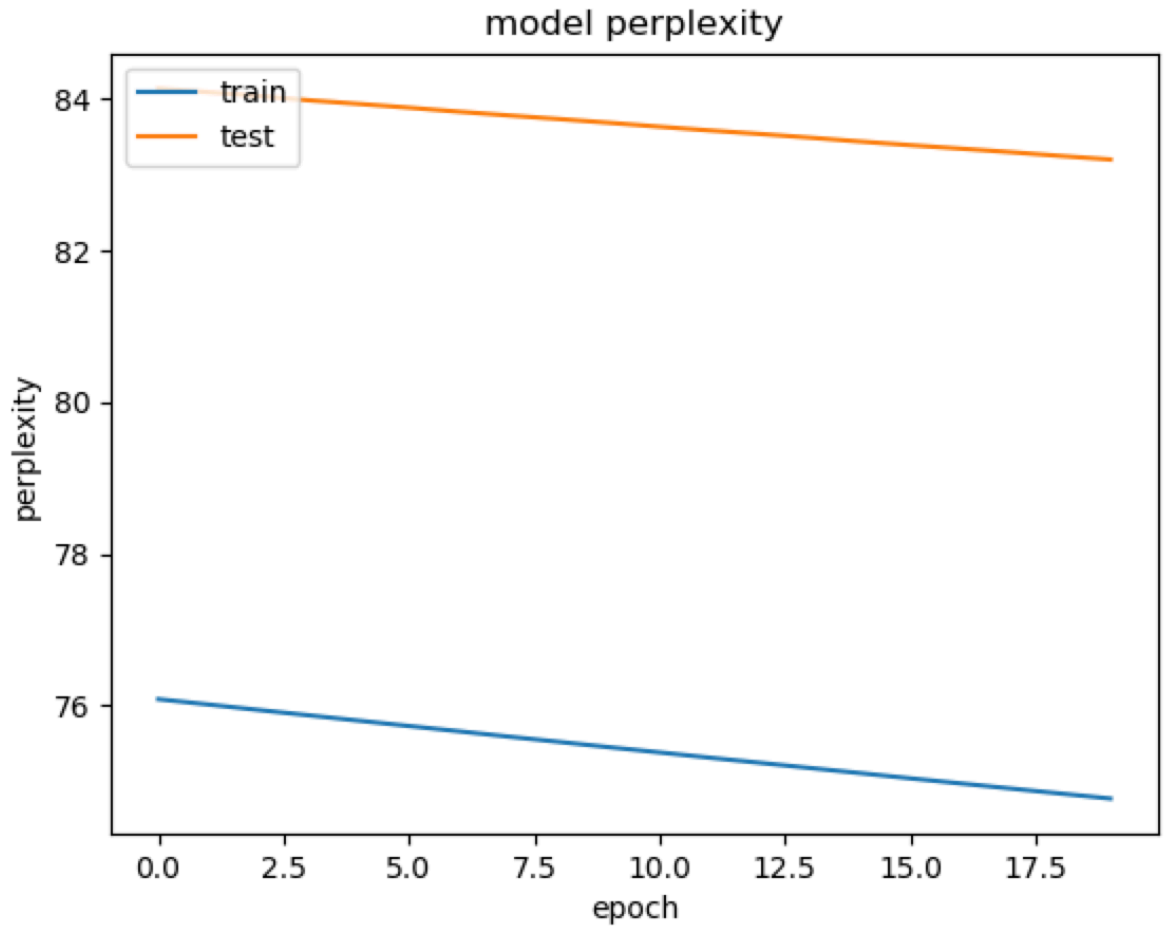
سوال ۲
الف

پیاده سازی شبکه مشابه حالت قبل است با این تفاوت که یک لایه‌ی Embedding به ابتدای مدل اضافه می‌شود تا با این کار یاد بگیریم که ورودی one-hot گرفته شده را چطور به یک نمایش ۵۰ تایی برسانیم.

ب

نمودار به شکل زیر است:

می‌توان دید که در این حالت عملکرد بدتری داشته ایم چون تعدادی پارامتر اضافه شده است و در نتیجه نیاز به زمان یادگیری بیشتری داریم پس با همان تعداد ایپاک نمی‌توان به دقت مشابه حالت نشان داده شده در سوال ۱ رسید. به علاوه Embedding های استفاده شده در الف خود پس از چند صد ایپاک تولید شده اند در نتیجه دقت و خوبی representation آن‌ها خیلی خیلی بالاتر است.



حجم کم داده و تعداد ایپاک‌های کم باعث می‌شود نمودار به این شکل نامطلوب باشد.