رامتین احسانی – سینا اسکندری

لینک گیت هاب: https://github.com/ramtin-ehsani/Movie-Genre-NLP

فاز نهایی پروژه درس مبانی پردازش زبان و گفتار

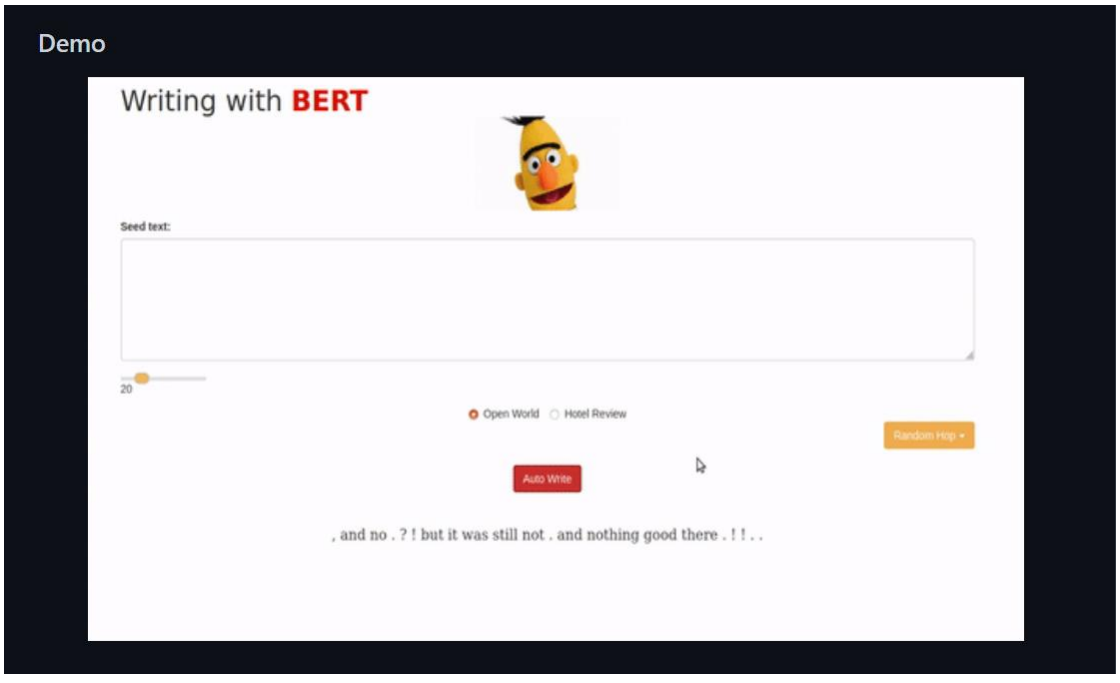
بخش اول: تولید جملات

برای این بخش، از آن جایی که دیتای ما به زبان انگلیسی میباشد از مدل pretrained شده BERT استفاده کردیم.

به این صورت که مدل را در کد لود کردیم و بر روی دیتای از پیش تهیه شده خودمان train کردیم. پس از به پایان رسیدن آموزش و تست این مدل، مدل نهایی را در drive ذخیره کردیم.

برای بخش تولید جملات از پروژه زیر کمک گرفتیم:

https://github.com/prakhar21/Writing-with-BERT



تمامی فایل های تولید شده توسط کد خودمان را به پروژه بالا میدهیم و با استفاده از مدل trained شده ما، این کد شروع به تولید جملات میکند.

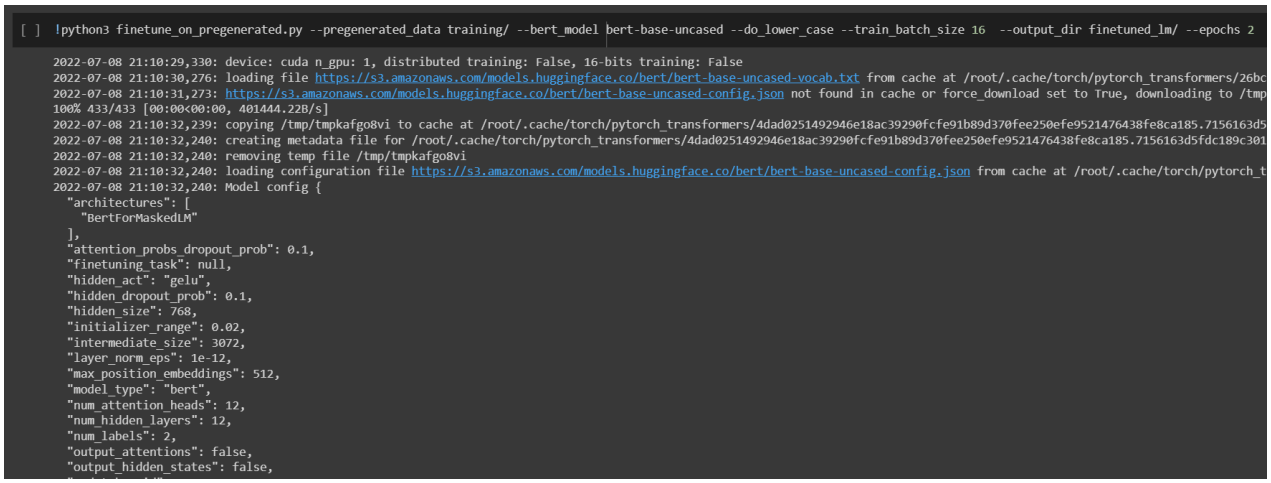
لینک مدل و سایر موارد تولید شده در این مرحله:

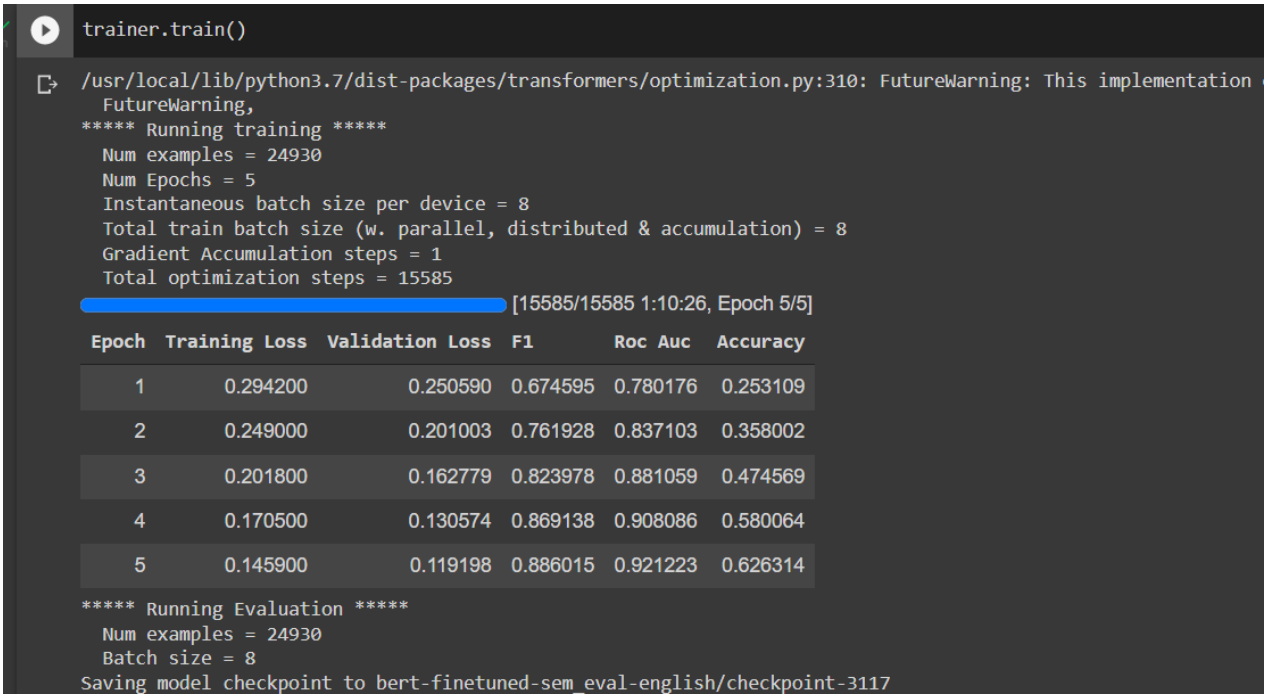
https://drive.google.com/drive/folders/1ZEW6xOXzSCHRDrdU2RG4F5KzFy9CGwCM?usp=sharing

لینک گوگل کولب:

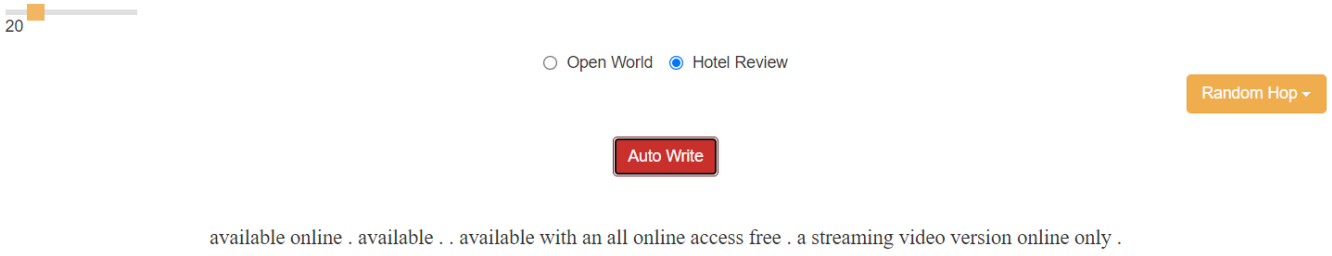
https://colab.research.google.com/drive/1vwxrLCMafSY0dWWNMeZ3EEUEjNAJVv6s?usp=sharing

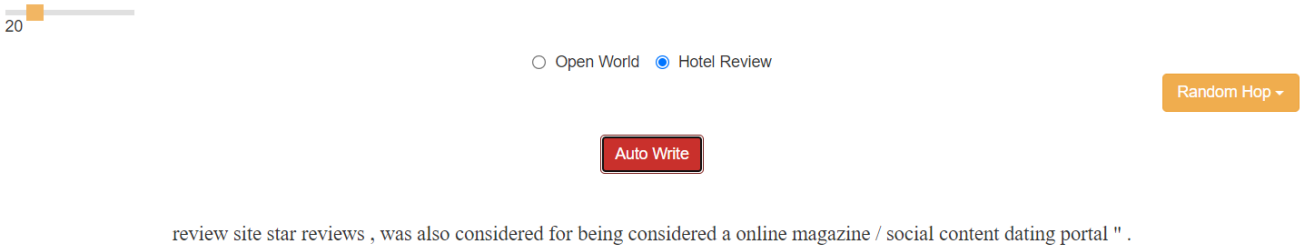
آموزش و evaluation مدل BERT:





نتایج حاصل شده:





دقت شود که اگر با گزینه های جایگزین مانند GPT2 و LM مبتنی بر LSTM/GRU مقایسه شود، BERT برای کارهایی مانند NLG مناسب نخواهد بود، فقط به این دلیل که در وهله اول به سبک معمولی رگرسیون خودکار آموزش داده نشده است. BERT به عنوان مدل زبان ماسک شده (MLM) در سبک دو جهته آموزش داده شده است. در MLM به‌جای پیش‌بینی هر نشانه بعدی، درصدی از نشانه‌های ورودی به‌صورت تصادفی پوشانده می‌شوند و تنها آن نشانه ها بر اساس کلمات باقی مانده در سمت چپ و راست آن پیش‌بینی می‌شوند، که به آن زمینه دو سویه غنی می‌دهد.

برای اجرای کد، فایل مدل ها (nlp) را از درایو دانلود کرده و در دایرکتوری قسمت BERT قرار دهید.

سپس app.py را اجرا کرده و با استفاده از فایل index.html به سرور request بزنید و جمله تولید کنید.

بخش دوم:

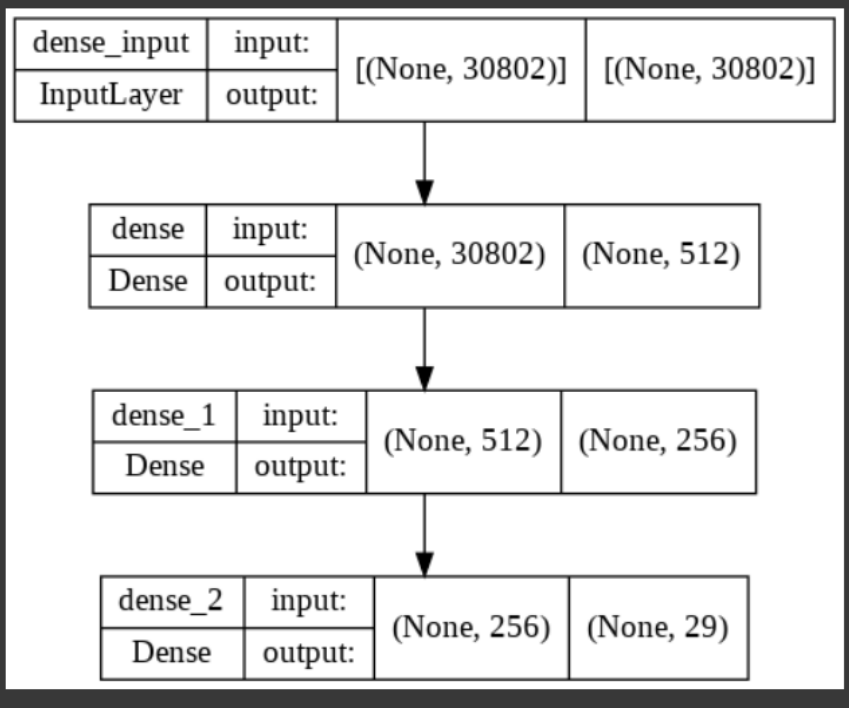
لینک گوگل کولب:

https://colab.research.google.com/drive/1SWUvSaA2pqashAqa7xxaIvcuSKBRWxOj?usp=sharing

برای این بخش از یک مدل ساده dense استفاده کردیم که در شکل زیر قابل مشاهده میباشد:

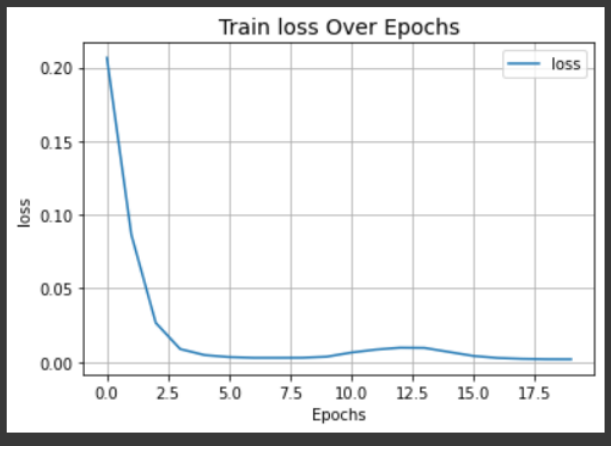
```
[19] print(model.summary())

Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 512)               15771136

 dense_1 (Dense)             (None, 256)               131328

 dense_2 (Dense)             (None, 29)                7453

=================================================================
Total params: 15,909,917
Trainable params: 15,909,917
Non-trainable params: 0
_________________________________________________________________
None
```

```
[22] from keras.utils.vis_utils import plot_model
     plot_model(model, to_file='model_plot4a.png', show_shapes=True, show_layer_names=True)
```

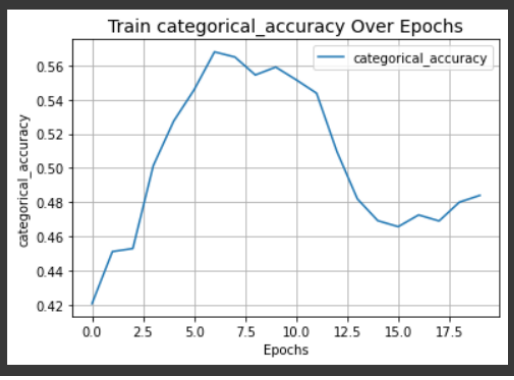


از آن جایی که از دیتای crawl شده و تهیه شده توسط خودمان در فاز های قبلی استفاده کردیم، و همینطور به دلیل اینکه کلاس های کتگوری برای این پروژه زیاد هستند و دیتای ما محدود، به طبع انتظار زیادی از این مدل در Evaluation نداریم. با این حال، این مدل حدود 35 درصد accuary به ما میدهد:

```
[23] plt.plot(history.history['loss'], label='loss')
     plt.xlabel("Epochs")
     plt.ylabel('loss')
     plt.title("Train {} Over Epochs".format('loss'), fontsize=14)
     plt.legend()
     plt.grid()
     plt.show()
```




```
[24] plt.plot(history.history['categorical_accuracy'], label='categorical_accuracy')
     plt.xlabel("Epochs")
     plt.ylabel('categorical_accuracy')
     plt.title("Train {} Over Epochs".format('categorical_accuracy'), fontsize=14)
     plt.legend()
     plt.grid()
     plt.show()
```




```
_, categorical_acc = model.evaluate(test_dataset)
print(f"Categorical accuracy on the test set: {round(categorical_acc * 100, 2)}%.")
```

```
49/49 [==============================] - 1s 10ms/step - loss: 0.7119 - categorical_accuracy: 0.3571
Categorical accuracy on the test set: 35.71%.
```

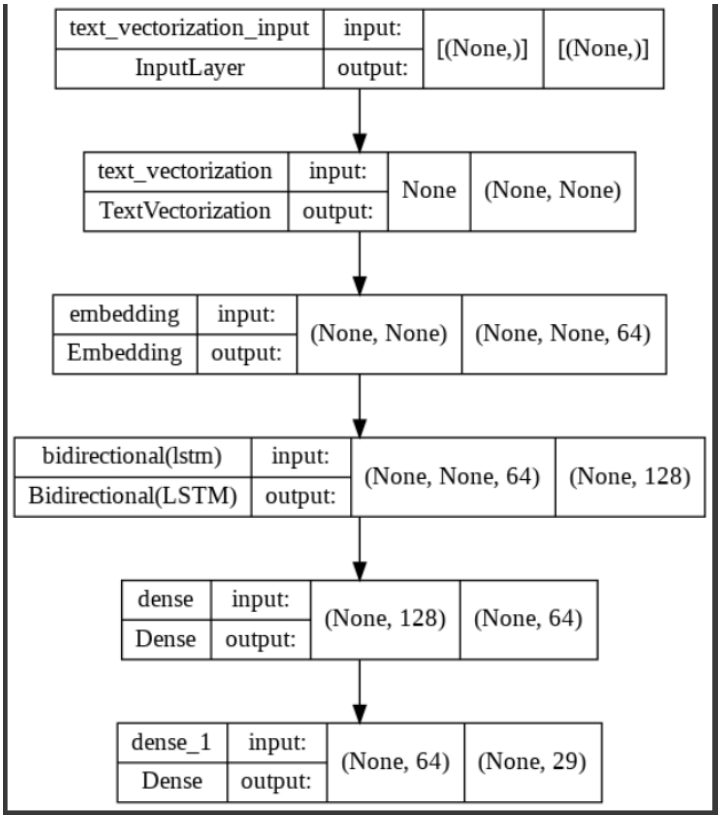
کد های این بخش در فایل Part2 قرار دارند.

بخش سوم:

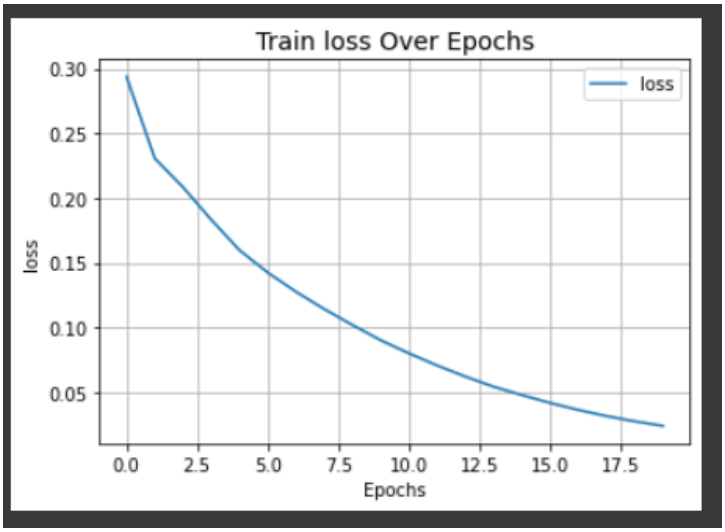
برای بهبود مدل طراحی شده، دو راه پیاده سازی را پیش گرفتیم. استفاده از LSTM، و استفاده از CNN.

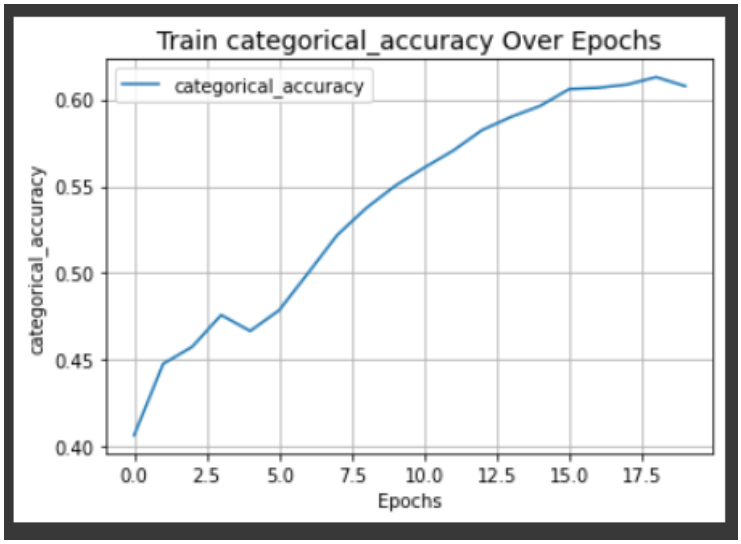
روش LSTM:

```
[15] print(model.summary())

Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 text_vectorization (TextVec  (None, None)             0
 torization)

 embedding (Embedding)       (None, None, 64)          1966400

 bidirectional (Bidirectiona  (None, 128)              66048
 l)

 dense (Dense)               (None, 64)                8256

 dense_1 (Dense)             (None, 29)                1885

=================================================================
Total params: 2,042,589
Trainable params: 2,042,589
Non-trainable params: 0
_________________________________________________________________
None
```



نتایج بدست آمده از مدل طراحی شده با lstm:






```
loss_metric, categorical_acc, f1_metric, precision_metr
print(f"Categorical accuracy on the test set: {round(ca
print(f"f1 metric on the test set: {f1_metric}.")
print(f"precision on the test set: {precision_metric}."
print(f"recall on the test set: {recall_metric}.")
```

```
49/49 [==============================] - 3s 8ms/step -
Categorical accuracy on the test set: 34.85%.
f1 metric on the test set: 0.46216341853141785.
precision on the test set: 0.48089316487312317.
recall on the test set: 0.4451489746570587.
```

همانطور که مشخص است همچنان پیشرفت خاصی بر روی نتایج مدل مشاهده نمیکنیم.

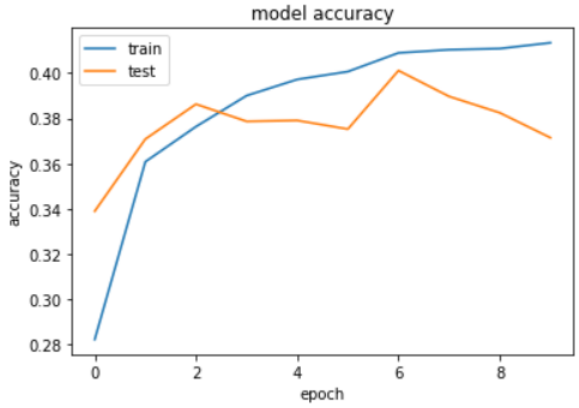
فایل گوگل کولب برای روش lstm:

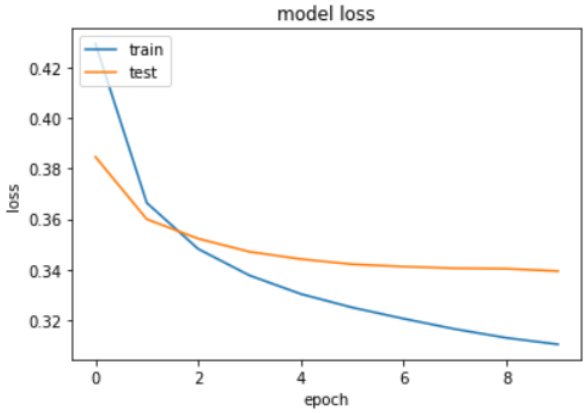
https://colab.research.google.com/drive/1Uq-vcH1fv-ePdpEeW-T-xkdhTIN6OiQR?usp=sharing

روش CNN:

در این روش ابتدا ابتدا لایه Embedding را به یک لایه Conv1D وارد می کنیم و سپس از Average Pooling و Max Pooling همزمان استفاده می کنیم و این ۲ را در انتها به هم متصل می کنیم. نتیجه آخرین لایه concatenate (None, 13) است که به لایه خروجی داده می شود.

```
Model: "model_2"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_3 (InputLayer)            [(None, 200)]        0
__________________________________________________________________________________________________
embedding_2 (Embedding)         (None, 200, 100)     3727600     input_3[0][0]
__________________________________________________________________________________________________
conv1d_2 (Conv1D)               (None, 198, 64)      19264       embedding_2[0][0]
__________________________________________________________________________________________________
global_average_pooling1d_2 (Glo (None, 64)           0           conv1d_2[0][0]
__________________________________________________________________________________________________
global_max_pooling1d_2 (GlobalM (None, 64)           0           conv1d_2[0][0]
__________________________________________________________________________________________________
concatenate_2 (Concatenate)     (None, 128)          0           global_average_pooling1d_2[0][0]
                                                                 global_max_pooling1d_2[0][0]
__________________________________________________________________________________________________
dense_2 (Dense)                 (None, 13)           1677        concatenate_2[0][0]
==================================================================================================
Total params: 3,748,541
Trainable params: 20,941
Non-trainable params: 3,727,600
__________________________________________________________________________________________________
```






```
195/195 [==============================] - 0s 2ms/step - loss: 0.3404 - acc: 0.3640
Test Score: 0.34044456481933594
Test Accuracy: 0.3640301525592804
```

باز هم پیشرفت خوبی در نتایج حاصل نمیشود.

دلیل اصلی این موضوع زیاد بودن کتگوری های ژانر های فیلم ها و محدود بودن دیتا در این زمینه میباشد.

چیزی حدود حداقل ۱۳ تا ژانر متفاوت در دیتا وجود دارد و هر فیلم حداقل ۲ یا ۳ ژانر دارد و طبیعی است که مدل ما به دیتای بیشتری در این زمینه احتیاج داشته باشد.

کد های این بخش در فایل Part3 قرار دارند.