به نام خدا

دانشگاه تهران

پردیس دانشکده‌های فنی

دانشکده برق و کامپیوتر

پـزدازش زبـان هـای طبـیـعـی

CA5

فاطمه سلیقه

۸۱۰۱۹۸۳۰۶

اردیبهشت ماه ۱۳۹۹

سوال ۱ تشخیص اسپم

برای اجرای این بخش از google colab استفاده شده و روی GPU اجرا شده است .

در ابتدا داده ها را از روی فایل spam.csv خوانده که به صورت زیر است :

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
575	spam	You have won ?1,000 cash or a ?2,000 prize! To...	NaN	NaN	NaN
1221	ham	Prakesh is there know.	NaN	NaN	NaN
3044	ham	Hello, yeah i've just got out of the bath and ...	NaN	NaN	NaN
2849	ham	She's fine. Good to hear from you. How are you...	NaN	NaN	NaN

سپس spam و ham را به map 0,1 می کنیم .

داده ها را به دو بخش train و test تقسیم می کنیم .

استفاده از bert

حال جملات و label های train و test را در ارایه های جداگانه قرار می دهیم .
سپس لازم است تا داد ها را tokenize کنیم به همین منظور از تابع BertTokenizer.from_pretrained استفاده می کنیم و با توجه به لینک مدل pretrained برای bert از مدل `'bert-large-cased'` استفاده می کنیم .

```
tokenizer = BertTokenizer.from_pretrained('bert-large-cased', do_lower_case=True)
```

حال با استفاده از تابع encode_plus موارد زیر را برای تک تک جملات انجام می دهیم :

۱. اضافه نمودن [CLS] و [SEP] به جملات
۲. تمام جملات باید دارای ماکزیمم اندازه ۱۲۸ باشند بنابراین عمل padding را برای جملات کوتاهتر و truncate را برای جملات بلندتر انجام می دهیم .
۳. تمام جملات را به صورت کلمه tokenize می کنیم .
۴. تمام جملات را encode می کنیم .

```
tokenizer.encode_plus(
      sent,
      add_special_tokens = True,
      max_length = 128,
      pad_to_max_length = True,
      return_attention_mask = True,
      return_tensors = 'pt',
)
```

حال یک لایه از bert را تعریف می کنیم . که در آن ابتداhub ، bert را به صورت زیر تعریف می کند :

```
self.bert = hub.Module(
    'https://tfhub.dev/google/bert_cased_L-12_H-768_A-12/1',
    trainable=self.trainable,
    name="{}_module".format(self.name)
)
```

سپس ماژول bert از روش زیر داده ها را دریافت می کند و خروجی مورد نظر را می دهد :

```
inputs = [K.cast(x, dtype="int32") for x in inputs]
input_ids, input_mask, segment_ids = inputs
bert_inputs = dict(input_ids=input_ids, input_mask=input_mask, segment_ids=segment_ids)
result = self.bert(inputs=bert_inputs, signature="tokens", as_dict=True)["pooled_output"]
```

همان طور که می بینیم ورودی باید مجموعه ای از input_ids و input_mask و segment_ids باشد .
که با استفاده از تابع tokenizer.encode_plus از جملات ورودی ایجاد می شود .

Bert دو نوع خروجی دارد pooled_output و sequence_output دارد . که pooled_output بازنمایی از کل جمله ورودی است اما sequence_output باز نمایی از تک تک token های جمله ورودی می باشد .

که ما در این مسئله چون می خواهیم جملات را کلاسبندی کنیم از pooled_output استفاده می کنیم .

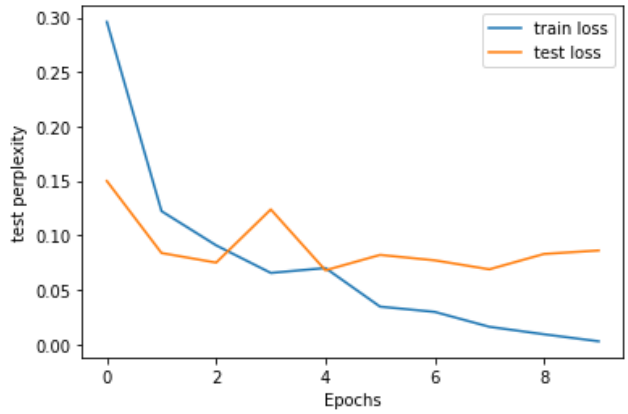
شبکه به صورت زیر طراحی شده است :

```
bert_output = BertLayer(n_fine_tune_layers=5)(bert_inputs)
dense = tf.keras.layers.Dense(768, activation='relu')(bert_output)
pred = tf.keras.layers.Dense(1, activation='sigmoid')(dense)
```

الف)

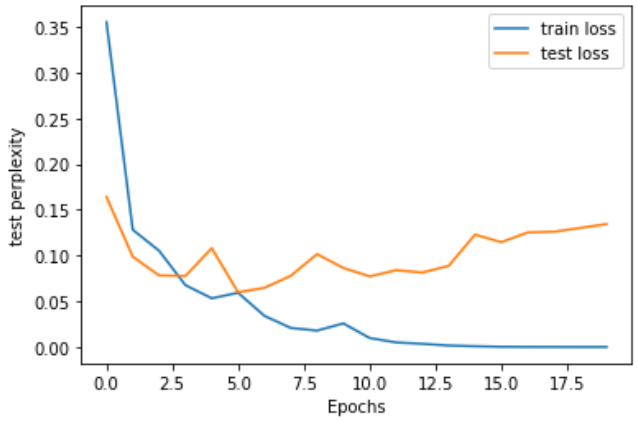
۱۰ ایپاک :

```
AUC :  0.9425278515687401
accuracy :  0.9791304347826087
precision :  0.9685534591194969
recall :  0.8901734104046243
F1 :  0.927710843373494
<matplotlib.legend.Legend at 0x7f29aae91fd0>
```



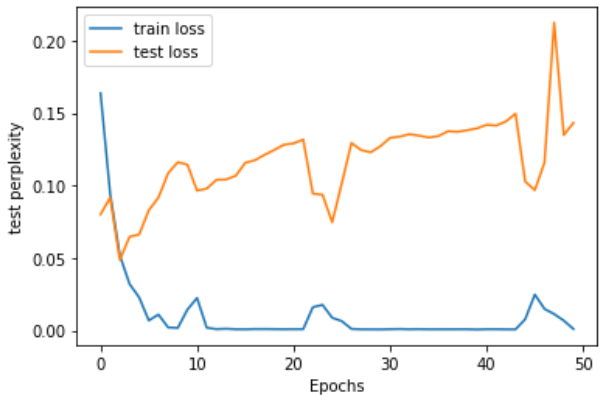
۲۰ ایپاک :

```
AUC :  0.9454180249791445
accuracy :  0.98
precision :  0.96875
recall :  0.8959537572254336
F1 :  0.9309309309309309
<matplotlib.legend.Legend at 0x7f29a3e5e2e8>
```



۵۰ ایپاک :

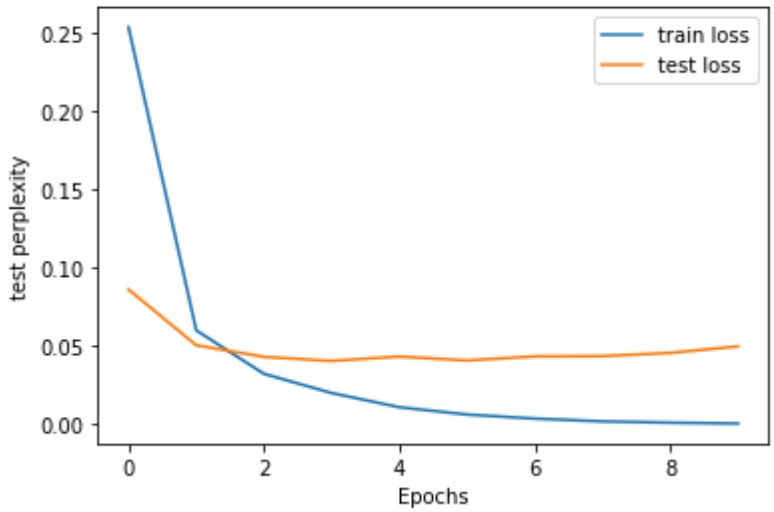
```
AUC :  0.9525258799171842
accuracy :  0.9811827956989247
precision :  0.9448275862068966
recall :  0.9133333333333333
F1 :  0.9288135593220338
<matplotlib.legend.Legend at 0x7f39857f32b0>
```



خیر تعداد تکرار بیشتر تاثیر زیادی در بهبود دقت طبقه بند ندارد . تنها ۱ درصد ممکن است تغییر ایجاد کند .

ب) بدون اعمال پیش پردازش های لازم :

```
AUC :  0.9659511077611532
accuracy :  0.9879931389365352
precision :  0.9813664596273292
recall :  0.9349112426035503
F1 :  0.9575757575757575
<matplotlib.legend.Legend at 0x7f2410f75d68>
```



عدم استفاده از پیش پردازش عملکرد را بهتر می کند . استفاده از پیش پردازش باعث باعث ایجاد نوسان در loss می شود . اما عدم استفاده از پیش پردازش باعث کاهش منظم loss شده و دقت و سایر شاخص ها را بهتر می کند . به نظرم علت آن می تواند باشد که همون طور که در درس گفته شده بین کلمات ورودی و خروجی رابطه وجود دارد و حتی حذف stop word ها هم می تواند مفهوم جمله را تغییر دهد .

استفاده از elmo

برای کار با ماژول elmoبه صورت زیر این ماژول را تعریف می کنیم :

```
url = "https://tfhub.dev/google/elmo/2"
embed = hub.Module(url)
```

حال برای استفاده از این ماژول به صورت زیر عمل می کنیم

```
embed(tf.squeeze(tf.cast(x, tf.string)),
      signature="default",
      as_dict=True)["default"]
```

که داده ها را به string تبدیل می کند و چون signature از نوع "default" است ، ورودی را بره صورت یک sequence دریافت می کند اما اگر signature از نوع "token" باشد وردی را به صورت tokenize شده دریافت می کند .

این مازول ورودی های شبکه را دریافت می کند و خروجی های embed شده با اندازه ۱۰۲۴ می دهد .

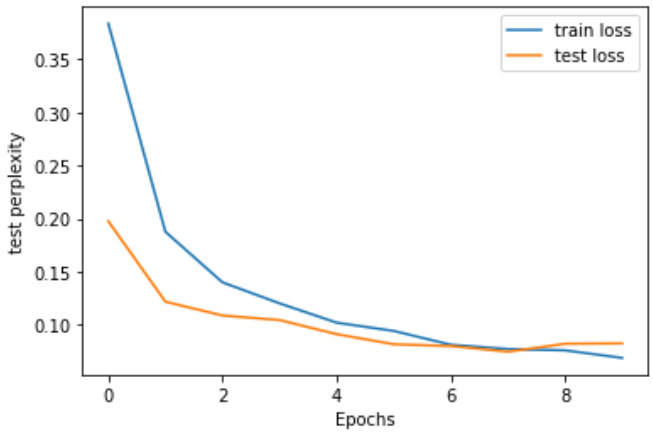
سپس این خروجی ها را به عنوان ورودی به یک لایه feedforward می دهیم که به صورت زیر تعریف شده است :

```
input_text = Input(shape=(1,), dtype=tf.string)
embedding = Lambda(ELMoEmbedding, output_shape=(1024, ))(input_text)
dense = Dense(1024, activation='relu')(embedding)
pred = Dense(2, activation='softmax')(dense)
model = Model(inputs=[input_text], outputs=pred)
ada = keras.optimizers.Adam(learning_rate=0.0002, beta_1=0.9, beta_2=0.999, amsgrad=False)
model.compile(loss='categorical_crossentropy', optimizer=ada, metrics=['accuracy'])
```

الف)

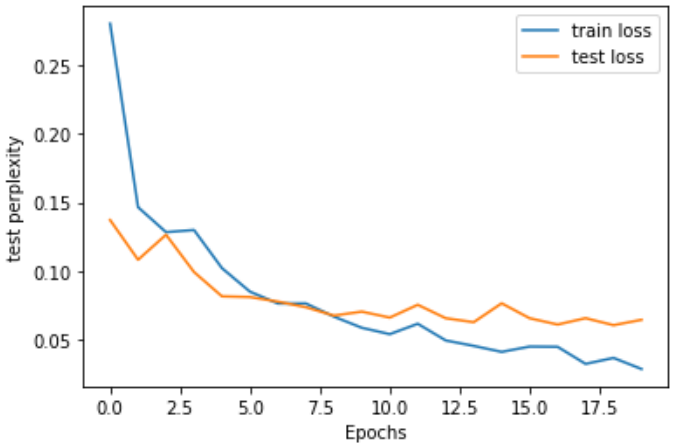
۱۰ ایپاک :

```
AUC :  0.9593407915313364
accuracy :  0.9725906277630415
precision :  0.8674698795180723
recall :  0.9411764705882353
F1 :  0.9028213166144202
<matplotlib.legend.Legend at 0x7f3e6fab2eb8>
```



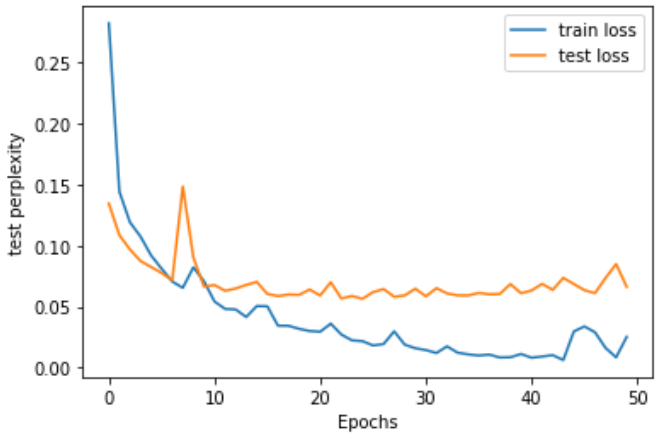
۲۰ ایپاک :

```
AUC :  0.9576065600064156
accuracy :  0.9743589743589743
precision :  0.8827160493827161
recall :  0.934640522875817
F1 :  0.9079365079365078
<matplotlib.legend.Legend at 0x7f3e6e646c18>
```



۵۰ ایپاک :

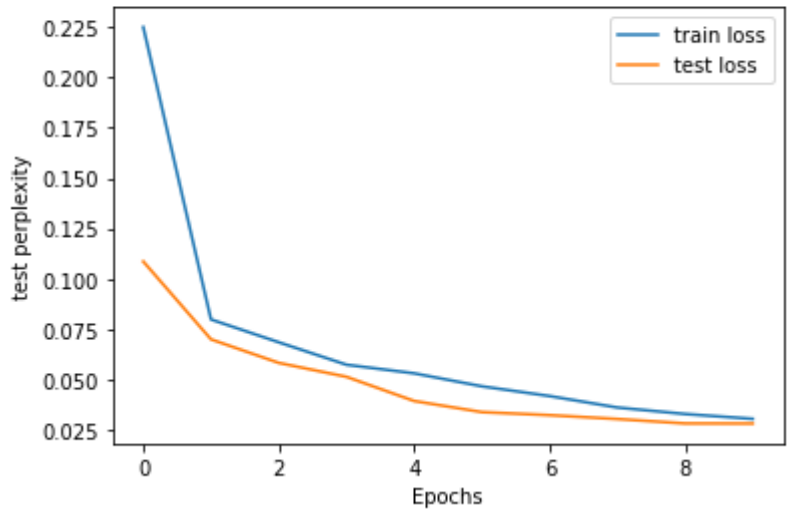
```
AUC :  0.9489354023818115
accuracy :  0.9832007073386384
precision :  0.971830985915493
recall :  0.9019607843137255
F1 :  0.9355932203389831
<matplotlib.legend.Legend at 0x7f3e6e1c7b70>
```



خیر تعداد تکرار بیشتر تاثیر زیادی در بهبود دقت طبقه بند ندارد . تنها ۱ درصد ممکن است تغییر ایجاد کند .

ب) بدون اعمال پیش پردازش های لازم :

```
AUC :  0.977034259823534
accuracy :  0.9906779661016949
precision :  0.9655172413793104
recall :  0.958904109589041
F1 :  0.9621993127147767
<matplotlib.legend.Legend at 0x7f3e6de72fd0>
```



عدم استفاده از پیش پردازش عملکرد را بهتر می کند . استفاده از پیش پردازش باعث باعث ایجاد نوسان در loss می شود . اما عدم استفاده از پیش پردازش باعث کاهش منظم loss شده و دقت و سایر شاخص ها را بهتر می کند . به نظرم علت آن می تواند باشد که همون طور که در درس گفته شده بین کلمات ورودی و خروجی رابطه وجود دارد و حتی حذف stop word ها هم می تواند مفهوم جمله را تغییر دهد .

ج)

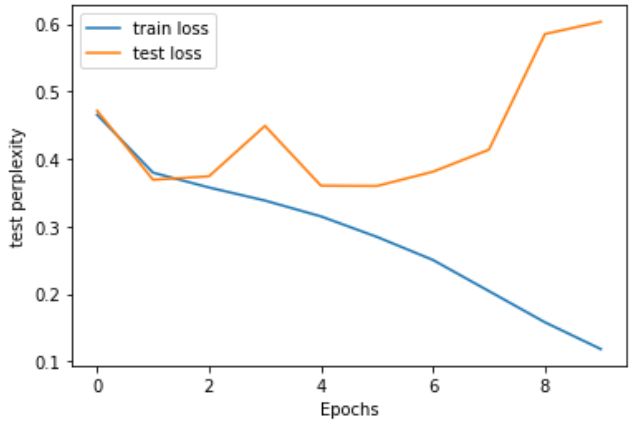
در تشخیص اسپم precidion مهم تر از recall است . در واقع در معیار precision می گوییم که چه تعداد از مواردی را که اسپم تشخیص داده ایم واقعا اسپم هستند و در recall می گوییم که چه تعداد از آن هایی که واقعا اسپم بوده اند را تشخیص داده ایم . بنابراین معیار precision مهم تر است زیرا اگر تعداد زیادی پیام هایی را که اسپم نیستند اسپم تشخیص دهیم ممکن است پیام مهمی باشند و اعتماد کاربر از بین برود .

سوال ۲ تحلیل احساس

استفاده از bert

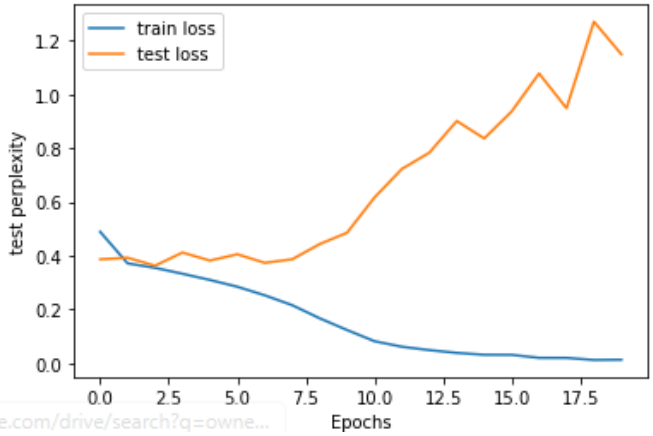
۱۰ ایپاک :

```
AUC :  0.8200460829493088
accuracy :  0.8198
precision :  0.8431538787621874
recall :  0.7892857142857143
F1 :  0.8153310104529617
<matplotlib.legend.Legend at 0x7f2591195320>
```



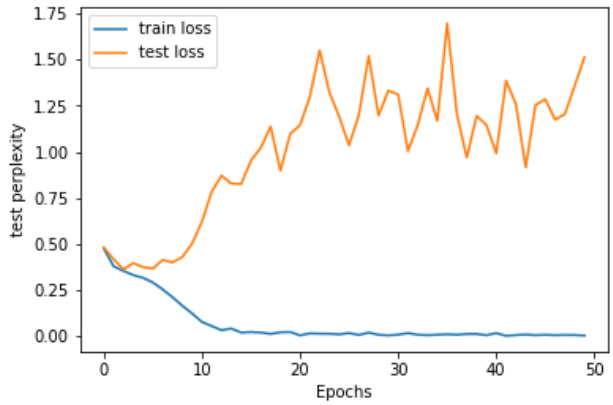
۲۰ ایپاک :

```
AUC :  0.8230926779313876
accuracy :  0.8234
precision :  0.8025878003696858
recall :  0.8615079365079366
F1 :  0.8310047846889952
<matplotlib.legend.Legend at 0x7f25923d2400>
```



۵۰ ایپاک :

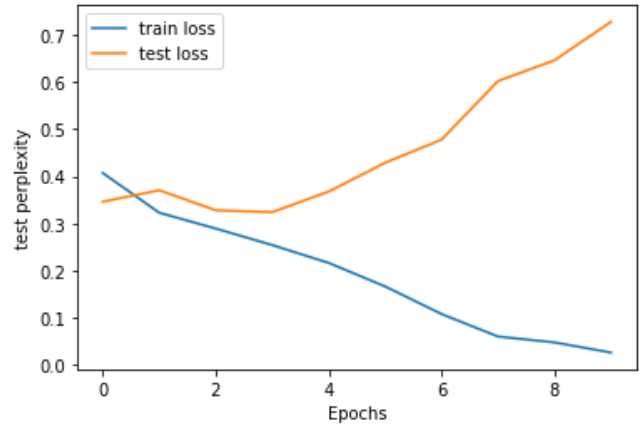
```
AUC :  0.8114375320020483
accuracy :  0.812
precision :  0.7758379888268156
recall :  0.8817460317460317
F1 :  0.825408618127786
<matplotlib.legend.Legend at 0x7f259442b390>
```



خیر تعداد تکرار بیشتر تاثیر زیادی در بهبود دقت طبقه بند ندارد . تنها ۱ درصد ممکن است تغییر ایجاد کند .

ب) بدون انجام پیش پردازش های لازم:

```
AUC :  0.8469784785960367
accuracy :  0.847
precision :  0.8415064731267163
recall :  0.8559457302474063
F1 :  0.8486646884272997
<matplotlib.legend.Legend at 0x7f2585f1fa58>
```



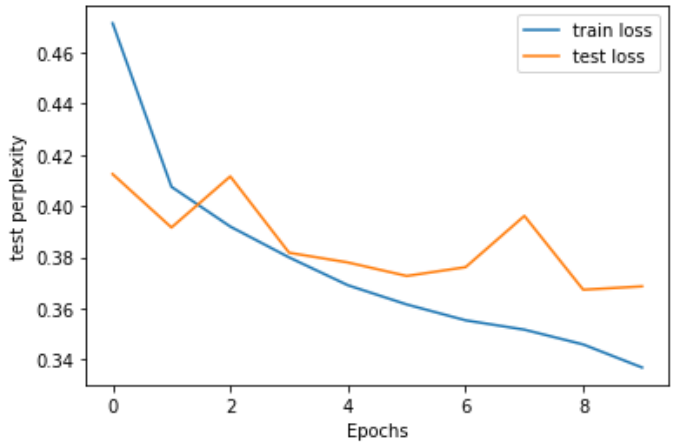
عدم استفاده از پیش پردازش عملکرد را بهتر می کند . استفاده از پیش پردازش باعث باعث ایجاد نوسان در loss می شود . اما عدم استفاده از پیش پردازش باعث کاهش منظم loss شده و دقت و سایر شاخص ها را بهتر می کند . به نظرم علت آن می تواند باشد که همون طور که در درس گفته شده بین

کلمات ورودی و خروجی رابطه وجود دارد و حتی حذف stop word ها هم می تواند مفهوم جمله را تغییر دهد .

استفاده از elmo

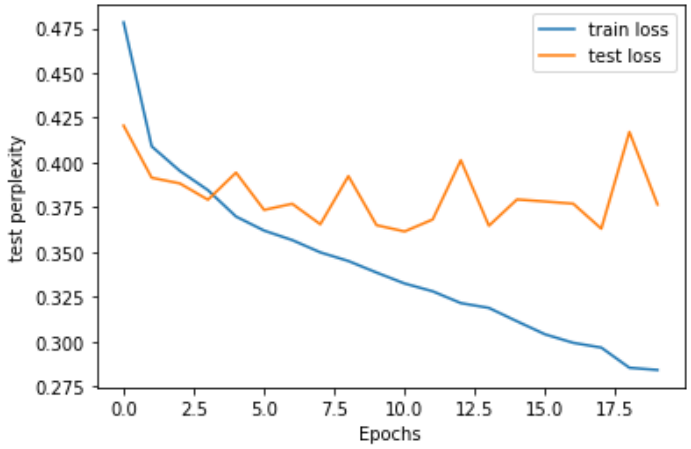
۱۰ ایپاک:

```
AUC :  0.8336172494695475
accuracy :  0.834
precision :  0.8192725909261342
recall :  0.8626135017765495
F1 :  0.8403846153846154
<matplotlib.legend.Legend at 0x7f20746ee2e8>
```



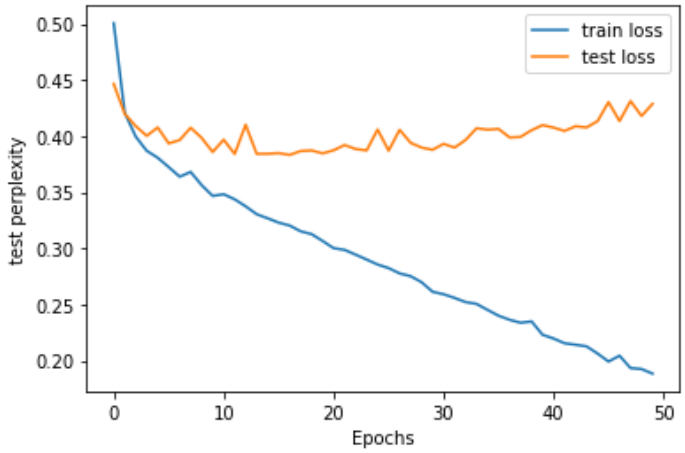
۲۰ ایپاک :

```
AUC :  0.8363159596928168
accuracy :  0.8356
precision :  0.8800533096401599
recall :  0.7820765890248716
F1 :  0.8281772575250836
<matplotlib.legend.Legend at 0x7f2071b573c8>
```



۵۰ ایپاک:

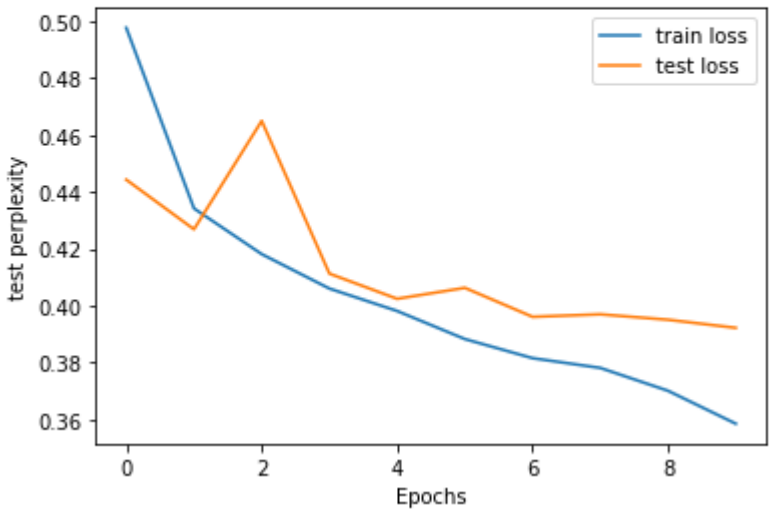
```
AUC :  0.8214115002322757
accuracy :  0.821
precision :  0.8474936278674596
recall :  0.7882259976293955
F1 :  0.8167860798362333
<matplotlib.legend.Legend at 0x7f7faaa2d128>
```



خیر استفاده از تعداد ایپاک بیشتر دقت را افزایش نمی دهد .

ب) بدون انجام پیش پردازش های لازم :

```
AUC :  0.812758613406519
accuracy :  0.8128
precision :  0.8142276422764227
recall :  0.8070104754230459
F1 :  0.810602994738972
<matplotlib.legend.Legend at 0x7f969efdc898>
```



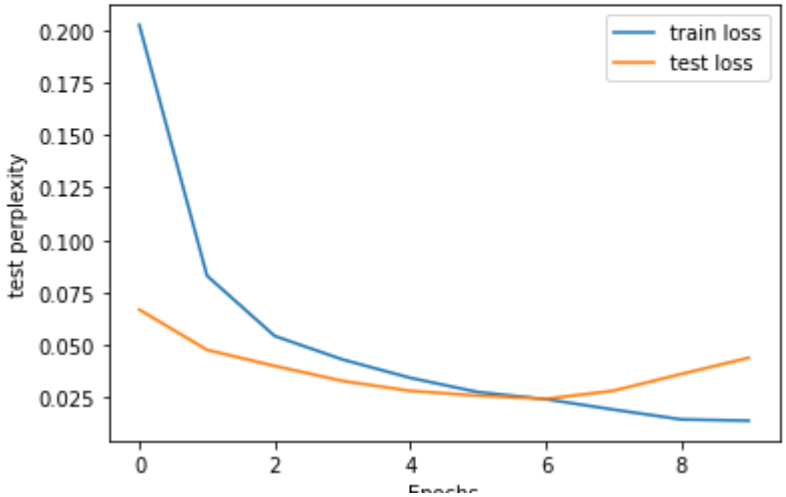
سوال ۳ امتیازی

سوال دوم :

تشخیص اسپم با elmo

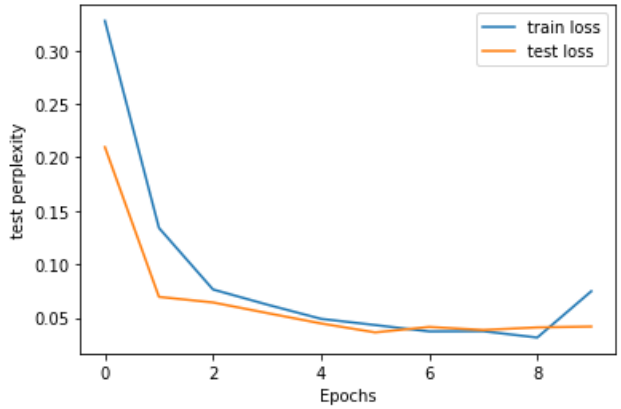
با دولایه :

```
AUC :  0.977034259823534
accuracy :  0.9906779661016949
precision :  0.9655172413793104
recall :  0.958904109589041
F1 :  0.9621993127147767
<matplotlib.legend.Legend at 0x7f3e6ce7da90>
```



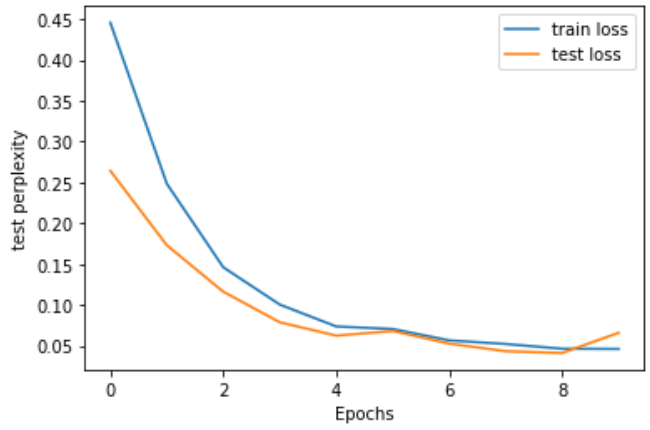
با پنج لایه :

```
AUC :  0.9770740043983996
accuracy :  0.985593220338983
precision :  0.9215686274509803
recall :  0.9657534246575342
F1 :  0.94314381270903
<matplotlib.legend.Legend at 0x7f3e6cdcf978>
```



با ۷ لایه :

```
AUC :  0.9727219734506241
accuracy :  0.9779661016949153
precision :  0.8703703703703703
recall :  0.9657534246575342
F1 :  0.9155844155844156
<matplotlib.legend.Legend at 0x7f3e6c0c4ef0>
```

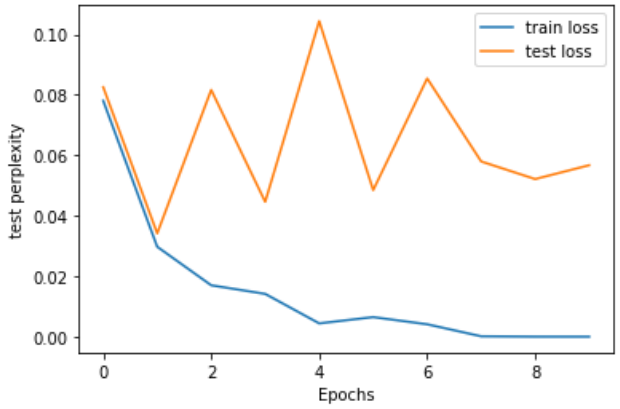


اگر تعداد لایه ها را از یک به دو افزایش دهیم . تماممعیار ها افزایش پیدا کرده و بهتر عمل می شود اما با افزایش بیشتر تعداد لایه ها نسبت به دو لایه تفاوت زیادی ندارد . حتی ممکن است بدتر عمل شود .

تشخیص اسپم با استفاده از bert :

با دو لایه :

```
AUC :  0.9758298564332049
accuracy :  0.9922813036020584
precision :  0.9938271604938271
recall :  0.9526627218934911
F1 :  0.972809667673716
<matplotlib.legend.Legend at 0x7f2030588b70>
```

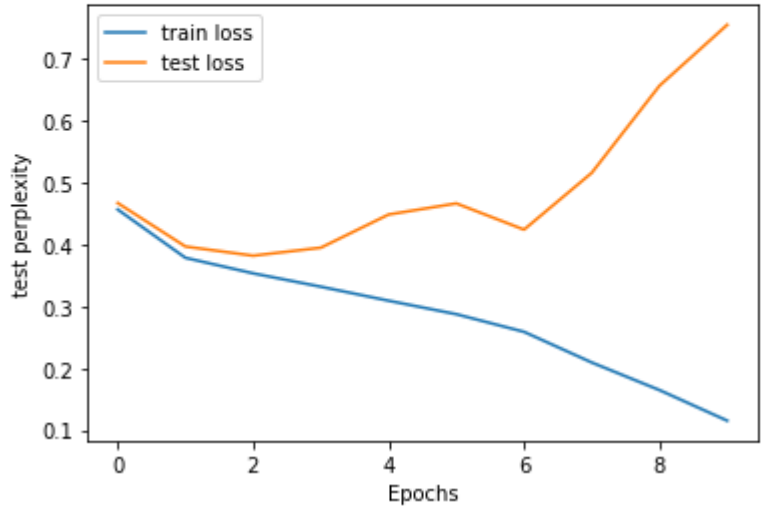


با ۵ و ۷ لایه امکان اجرا وجود نداشت و با خطای حافظه مواجه می شد .

تحلیل احساس با استفاده از bert

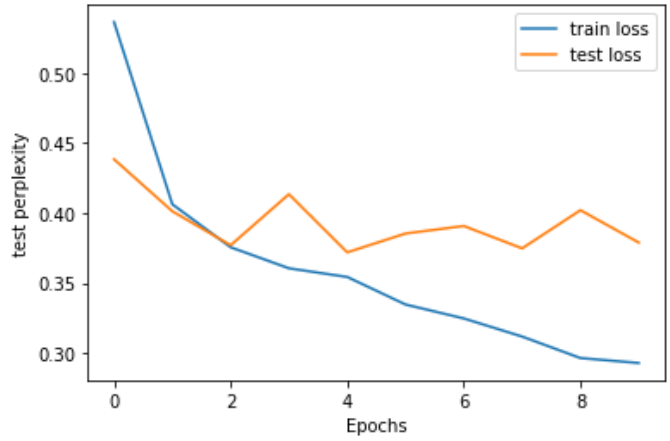
با دو لایه :

```
AUC :  0.7666064741177644
accuracy :  0.7716
precision :  0.7091015510681885
recall :  0.9424348502528199
F1 :  0.8092852371409485
<matplotlib.legend.Legend at 0x7f258be42cc0>
```



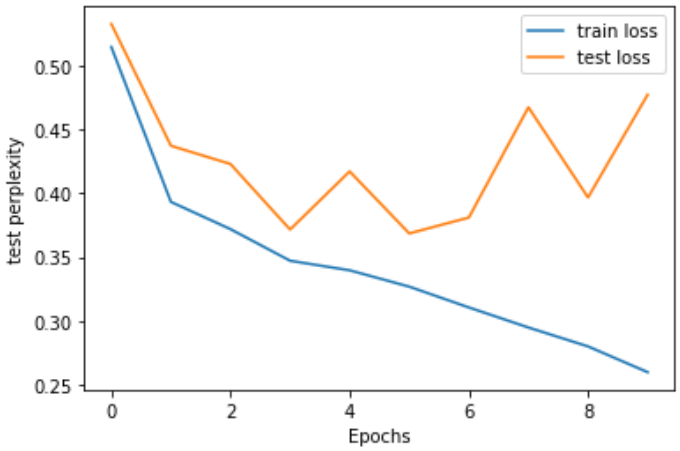
با ۵ لایه :

```
AUC :  0.828522441017449
accuracy :  0.8284
precision :  0.784375
recall :  0.9050480769230769
F1 :  0.8404017857142858
<matplotlib.legend.Legend at 0x7f40ac642160>
```



با ۷ لایه:

```
AUC :  0.7948557948308349
accuracy :  0.7946
precision :  0.7227782832878374
recall :  0.9547275641025641
F1 :  0.8227170723286725
<matplotlib.legend.Legend at 0x7f40ac680278>
```



نتیجه ای مشاهده شده است این است که افزایش تعداد لایه ها لزوما باعث افزایش دقت و بهتر شدن شبکه نمی شود .