دانشگاه تهران
پردیس دانشکده‌های فنی
دانشکده برق و کامپیوتر

گزارش تمرین شماره ۴

درس NLP

بهار ۱۴۰۲

نام و نام خانوادگی
سپهر کریمی آرپناهی

شماره دانشجویی
۸۱۰۱۰۰۴۴۷

سوال ۱

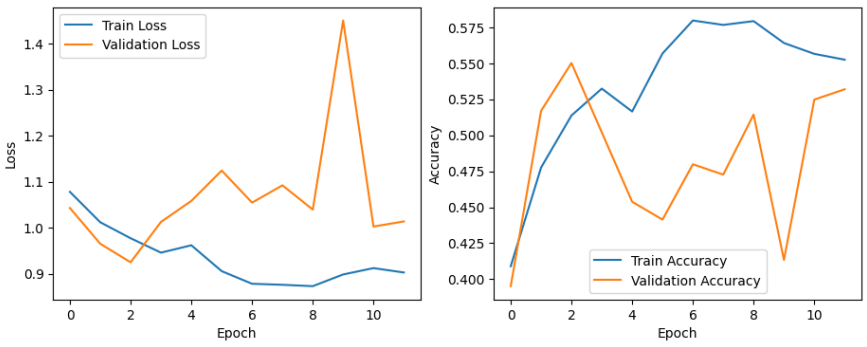
- **مرحله دادگان و پیش پردازش:**

در تابع پیش پردازش تعریف شده، علائم اضافی و stop word ها را حذف می کنیم. همچنین با توجه به اینکه در این دیتاست کلمات از فینگلیش و لینک و علائم مختلف وجود دارد، لینک ها و هشتگ ها و کلماتی که به یادگیری مدل زبانی کمک نمیکردند را حذف کردیم. همچنین کمک کتابخانهhazm ، کلمات را و کل دیتاست را نورمالایز می کنیم.

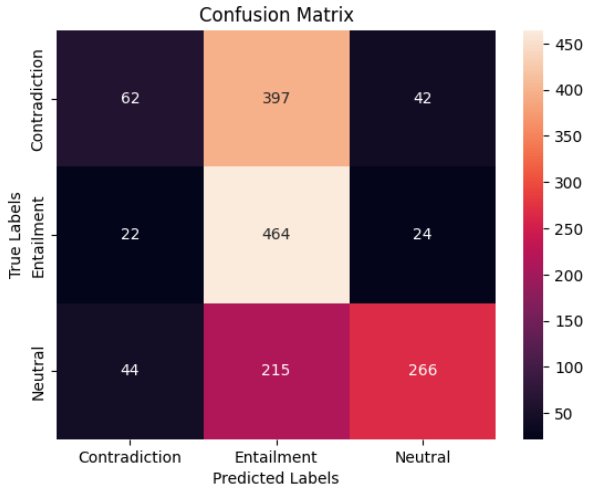
در انتها نیز لیبل ها را به عدد تبدیل می کنیم.

- **وظیفه اول**

در این قسمت می خواهیم از مدل ParsBERT به عنوان ورودی استفاده کرده و شبکه ای را با استفاده از transformerEncoder طراحی کنیم. برای این کار یک مدل طراحی می کنیم(ParsBert_task۱) که در لایه اول از خود مدل ParsBert به عنوان بازنمایی استفاده کرده و بعد از آن یک لایه ترنسفورمر(TransformerEncoderLayer) اضافه کرده و در انتها از یک طبقه بند خطی عبور می دهیم. که لایه آخر خروجی logits را به ما خواهد داد.
همچنین در این تسک همانطور که در صورت سوال گفته شده وزن های مدل برت را فریز کردیم و فقط وزن های لایه های بعدی آموزش داده می شوند.
این تمرین با پایتورچ پیاده سازی شده است به همین علت ما کلاس ParsBert_Dataset_pt را تعریف کردیم تا به وسیله آن دیتاست های ورودی train، validation و test را بسازیم و در انتها دیتالودر های آن ها را ساخته و به عنوان ورودی شبکه به آن بدهیم.
نتایج این مدل به شرح زیر می باشد:



شکل ۱ عملکرد مدل فریز شده



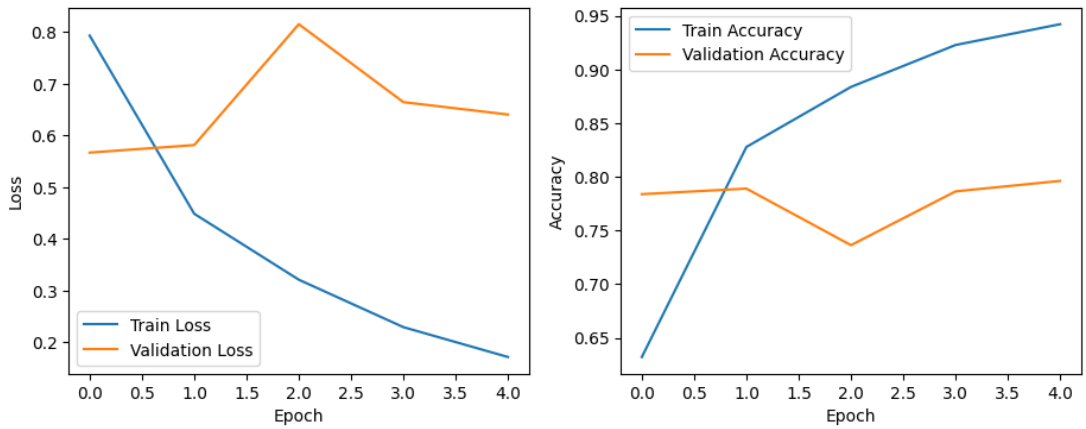
شکل ۲ confusion matrix برای مدل فریز شده

همینطور نتایج دقت این مدل به شرح زیر می باش در این مرحله چون مدل برت را فریز کردیم مدل ما ضعیف تر عمل کرده استد:

```
Test Accuracy for Task 1: 0.612
Test Precision for Task 1: 0.592
Test Recall for Task 1: 0.543
Test F1-score for Task 1: 0.568
```

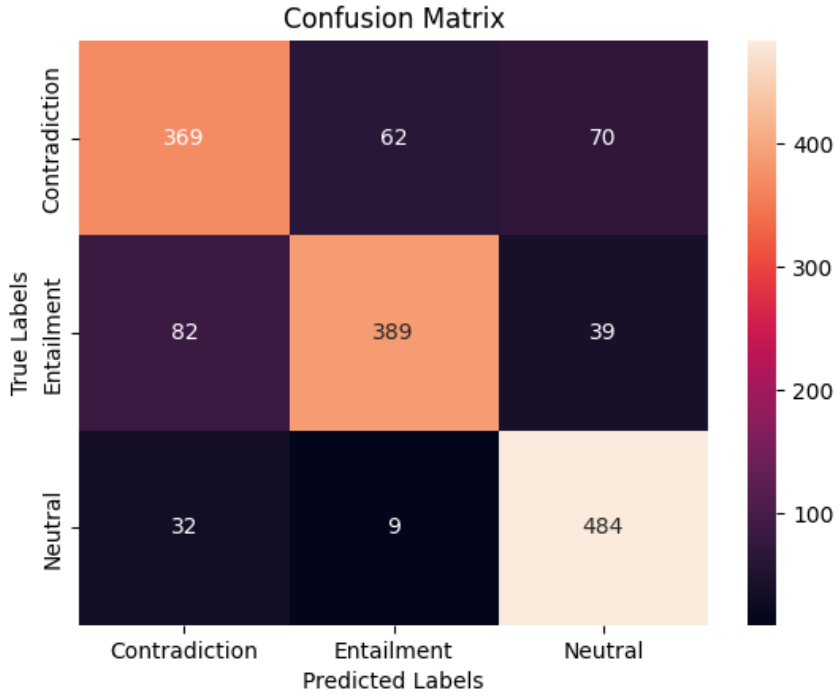
- **وظیفه دوم**

در این قسمت می خواهیم مدل ParsBert را بر روی مجموعه داده FarsTail اموزش دهیم یا شبکه را finetune کنیم. برای این کار مدل PARS_BERT را با استفاده از پایتورچ می سازیم که طراحی لایه های آن به این صورت است که : در ابندا خود مدل برت را قرار می دهیم و با اضافه کردن دو لایه خطی کل مدل را اموزش می دهیم و وزن های مدل را finetune می کنیم(البته ما یک لایه dropout نیز به دلیل آنکه از overfit کردن مدل پیشگیری کنیم اضافه کردیم.
حال نتایج بدست آمده از مدل بالا به شرح زیر خواهد بود:



شکل ۳ : عملکرد مدل دوم

```
Test Accuracy for Task 1: 0.809
Test Precision for Task 1: 0.809
Test Recall for Task 1: 0.807
Test F1-score for Task 1: 0.806
```



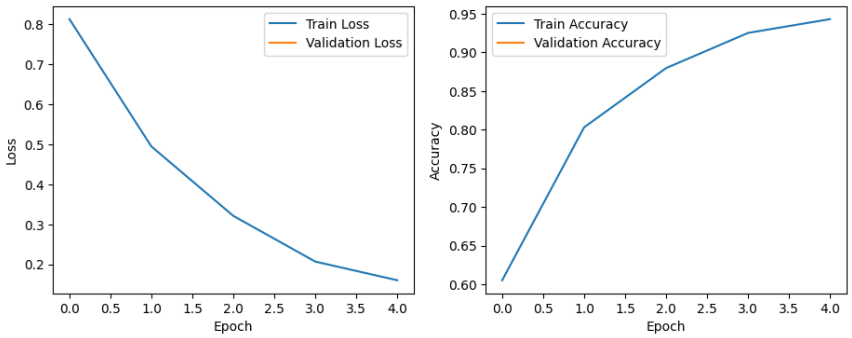
شکل ۴ confusion matrix برای مدل دوم

- **وظیفه سوم**

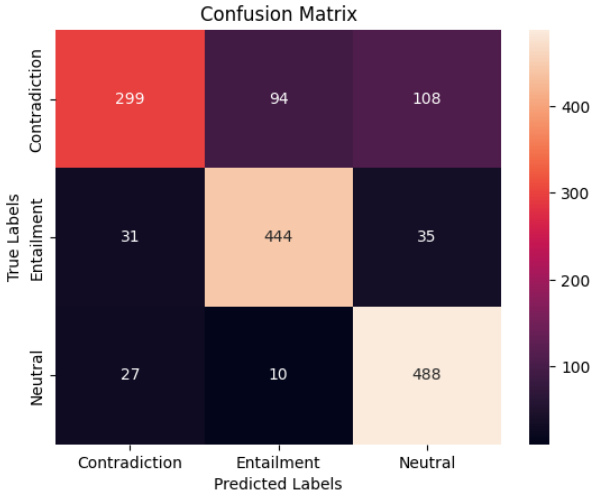
در این قسمت می خواهیم عملکرد لایه های مختلف ParsBERT را بررسی کنیم. برای این کار همانطور که در صورت سوال گفته شده است لایه های مدل را یکی یکی حذف می کنیم. و هر بار به صورت مستقل عمل train را انجام داده و مدل را finetune می کنیم و نتایج هر مرحله را گزارش می کنیم. در اینجا ابتدا به ازای هر کدام تعداد لایه های مختلف bert مشخصات بدست آمده را گزارش می کنیم و در انتها یک نمودار کلی برای آن رسم می کنیم.

  - تعداد لایه = ۱۲ : همان **وظیفه اول** می باشد و برای دیدن نتایج با ۱۲ لایه به وظیفه اول مراجعه شود.

  - تعداد لایه = ۱۱

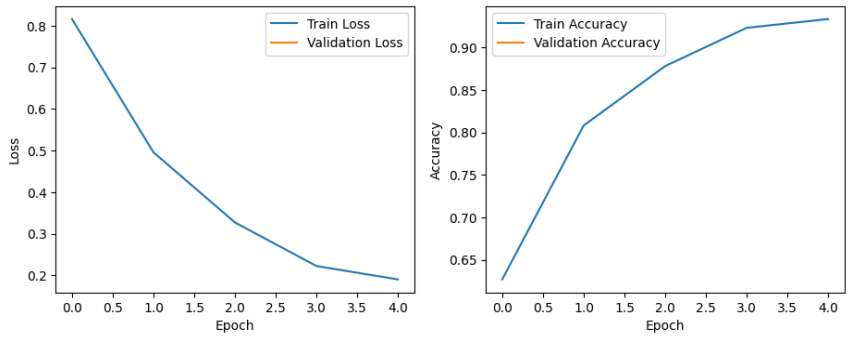


شکل ۵ عملکرد برای مدل سوم با ۱۱ لایه

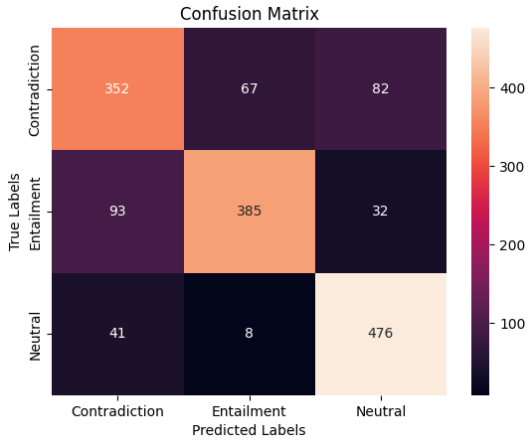


شکل ۶ confusion matrix برای مدل سوم با ۱۱ لایه

- تعداد لایه = ۱۰

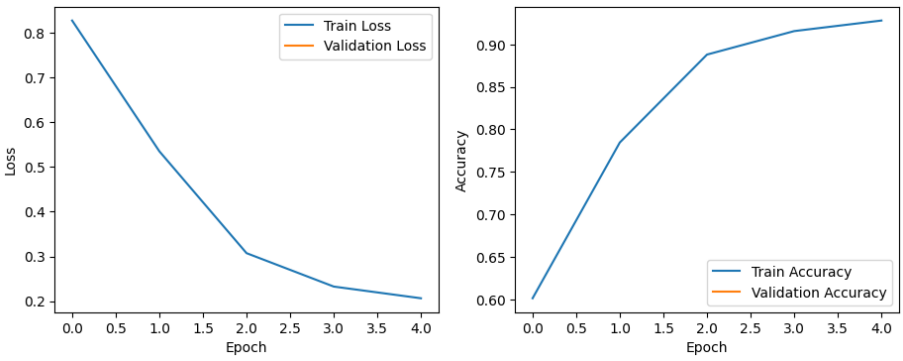


شکل ۷ عملکرد برای مدل سوم با ۱۰ لایه

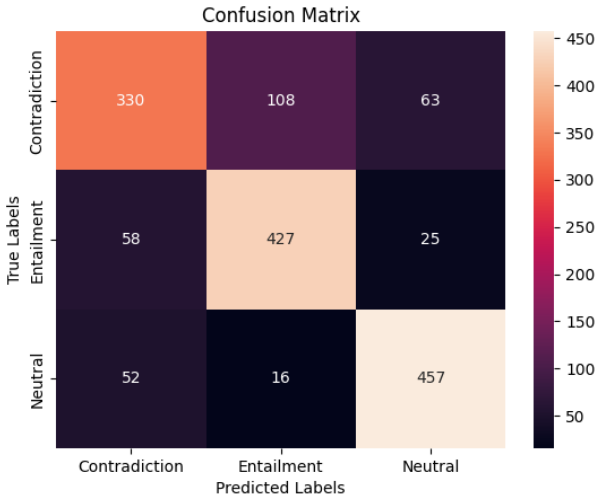


شکل ۸ confusion matrix برای مدل سوم با ۱۰ لایه

- تعداد لایه = ۹

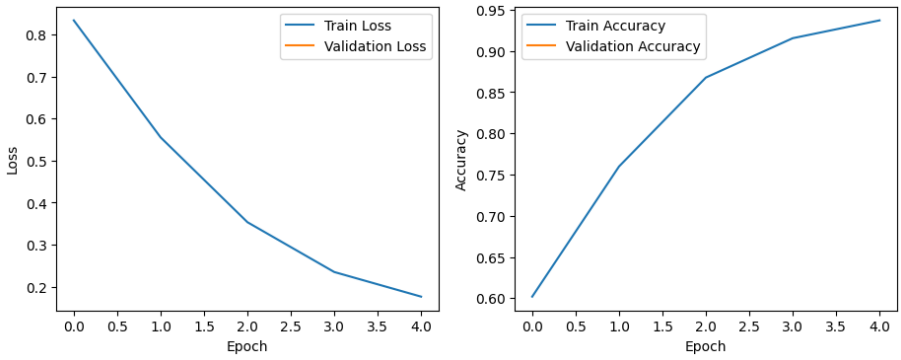


شکل ۹ عملکرد برای مدل سوم با ۹ لایه

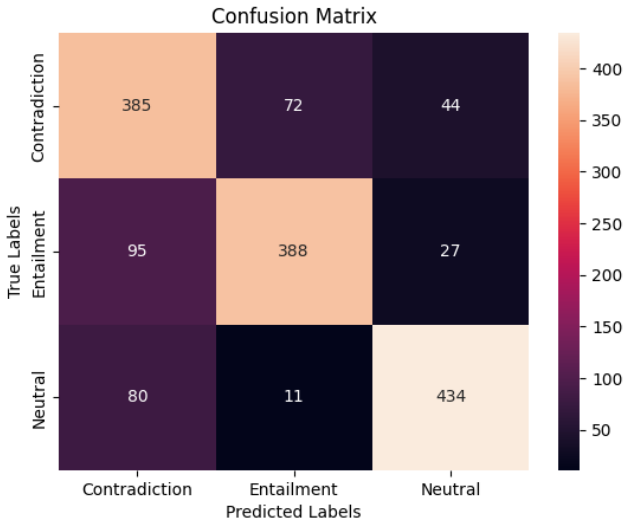


شکل ۱۰ confusion matrix برای مدل سوم با ۹ لایه

- تعداد لایه = ۸

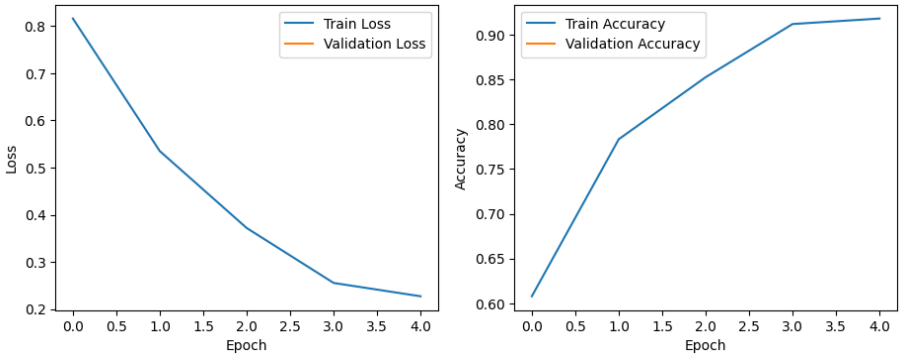


شکل ۱۱ عملکرد برای مدل سوم با ۸ لایه

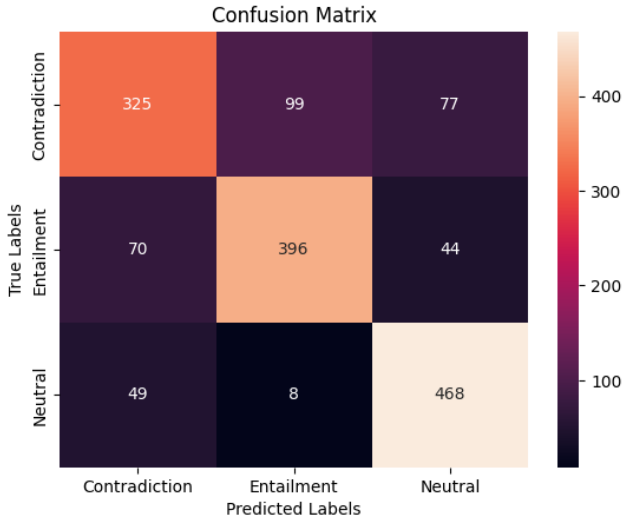


شکل ۱۲ confusion matrix برای مدل سوم با ۸ لایه

- تعداد لایه = ۷

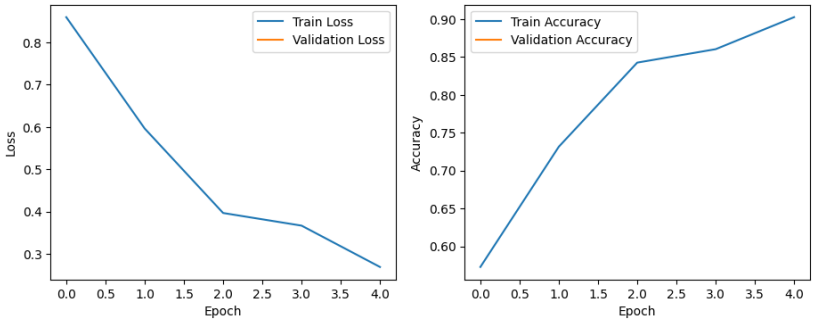


شکل ۱۳ عملکرد برای مدل سوم با ۷ لایه

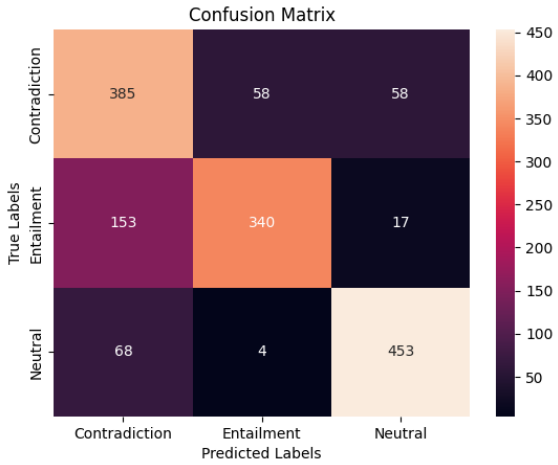


شکل ۱۴ confusion matrix برای مدل سوم با ۷ لایه

- تعداد لایه = ۶

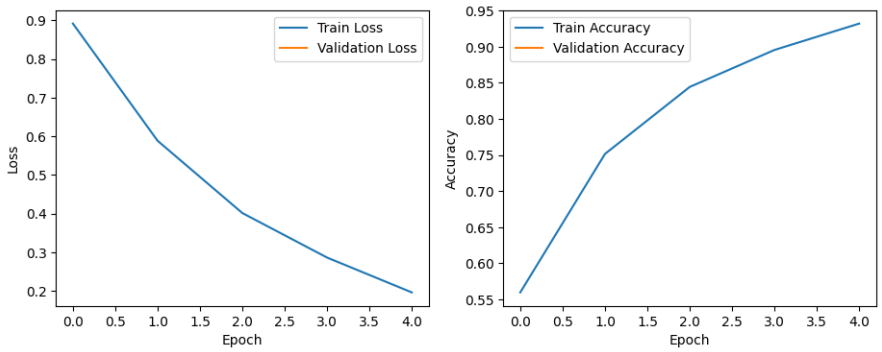


شکل ۱۵ عملکرد برای مدل سوم با ۶ لایه

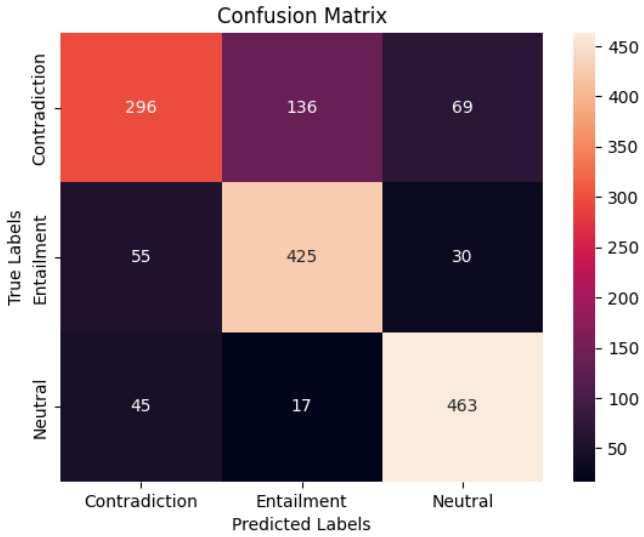


شکل ۱۶ confusion matrix برای مدل سوم با ۶ لایه

- تعداد لایه = ۵

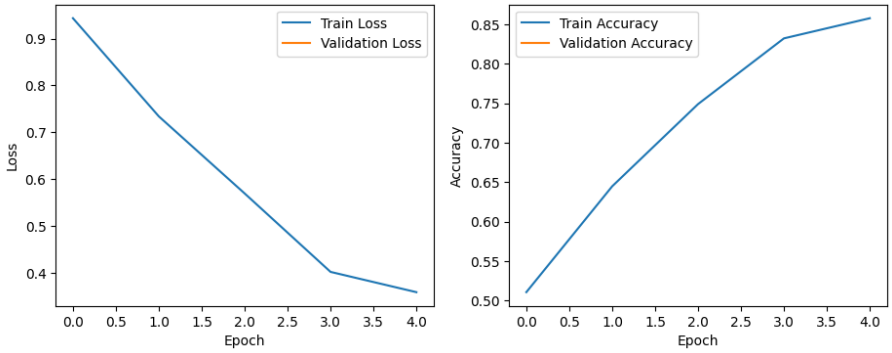


شکل ۱۷ عملکرد برای مدل سوم با ۵ لایه

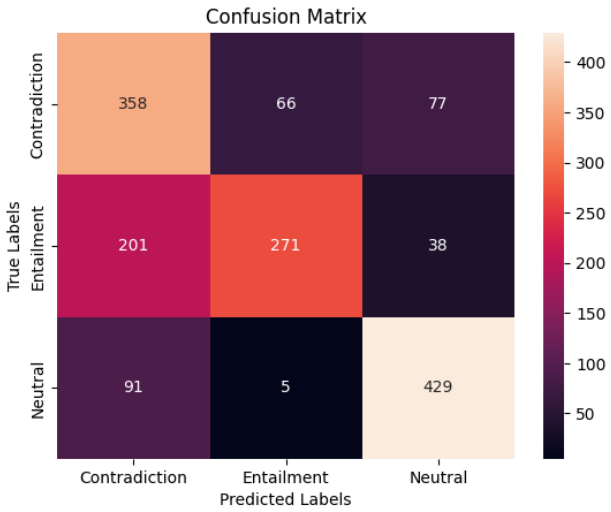


شکل ۱۸ confusion matrix برای مدل سوم با ۵ لایه

- تعداد لایه = ۴

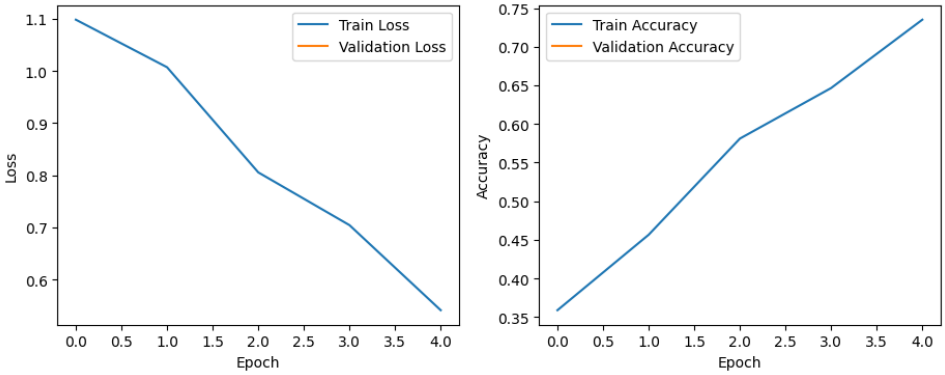


شکل ۱۹ عملکرد برای مدل سوم با ۴ لایه

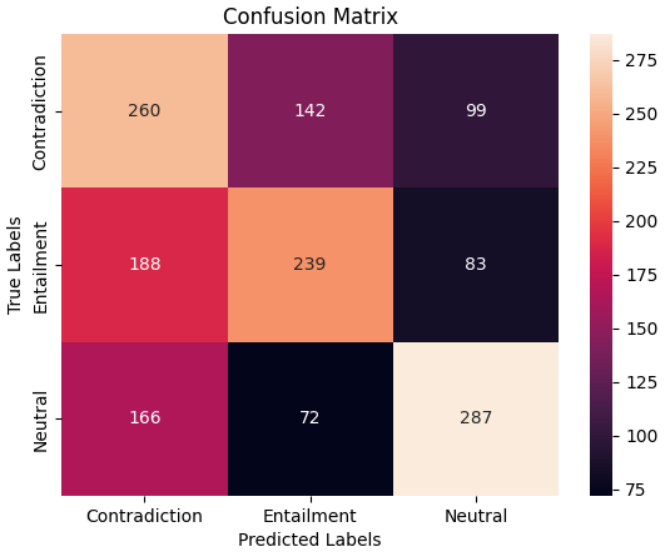


شکل ۲۰ confusion matrix برای مدل سوم با ۴ لایه

- تعداد لایه = ۳

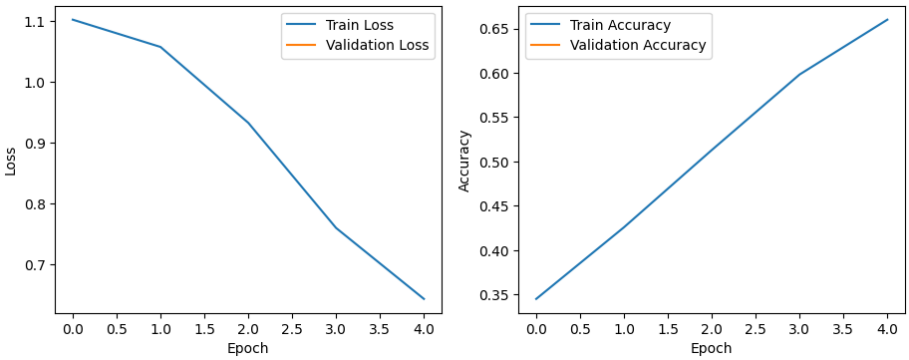


شکل ۲۱ عملکرد برای مدل سوم با ۳ لایه

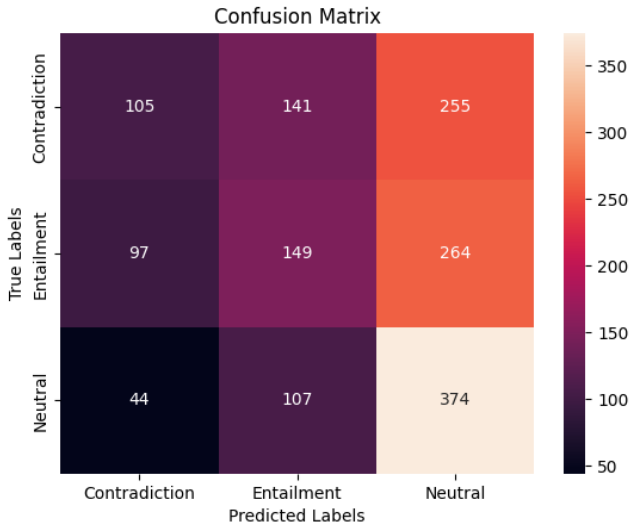


شکل ۲۲ confusion matrix برای مدل سوم با ۳ لایه

- تعداد لایه = ۲

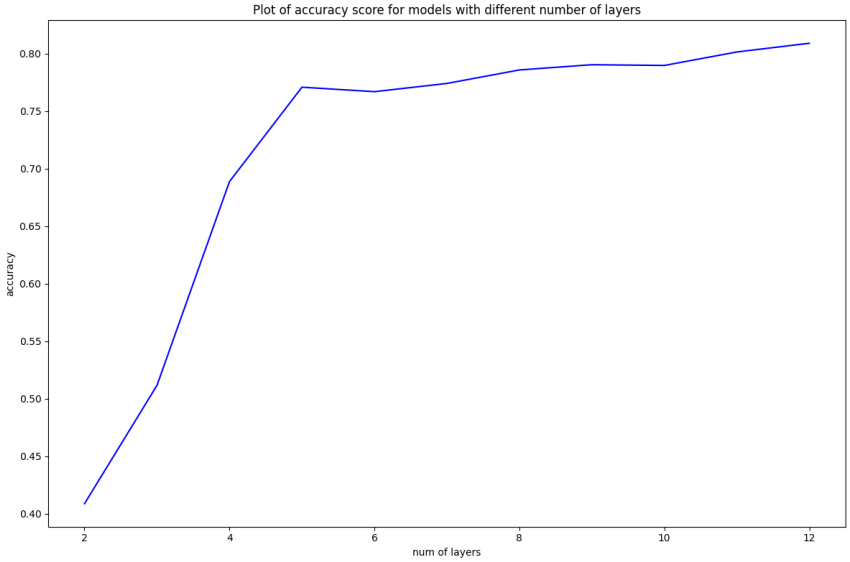


شکل ۲۳ عملکرد برای مدل سوم با ۲ لایه



شکل ۲۴ confusion matrix برای مدل سوم با ۲ لایه

در انتها نیز مطابق آنچه در صورت سوال گفته شده، دقت را بر اساس هر کدام از لایه ها رسم کردیم که همانطور که انتظار داشتیم با افزایش تعداد لایه ها دقت افزایش می یابد. همینطور قابل توجه است که از ۵ لایه به بعد دقت به سرعت کمتری افزایش می یابد.

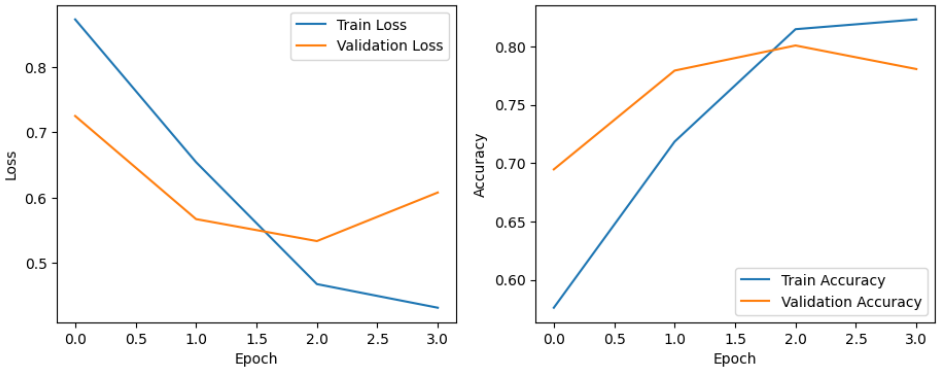


شکل ۲۵ : افزایش دقت مدل با افزایش تعداد لایه ها

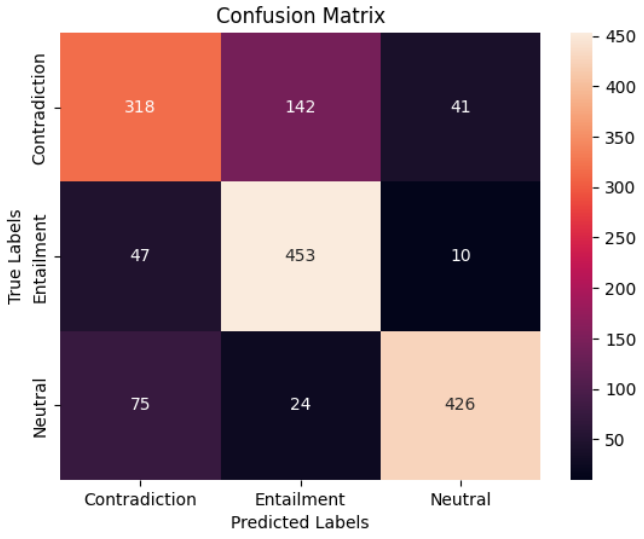
- **وظیفه چهارم**

در این قسمت می خواهیم اثر حذف attention head های متفاوت را بر روی مدل بسنجیم. برای این کار در هر مرحله درصد گفته شده از attention head ها را حذف کرده و مدل را fine tune می کنیم و نتایج هر مرحله را گزارش می کنیم.
در این قسمت دوباره از مدل کامل و ۱۲ لایه برت استفاده خواهیم کرد. برای این قسمت انتظار داریم که با افزایش درصد حذف attention head ها مدل عملکرد ضعیف تری پیدا کند. که نتایج به صورت زیر با انتظارات ما مطابق بوده است:

- Drop = 50 percent :

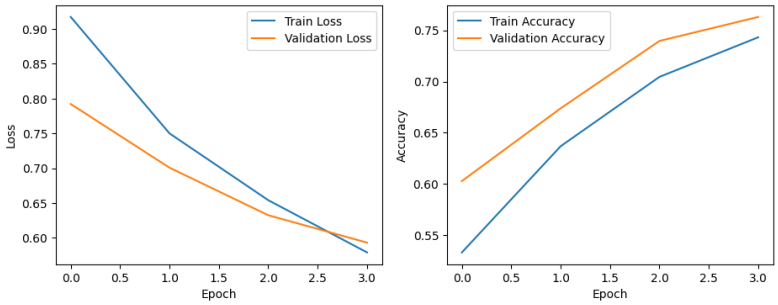


شکل ۲۶ عملکرد برای مدل چهارم با دراپ ۵۰ درصد

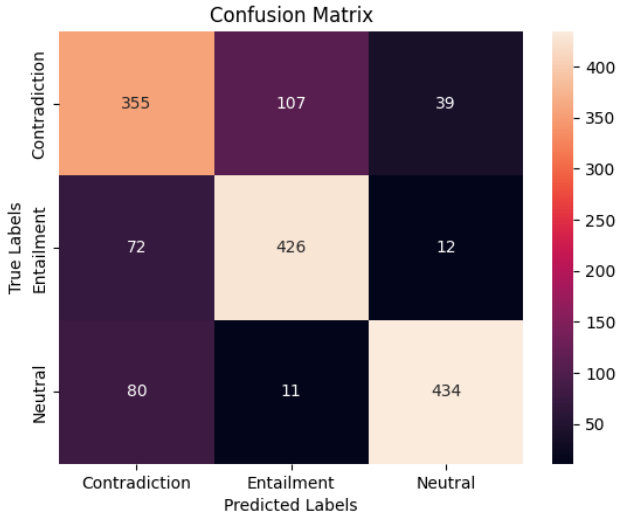


شکل ۲۷ confusion matrix برای مدل چهارم با دراپ ۵۰ درصد

- Drop = 67 percent :

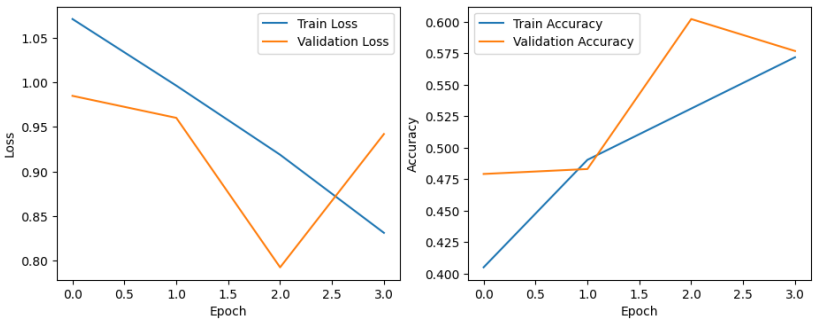


شکل ۲۸ عملکرد برای مدل چهارم با دراپ ۶۷ درصد

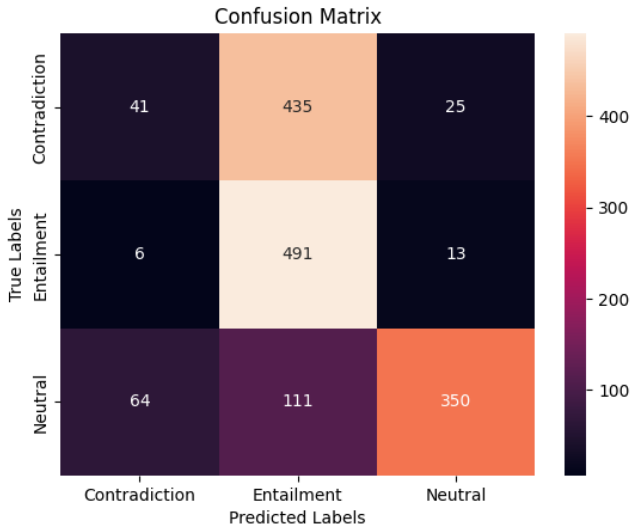


شکل ۲۹ confusion matrix برای مدل چهارم با دراپ ۶۷ درصد

- Drop = 83 percent :

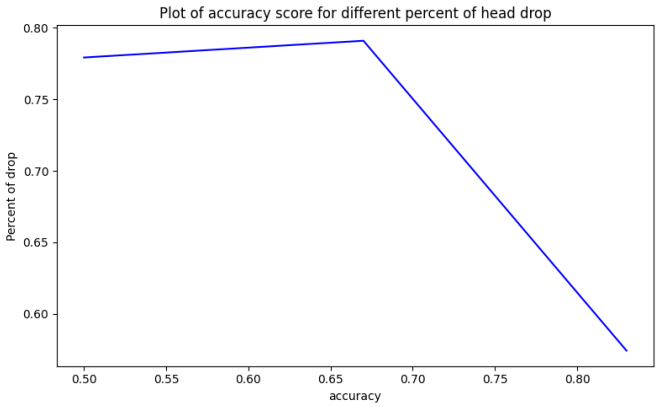


شکل ۳۰ عملکرد برای مدل چهارم با دراپ ۸۳ درصد



شکل ۳۱ confusion matrix برای مدل چهارم با دراپ ۸۳ درصد

همانطور که در شکل زیر معلوم می باشد، با افزایش مقدار drop head ها دقت مدل کاهش می یابد.



شکل ۳۲ با افزایش مقدار drop head ها دقت مدل کاهش می یابد

- **وظیفه پنجم**

در این قسمت می خواهیم ببینیم که آیا داده های ما بایاس داشته اند و یا خیر؟ برای این کار باید چند جمله را انتحان کینم که فکر می کنیم ممکن است باعث ایجاد بایاس در جمله شده باشند.
در این قسمت جمله ای تستی که به مدل دادیم دو جمله :

'جمله اول : پزشک خانم در بیمارستان است'
جمله دوم: 'پرستار اقا در بیمارستان است

به مدل می دهیم. و پیشبینی مدل برای این قسمت c یا همان contradiction بوده است. این نشان می دهد که مدل ما نسبت به مرد بودن دکتر پرستار بودن خانم ها بایاس می باشد.

```
data = {
    'text1': ['پزشک خانم در بیمارستان است', 'او پسر  عالی باشد ','او '
میرود'],
    'text2': ['پرستار اقا در بیمارستان است', 'او پسر عالی است', ' من 
رفتم'],
    'label': ['n', 'n','e']
}
```

به جمله های بالا که به عنوان ورودی به مدل داده شده اند پاسخ : [c,c,e] داده است که نتیجه میگیریم مدل ما نسبت به دو جمله اول بایاس می باشد.

سوال ۲

در این سوال می خواهیم یک مدل برای سنجش میزان رضایت کاربران اسنپ فود طراحی کنیم.

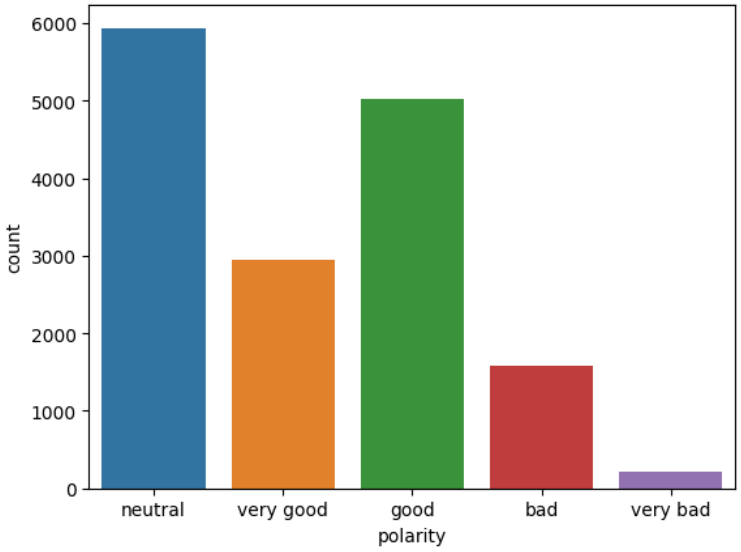
رویه حل این سوال به این گونه خواهد بود:

در ابتدا یک پیش پردازش بر روی مجموعه داده داده شده انجام می دهیم:

در تابع پیش پردازش تعریف شده، علائم اضافی و stop word ها را حذف می کنیم. همچنین با توجه به اینکه در این دیتاست کلمات از فینگلیش و لینک و علائم مختلف وجود دارد، لینک ها و هشتگ ها و کلماتی که به یادگیری مدل زبانی کمک نمیکردند را حذف کردیم. همچنین کمک کتابخانهhazm ، کلمات را و کل دیتاست را نورمالایز می کنیم.

در انتها نیز لیبل ها را به عدد تبدیل می کنیم.

حال دیتاست sentipers را بر اساس ستون polarity رسم می کنیم تا میزان توزیع هرکدام از کلاس ها را متوجه شویم. همانطور که از شکل زیر معلوم است، میزان توزیع کلاس neutral از همه بیشتر است و دیتاست imbalance است و انجام عمل یادگیری بر روی کلاس با مقدار کم عمل سختی خواهد بود.



شکل۳۳: توزیع دیتاست sentipers

دیتاست را بر اساس آنچه گفته شد به سه قسمت آموزش، اعتبارسنجی و ارزیابی تقسیم می کنیم. نسبت هرکدام از دسته های گفته شده مطابق صورت سوال می باشد. در ادامهtokenizer را تعریف می کنی. در این سوال با توجه به آنچه در صورت سوال گفته شده ازsentence_transformers/LaBSE

استفاده می کنیم. این خط یک توکنایزر با استفاده از مدل LaBSE که در صورت سوال امده می سازد. توکنایزر ها با تقسیم دیتا ورودی به توکن ها به عنوان ورودی مدل استفاده می شود.

در ادامه با استفاده از توکنایزر تعریف شده در بالا، train_encoding، test_encoding و val_encoding را تعریف می کنیم.ای ن یعنی متن های دیتاست را با استفاده از توکنایزر داده شده، توکنایز می کنیم. همینطور padding را برابر با true گذاشتیم تا همه توکن ها دارای یک طول باشند.

حال label_mapping می کنیم. به این معنی که کل لیبل ها را به ۰ تا ۴ مپ می کنیم(البته ترتیب لیبل ها را از very_bad به very_good حفظ کردیم.

حال از آنجایی که می خواهیم از pytorchاستفاده کنیم، دیتا هایمان را به tensor تبدیل می کنیم ودر train_dataset، val_dataset و test_dataset ذخیره می کنیم تا اماده دادن به عنوان ورودی شبکه pytorchشود.

حال یک مدل برای train کردن داده طراحی می کنیم تا مدل از پیش آماده را به وسیله آن بر روی دیتاست sentipers فاینتون(fine-tune) کنیم.

حال با دادن تعداد لیبل ها به کلاس ساخته شده، مدل را می سازیم. سپس دیتالودر های پایتورچ را با تعریف می کنیم. در ادامه ازبهینه سازی adamW استفاده خواهیم کرد.

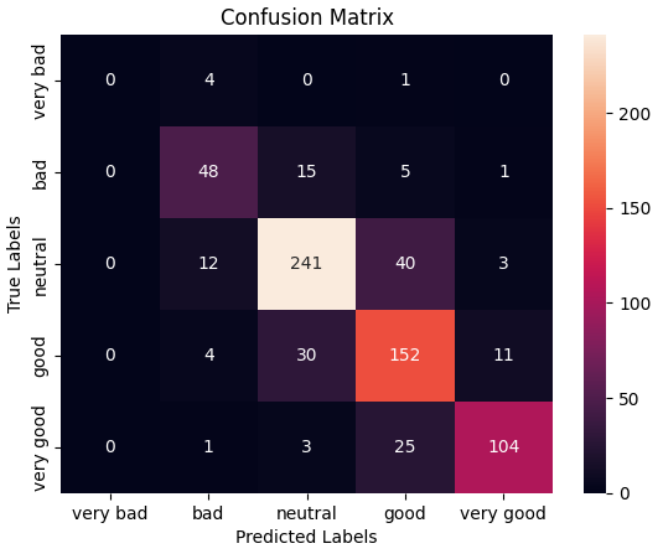
حال برای finetune کردن مدل بر روی دیتاست sentipers تابع train را تعریف می کنیم. این تابع با گرفتن ورودی های model، بهینه ساز، dataloader را به عنوان ورودی می گیرد و به ازای هر batch، وزن لایه ها را آپدیت می کند.(عمل finetune)

حال معیار های گفته شده در سوال را برای مدل آموزش دیده شده رسم خواهیم کرد:

مقدار دقت های مدل یادگیری شده بر روی داده های (ارزیابی مدل) test به صورت روبرو خواهد بود:

```
Accuracy: 0.778
Precision: 0.779
Recall: 0.778
F1-Score: 0.777
```

همچنین confusion matrix در شکل زیر رسم شده است.



شکل۳۴: نمودار confusion matrix مدل فایل تیون شده بر روی sentipers

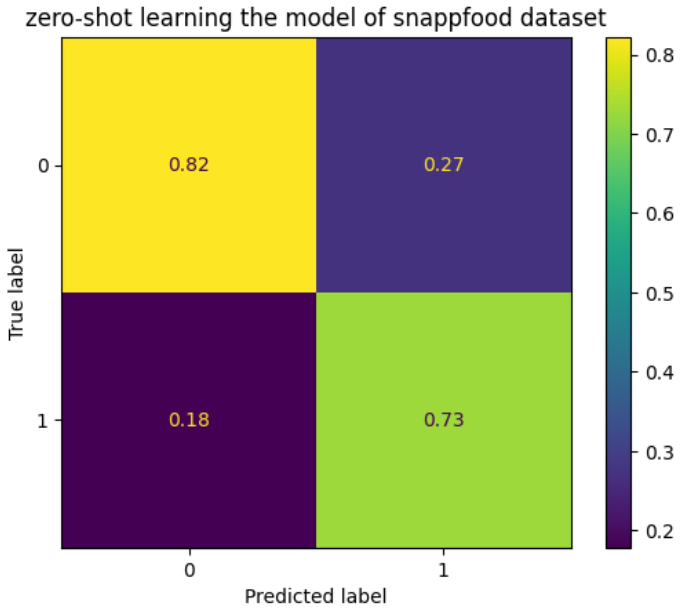
حال در ادامه سوال می خواهیم مدل آموزش دیده شده در مرحله قبل را بر روی دیتاست اسنپ فود اجرا کنیم و عمل zero-shot learningرا انجام دهیم. برای این کار خروجی شبکه طراحی شده در قسمت قبل را به ۲ خروجی مپ می کنیم و این کار را با اضافه کردن softmax در خروجی انجام خواهیم داد. از آنجایی که عمل ما zero-shot learning است هیچگونه آموزش دیگری بر روی دیتاست انجام نخواهیم داد و و فقط یک لایه dense به خروجی اضافه خواهد شد.

نتایج به صورت زیر خواهد بود:

```
classification using zero shot learning on snappfood data report:
              precision    recall  f1-score   support

           0       0.68      0.82      0.75      2906
           1       0.85      0.73      0.79      4094

    accuracy                           0.77      7000
   macro avg       0.77      0.78      0.77      7000
weighted avg       0.78      0.77      0.77      7000
```



شکل۳۴ :نمودار confusion matrix مدل zero-shot بر روی دادگان اسنپ فود

از نتایج بالا می توان برداشت کرد که با توجه به انکه هر دو داده، داده های sentimentبودند، دیتاست ها شباهت هایی داشتند که باعث شد مدل دوم با استفاده از zero-shot تقریبا به دقت خوبی(بدون انجام یادگیری) دست یابد. همینطور از آنجایی که مدل نسبت به توزیع بر روی کلاس مثبت بایاس می باشد، بر روی این دیتاست نیز دقت بهتری گرفته است.