بسم الله الرحمن الرحیم

دانشکده مهندسی برق و کامپیوتر

بازیابی هوشمند اطلاعات - تمرین چهارم

سید مهدی رضوی

استاد : خانم دکتر شاکری

آذر ماه ۱۴۰۲

فهرست مطالب

فهرست تصاویر

۱ تمرین اول

در تصاویر زیر می‌توانید دقت این مدل به ازای label های مختلف را مشاهده‌بفرمایید.

- خیر. تعداد تگ‌های هر دسته متعادل نمی‌باشد. تعداد و میزان هر تگ را می‌توان از نمودار پایین مشاهده نمود. بیشترین تگ مربوط به تگ مثبت و کمترین تگ مربوط به تگ خنثی می‌باشد.
- معماری : در واقع مدل ما در ۳ اپیاک آموزش دیده است. بر اساس دو معیار زیر که اولی به میزان جریمه ، و دومی به میزان دقت اشاره می‌کند ، به آموزش مدل خود بر حسب داده آموزشی می‌پردازیم.

$$\text{loss} = \text{tf.keras.losses.SparseCategoricalCrossentropy(fromlogits=True)}$$
$$\text{metric} = \text{tf.keras.metrics.SparseCategoricalAccuracy('accuracy')}$$

سپس با استفاده از کد زیر بهترین وزن‌دهی‌ها را ذخیره خواهیم کرد.

$$\text{bertmodel.saveweights(modelsavepath)}$$

- با توجه به نمودارهای زیر مشخص است که مدل آموزش دیده دچار بیش برازش نشده است ، چرا که هیچ گونه اثری از انطباق بر روی نمودار آموزش و اعتبارسنجی مشاهده نمی‌شود.
- مراحل ذکر شده در داکیومنت تمرین چهارم به صورت دقیق در کد ، پیاده‌سازی گردیده‌است. همچنین توضیحات مختصری در قالب کامنت در سلول‌های Text قرار داده‌شده‌است.

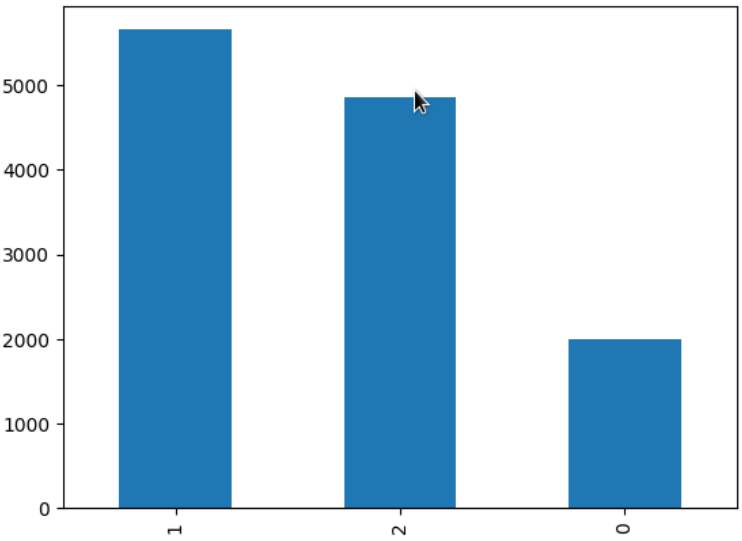
نکته جالبی که می‌توان به آن اشاره کرد این است که در قسمت تابع ضرر (Loss Function) در صورتی که یکی از تگ های داده را به یک عدد منفی مثل -۱ نظیر کنیم ، به شدت میزان دقت مدل کاهش می‌یابد. (نتیجه‌ای که شخصا به آن رسیدم این بود که این تابع محاسبه ضرر به ازای منفی بودن تگ‌ها به شدت دقت پایینی را محاسبه خواهد کرد ، به طوری که دقت مدل من در حدود ۱۵ درصد بوده است.)

```
# data['text']=data['text'].map(preprocess_sentence)                    # Clean the tex
num_classes=len(data.label.unique())
data['label'] = data['label'].map({'neutral' : 0 , 'positive' : 1 , 'negative' : 2})

print(f'num_classes : {num_classes}')
```

```
Available labels:  ['neutral' 'negative' 'positive']
num_classes : 3
```

```
data['label'].value_counts().plot.bar()
```

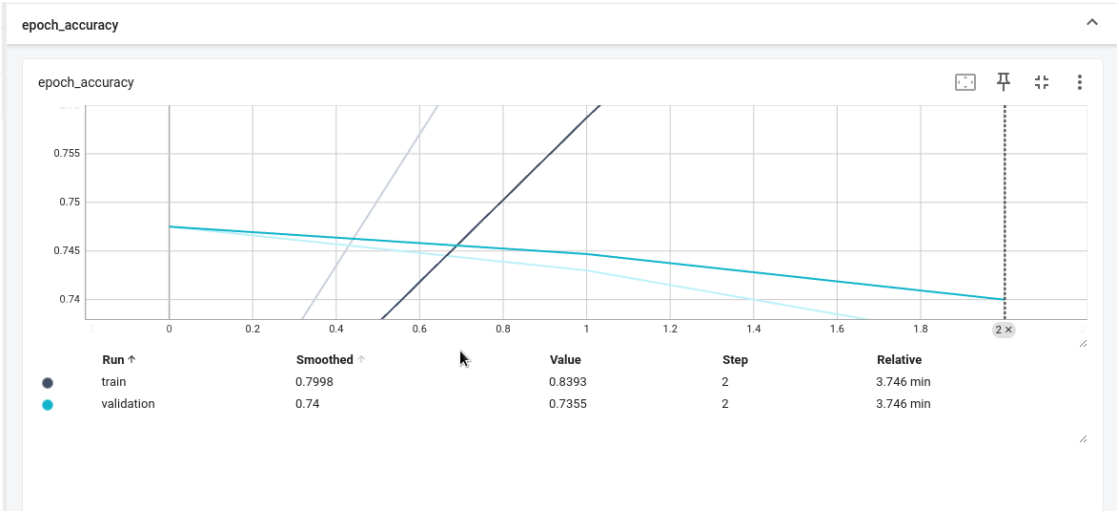
```
<Axes: >
```



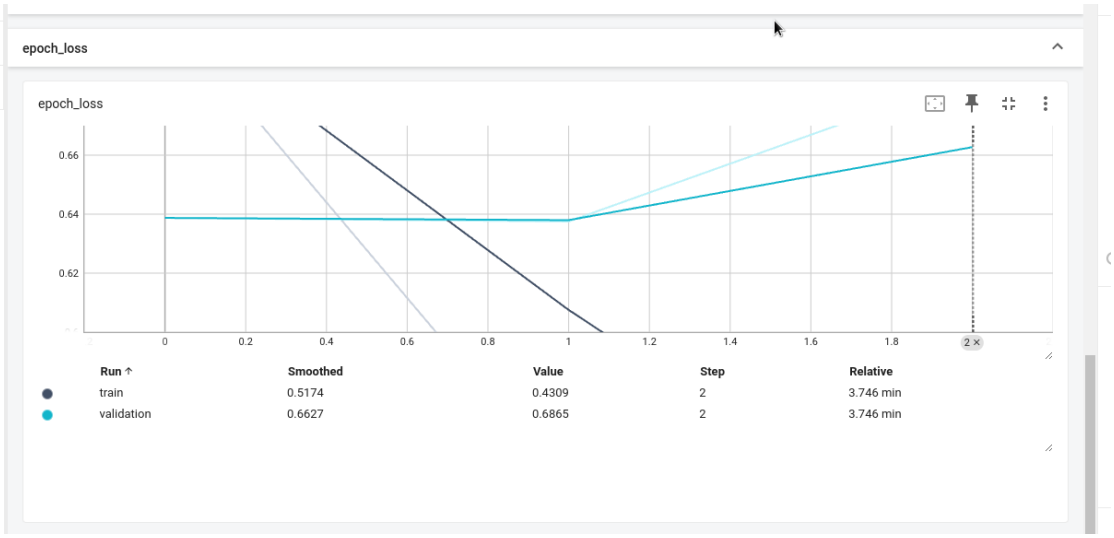
شکل ۱: All labels in dataset

```
  "vocab_size": 30522
}

loading weights file model.safetensors from cache at /root/.cache/huggingface/hub/models--bert-base-uncased/snapshots/1dbc166cf8765166998eff31ade2eb64c8a40076/model.safetensors
Loaded 109,482,240 parameters in the TF 2.0 model.
All PyTorch model weights were used when initializing TFBertForSequenceClassification.

Some weights or buffers of the TF 2.0 model TFBertForSequenceClassification were not initialized from the PyTorch model and are newly initialized: ['classifier.weight', 'classifier.bias']
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
79/79 [==============================] - 14s 139ms/step
F1 score 0.6344937716771927
Classification Report
              precision    recall  f1-score   support

           0       0.42      0.18      0.25       401
           1       0.80      0.88      0.84      1123
           2       0.76      0.86      0.81       975

    accuracy                           0.76      2499
   macro avg       0.66      0.64      0.63      2499
weighted avg       0.73      0.76      0.73      2499
```

شکل ۲: همه مقیاس‌های مدنظر



شکل ۳: tensorflow accuracy



شکل ۴: tensorflow loss

۲ تمرین دوم

- به این دلیل که داده‌های ما باید حداقل دارای یک Relevant Query باشند. و یا حداقل باید از سایر تکنیک‌ها برای پیش‌پردازش متن استفاده شود.

  مانند حذف کلمات بسیار پرتکرار و یا همان به اصطلاح مرسوم Stop Words.

  و همچنین یکسان کردن کلمات هم‌خانواده Stemming

  اما همانطور که قبل‌تر ذکرشد ، مساله اصلی ما ایجاد و به عبارت بهتر پیش‌بینی Relevant Query می‌باشد.

  به ازای هر داکیومنت ، یا به عبارتی شفاف‌تر در حوزه کتاب باید به ازای هر پاراگراف ، فصل ، نیم بند ، بخش و یا هر دسته‌بندی دیگری باید یک کوئری مرتبط داشته‌باشیم.

  مهمترین چالش‌پیش رو اولیه این می‌تواند باشد.

  جلوتر به چگونگی ایجاد و یا پیش‌بینی کوئری مرتبط با سند می‌پردازیم.

- برای آموزش دادن به مدل برای اسناد با حجم داده زیاد ابتدا بایستی از تکنیک‌های پیش پردازش متن استفاده کنیم.

  (Preprocessing Techniques)

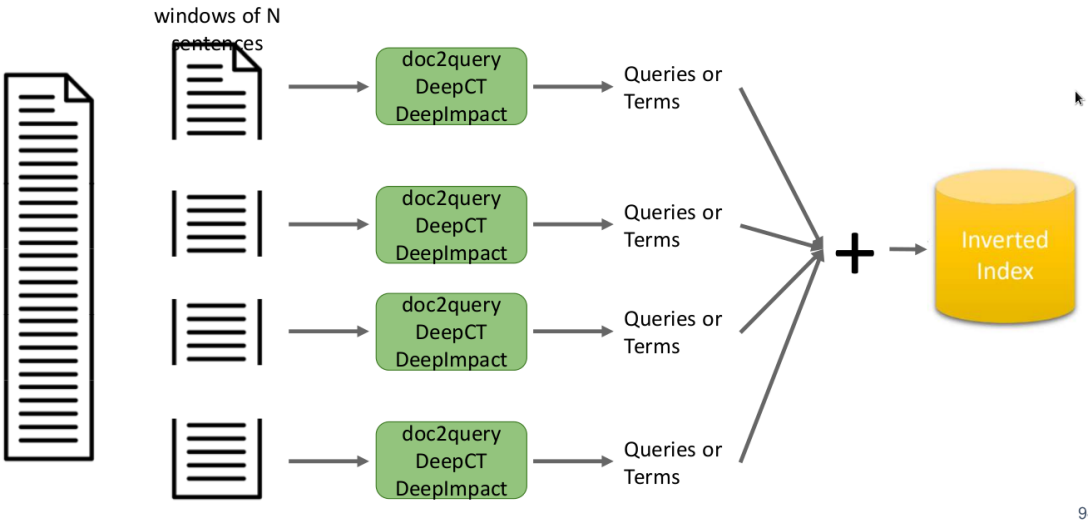
  در زیر به چند تکنیک از این روش خواهیم پرداخت :

  ۱. Query Reformulation
  ۲. doc2query
  ۳. DeepCT
  ۴. DeepImpact(Combine doc2query with DeepCT)

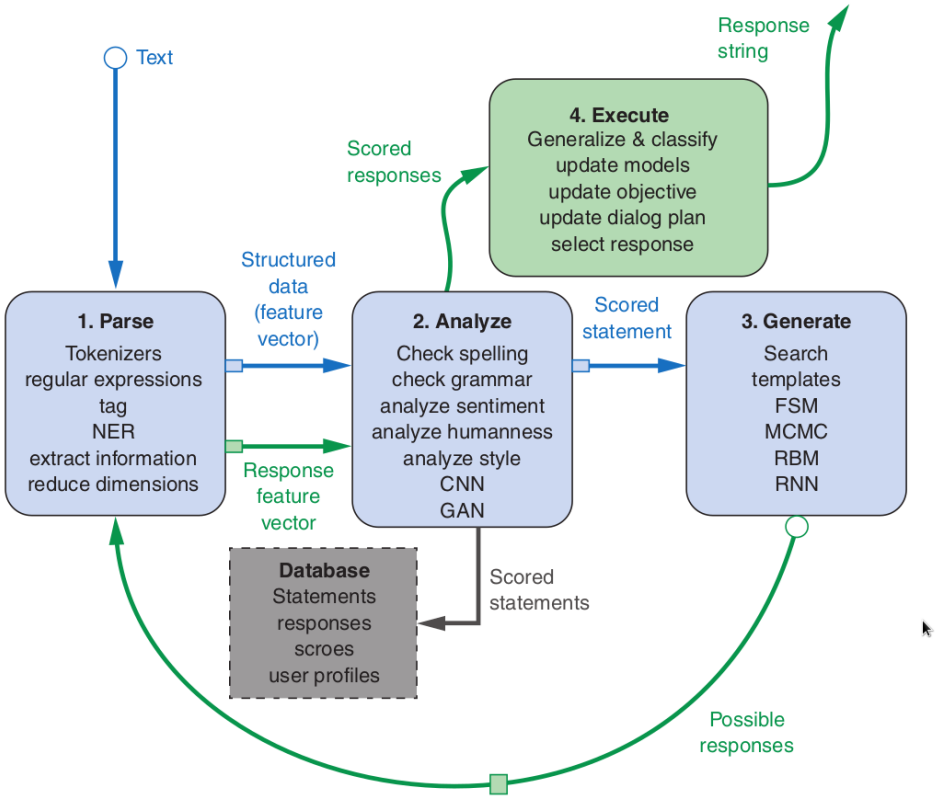
- قبل از هرکاری می‌بایستی که عملیات پیش‌پردازش بر روی اسناد انجام شود. گسترش هر داکیومنت با پیش‌بینی کوئری مرتبط آن با مدل‌های Transformer صورت خواهدگرفت. (مانند Seq2Seq)

  ساختن یک مدل زبانی بزرگ شامل موارد زیر است :

  ۱. مجموعه داده (جمع‌آوری و تهیه یک مجموعه داده وسیع از کتب و منابع مرتبط)
  ۲. مدل زبانی (استفاده از مدل زبانی بزرگ بر اساس شبکه‌های عصبی)
  ۳. ماژول پرسش و پاسخ (ایجاد یک ماژول پرسش و پاسخ که وظیفه پردازش سوالات کاربر را برعهده دارد.)
  ۴. مدیریت دانش (در واقع ایجاد یک نوع ایندکس بر روی متون کتاب‌ها بر اساس موضوعات و مفاهیم مختلف)
  ۵. بهبود پرسش و پاسخ (ارزیابی پاسخ‌های تولیدشده بر اساس برخی معیارهای ارزیابی)



شکل ۵: نمایی از عملیات پیش‌پردازش بر روی متون بزرگ



شکل ۶: معماری یک سیستم مدل زبانی بزرگ ساده

۳ تمرین سوم

برای منظور این سوال که کشف تقلب است ، ما یکی از بهترین کارهایی که میتوانیم انجام دهیم در وهله نخست : یافتن اسناد مشابه با استفاده از معیار صحیح شناسایی تشابه بین اسناد است.
در نتیجه باید یک حدی را تعیین کنیم که از این حد به بعد ، میزان تشابه برای ما **غیر معمول** به نظر برسد. (یافتن یک آستانه)
در واقع مساله ما به این مساله تغییر می‌کند : آیا می‌توان سند یا اسنادی را یافت که بیش از آستانه به یک سند و یا چندین سند مشابه باشند ؟
همانند سوال‌های اول و دوم عملیات tokenization برای تبدیل Sequence به بردار Vector باید صورت پذیرد.
در صورتی دو بردار شبیه هم خواهند بود که میزان تشابه کسینوسی دو بردار هم‌جهت بودن دو بردار و نزدیک‌تر بودن زاویه بین دو بردار را تایید کند.
برای این منظور ابتدا باید به پیش‌پردازش داده بپردازیم . سپس با استفاده از مدل Bert به ساختن بردار ایندکس بپردازیم.
سپس پس از دریافت مشابه‌ترین اسناد به سند اضافه شده ، باید به گام دوم سوال که یافتن بخش‌هایی از سند است که مورد تقلب قرارگرفته‌است بپردازیم.
برای این قسمت نیز می‌توانیم همانند قسمت قبل عمل‌کنیم. با این تفاوت که هر پاراگراف را یک سند در نظربگیریم و به محاسبه شباهت کسینوسی بپردازیم. البته در این مرحله بهتر است از برخی روابط بین کلمات نیز استفاده کنیم تا الگوریتم ما بتواند بهتر کلمات مترادف و یا کلماتی که بهترین روابط **جانشینی** را دارند ، استفاده کنیم.