به نام خدا

دانشگاه تهران

پردیس دانشکده‌های فنی

دانشکده برق وکامپیوتر

درس پردازش زبان های طبیعی

پروژه شماره : 4

نام و نام خانوادگی : محمدرضا بختیاری

شماره دانشجویی : 810197468

خرداد 1401

فهرست سوالات

3 ..... Part 1- ParsiNLU dataset classification
3 ..... 1- Train data analysis
5 ..... 2- XLM-RoBERTa
6 ..... 3- ParsBERT
8 ..... Part 2- Multilingual classification
9 ..... 1- English corpus
12 ..... 2- Persian corpus
13 ..... 3- Both English and Persian corpus
15 ..... Part 3- Cross-lingual zero-shot transfer learning (Optional)
15 ..... Q1
15 ..... Q2
16 ..... Q3
17 ..... Attachment

Part 1- ParsiNLU dataset classification

1- Train data analysis

برای شروع ابتدا نیاز است تنظیمات لازم را انجام دهیم و پکیج های مورد نیاز در طول سوال را نصب کنیم :

Setup

```
!pip install transformers
!pip install sentencepiece
!pip install datasets
```

شکل 1-1 : نصب پکیج های مورد نیاز

در ادامه پکیج های مورد نیاز را وارد[1] می کنیم :

Import packages

```
[ ] import random
    import pandas as pd
    from transformers import MT5ForConditionalGeneration, MT5Tokenizer
    from datasets import load_dataset
    from IPython.display import display, HTML
```

شکل 1-2 : ایمپورت کردن پکیج های مورد نیاز

پس از آن نیاز است داده های آموزش[2] , آزمون[3] و اعتبارسنجی[4] را بارگیری کنیم :

Load Dataset and split into {train,test,validation}

```
[ ] model_size="base"
    model_name = f"persiannlp/mt5-{model_size}-parsinlu-snli-entailment"
    dataset = load_dataset("persiannlp/parsinlu_entailment")
    parsinlu_train = load_dataset('persiannlp/parsinlu_entailment', split='train')
    parsinlu_test = load_dataset('persiannlp/parsinlu_entailment', split='test')
    parsinlu_valid = load_dataset('persiannlp/parsinlu_entailment', split='validation')
    tokenizer = MT5Tokenizer.from_pretrained(model_name)
    model = MT5ForConditionalGeneration.from_pretrained(model_name)
```

شکل 1-3 : بارگیری دادگان آموزش , آزمون و اعتبارسنجی

حال می توانیم نگاهی کلی به دادگان بیاندازیم تا ببینیم نیاز به پیش پردازش خاصی داریم یا خیر.

با استفاده از تابع زیر , 10 نمونه از دادگان آموزش را به صورت تصادفی مشاهده می کنیم :

---

[1] Import
[2] Train
[3] Test
[4] Validation

Display random examples (train data)

```
[ ] def display_random_examples(dataset=parsinlu_train, num_examples=4):
        assert num_examples < len(dataset)

        random_picks = []
        for i in range(num_examples):
            random_pick = random.randint(0,len(dataset)-1)
            random_picks.append(random_pick)

        df = pd.DataFrame(dataset[random_picks])
        display(HTML(df.to_html()))

    display_random_examples(parsinlu_train, 10)
```

شکل 4-1 : نمایش 10 نمونه از دادگان آموزش به صورت تصادفی

که خروجی آن به شکل زیر خواهد بود :

	sent1	sent2	category	label
0	فرانسیسکو او به زبان اسپانیایی صدا زد ، Unbar.	او فرانسیسکو را به زبان اسپانیایی صدا می کند.	translation-train	e
1	و همانطور که احتمالاً می دانید ، در ارائه تضمین های دیناری آینده خود ، سرویس پستی از داده های مالی و عملیاتی سال مالی ۱۹۹۸ به عنوان معیار استفاده کرد.	سرویس پستی قادر به پاقتن داده های مالی و عملیاتی نبود.	translation-train	c
2	من فکر نمی کنم آنها حس مسئولیت داشته باشند.	من فکر نمی کنم آنها احساس مسئولیت کنند که دیگران را ایمن نگه دارند.	translation-train	n
3	همچنین ، اگر مالیاتهای غیر مستقیم ایالتی و فدرال در سطح فعلی باقی بمانند ، به دلیل کاهش فروش ، درآمد مالیاتی با گذشت زمان کاهش می یابد.	مالیاتهای غیر مستقیم ایالتی و فدرال کمتر از یک سال در این سطح بوده اند.	translation-train	n
4	مرگ او به علت سرطان ریه بود. اگر چه در تاریخ اجراهای تلویزیونی به نام کافق به عنوان یک کمدین دگربسته می‌شود، اما او خود را یک کمدین نمی‌دانست و خود را با نام «مرد رقص و آواز» لقب داده بود.	او همواره خود را از برترین کمدین‌های تاریخ می‌دانست ولی در جوانی در اثر سرطان در گذشت.	natural-wiki	c
5	حدود سال‌های ۱۳۵۰ سید محمود طالقانی اجازهٔ دریافت وجوهات برای کمک به مردم نیازمند و انجام فعالیت‌های مددکاری را به منصوریان داد.	طالقانی در دههٔ پنجاه، یکی از روحانیون محبوب آن زمان بود.	natural-wiki	n
6	واتیکان دوم باعث ایجاد یک مذهب کاتولیک کمتر سلسله مراتبی و بیرونی تر شد و زمینه ای برای نوآوری های غیرقابل تصور مانند اجازه پوشیدن لباس شخصی به راهبه ها و برگزاری مراسم به زبان انگلیسی و سایر زبان های مدرن فراهم آورد.	واتیکان دوم باعث تمرکز بیشتر قدرت و ساختار سلسله مراتبی شد.	translation-dev	c
7	درحمله ای جداگانه در بزرگراه پایتخت به فرودگاه مردان مسلح به دو اتومبیل که احتمالا حامل پیمانکاران غیر نظامی بود، حمله کردند.	صبح امروز در فرودگاه پایتخت پرواز های زیادی انجام شد.	natural-voa	n
8	این خنده دار است	این یک داستان مضحک است.	translation-dev	e
9	نقاشانی که به تعامل نور با مدل‌ها و محیط اطرافشان اهمیت بسزایی می‌دادند.	نقش نور در نقاشی شان بسیار مهم بود.	natural-wiki	e

شکل 5-1 : تعدادی از دادگان آموزش که به صورت تصادفی انتخاب شده اند

با مشاهده و ارزیابی کل دادگان آموزش (در اینجا به علت محدودیت و طولانی نشدن گزارش فقط 10 نمونه تصادفی از دادگان را مشاهده کردیم) مشاهده می کنیم برچسب[1] تعدادی از دادگان خارج از برچسب تعریف شده اولیه (e,c,n) بوده که نیاز داریم این مشکل را به گونه ای رفع کنیم , زیرا وجود چنین دادگانی با برچسبی خارج از برچسب کلاس های ما ممکن است به عملکرد مدل آسیب برساند.

برای رفع این مشکل دو راهکار داریم :

1- حذف کامل این دادگان از دادگان آموزش.

2- انتخاب یک برچسب مناسب برای این دادگان (یا به صورت تصادفی یا به صورت دستی برای هر یک از این دادگان با توجه به sent1 , sent2 خودمان برچسب گذاری مناسب را انجام دهیم).

که در اینجا برای صرف جویی در زمان و دخل و تصرف نکردن در دادگان آموزش , راهکار اول را انتخاب می کنیم.

[1] Label

2- XLM-RoBERTa

در ادامه نیاز داریم یک شبکه عصبی عمیق طراحی کنیم و در نهایت دقت مدل را بر روی دادگان آزمون بررسی کنیم. جهت اعمال جاسازی کلمات[1] از مدل های چند زبانه بِرت[2] استفاده می کنیم.

از مدل XLM-RoBERTa به منظور پیاده سازی این بخش استفاده می کنیم :

XLM-RoBERTa

```
[ ] from transformers import pipeline
    unmasker = pipeline('fill-mask', model='xlm-roberta-base')
    from transformers import AutoTokenizer, AutoModelForMaskedLM
    tokenizer = AutoTokenizer.from_pretrained('xlm-roberta-base')
    model = AutoModelForMaskedLM.from_pretrained("xlm-roberta-base")
    # prepare input
    text = "Replace me by any text you'd like."
    encoded_input = tokenizer(text, return_tensors='pt')
    # forward pass
    output = model(**encoded_input)
```

شکل 6-1 : مدل XLM-RoBERTa به همراه یک نمونه ورودی

در ادامه مدل خود را ساخته و پس از آن نوبت به ارزیابی عملکرد مدل بر روی دادگان آزمون می رسد:

Building a model

```
[ ] def run_model(premise, hypothesis, **generator_args):
        input_ids = tokenizer.encode(f"{premise}<sep>{hypothesis}", return_tensors="pt")
        res = model.generate(input_ids, **generator_args)
        output = tokenizer.batch_decode(res, skip_special_tokens=True)
        #print(output)
        return output
```

شکل 7-1 : ساخت مدل نهایی

همان طور که می دانیم XLM-RoBERTa یک مدل چند زبانه است که بر روی 100 زبان مختلف آموزش دیده است. برخلاف برخی از مدل‌های چند زبانه XLM ، برای فهمیدن اینکه کدام زبان استفاده می‌شود ، نیازی به تنسورهای زبانی ندارد و باید بتواند زبان صحیح را از شناسه های ورودی تعیین کند.

توجه : زمان اجرای کد برای بررسی عملکرد مدل بر روی دادگان آزمون که در ادامه آمده است به علت حجم دادگان و پیچیدگی مدل و تسک مربوطه (Textual entailment) در حالت عادی بیش از یک ساعت و با استفاده از GPU حدود 20-30 دقیقه به طول می انجامد , لذا پیشاپیش از صبر و شکیبایی شما کمال تشکر را دارم.

[1] Word embedding
[2] Multilingual

در ادامه نیز دقت مدل را روی دادگان آزمون بررسی می کنیم :

Evaluating the model

```
[ ] Predicted_Class = ''
    Real_Class = ''
    sent1 = ''
    sent2 = ''
    TP = 0
    FN = 0
    for Test in range(0,parsinlu_test.num_rows):
      sent1 = parsinlu_test['sent1'][Test]
      sent2 = parsinlu_test['sent2'][Test]
      Real_Class = parsinlu_test['label'][Test]
      Predicted_Class = run_model(sent1,sent2,)
      if (Predicted_Class[0] == Real_Class):
        TP = TP + 1
      else:
        FN = FN + 1
    print('Accuracy using XLM-RoBERTa is : ',((TP) / (TP+FN))*100,' %')
```

شکل 8-1 :

که در نهایت دقتی نزدیک به 60 درصد بر روی دادگان آزمون به دست خواهد آمد :

```
Accuracy using XLM-RoBERTa is :  60.17910447761194  %
```

شکل 9-1 : دقت به دست آمده بر روی دادگان آزمون با مدل XLM-RoBERTa

3- ParsBERT

در این قسمت نیز روند کار و پیاده سازی ها مشابه قسمت قبل بوده با این تفاوت که در اینجا از مدل ParsBERT به منظور پیاده سازی این بخش استفاده می کنیم :

ParsBERT

```
[ ] # Import generic wrappers
    from transformers import AutoModel, AutoTokenizer
    # Define the model repo
    model_name = "persiannlp/parsbert-base-parsinlu-entailment"
    # Download pytorch model
    model = AutoModel.from_pretrained(model_name)
    tokenizer = AutoTokenizer.from_pretrained(model_name)
    # Transform input tokens
    inputs = tokenizer("Hello world!", return_tensors="pt")
    # Model apply
    outputs = model(**inputs)
```

شکل 10-1 : مدل ParsBERT به همراه یک نمونه ورودی

در ادامه نیز مشابه با آنچه در قسمت قبل انجام دادیم , عملکرد مدل را برای دادگان آزمون بررسی می کنیم و خواهیم داشت :

```
Accuracy using ParsBERT is :  66.09413478264264  %
```

شکل 11-1 : دقت به دست آمده بر روی دادگان آزمون با مدل ParsBERT

مشاهده می کنیم دقت به دست آمده بر روی مدل PasrBERT از مدل XLM-RoBERTa اندکی بیشتر بوده , اگر چه مدل XLM-RoBERTa یک مدل بسیار قوی بوده و بر روی 100 زبان مختلف و حدود 2.5 ترابایت داده آموزش داده شده است اما به علت این که در این جا دادگان ما فارسی هستند و مدل ParsBERT نیز بر روی دادگان فارسی آموزش داده شده است , از قبل هم انتظار می رفت که در اینجا این مدل دقت بیشتری را برای ما به همراه داشته باشد.

توجه : در هنگام اجرای کدهای این بخش , سل های مربوط به XLM-RoBERTa و ParsBERT را با هم اجرا نکنید (به ارور برخورد خواهید کرد) , و نیاز است هر کدام را به طور جداگانه اجرا کنید. (یک بار مدل اول اجرا شود و پس از دیدن نتایج , مدل دوم اجرا شود و نتایج حاصل شود)

Part 2- Multilingual classification

برای شروع ابتدا نیاز است تنظیمات لازم را انجام دهیم و پکیج های مورد نیاز در طول سوال را نصب کنیم :

Setup

```
[ ]  !pip install tensorflow
     !pip install tensorflow_hub
     !pip install tensorflow_text
     !pip install tensorflow-text==2.4.1
     !pip install tf-models-official==2.4
     !pip install pandas
```

شکل 2-1 : نصب پکیج های مورد نیاز

در ادامه پکیج های مورد نیاز را وارد[1] می کنیم :

Importing important packages

```
[ ]  import tensorflow as tf
     import tensorflow_hub as hub
     import tensorflow_text as text
     import pandas as pd
     import numpy as np
     from sklearn.model_selection import train_test_split
```

شکل 2-2 : ایمپورت کردن پکیج های مورد نیاز

پس از آن نیاز است داده های آموزش[2] , آزمون[3] و اعتبارسنجی[4] را بارگیری کنیم :

Loading dataset

```
[ ]  # Load the xlsx file
     df = pd.read_excel('train.xlsx')
     df_test = pd.read_excel('test.xlsx')
     df_val = pd.read_excel('valid.xlsx')
     # Read the values of the file in the dataframe
     data = pd.DataFrame(df, columns=['source', 'targets', 'category'])
     data_test = pd.DataFrame(df_test, columns=['source', 'targets', 'category'])
     data_val = pd.DataFrame(df_val, columns=['source', 'targets', 'category'])
```

شکل 2-3 : بارگیری دادگان آموزش , آزمون و اعتبارسنجی

[1] Import
[2] Train
[3] Test
[4] Validation

پس از آن نیاز داریم تا از تعادل[1] دادگان خود اطمینان حاصل کنیم :

Balancing dataset

```
[ ] df_quran = df[df['category']=='quran']
    df_bible = df[df['category']=='bible']
    df_mizan = df[df['category']=='mizan']
    df_test_quran = df_test[df_test['category']=='quran']
    df_test_bible = df_test[df_test['category']=='bible']
    df_test_mizan = df_test[df_test['category']=='mizan']
    df_balanced = pd.concat([df_mizan, df_bible, df_quran])
    df_test_balanced = pd.concat([df_test_mizan, df_test_bible, df_test_quran])
```

شکل 2-4 : متعادل کردن دادگان

در ادامه برای هر یک از برچسب[2] های خروجی یک عدد متناظر با آن در نظر می گیریم : (quran معادل با 1 , bible معادل با 0.5 و mizan معادل با 0)

Adding labels

```
[ ] # {quran = 1 , bible = 0.5 , mizan = 0}
    df_balanced['quran']=df_balanced['category'].apply(lambda x: 1 if x=='quran' else 0.5 if x=='bible' else 0)
    df_test_balanced['quran']=df_test_balanced['category'].apply(lambda x: 1 if x=='quran' else 0.5 if x=='bible' else 0)
```

شکل 2-5 : گسسته سازی برچسب های خروجی

حال دادگان آموزشی آماده می باشند و در ادامه به طراحی شبکه های عصبی عمیق و بررسی عملکرد مدل در هر بخش می پردازیم.

1- English corpus

برای پیاده سازی این قسمت از مدل BERT استفاده خواهیم کرد. در ابتدا لازم است مدل مورد نظر را دریافت کرده :

Downloading the BERT model

```
[ ] bert_preprocess = hub.KerasLayer("https://tfhub.dev/tensorflow/bert_en_uncased_preprocess/3")
    bert_encoder = hub.KerasLayer("https://tfhub.dev/tensorflow/bert_en_uncased_L-12_H-768_A-12/4")
```

شکل 2-6 : دریافت مدل مورد نظر (در اینجا BERT)

سپس با استفاده از TensorFlow شروع به ساخت مدل می کنیم.

[1] Balance
[2] Label

اولین گام مقداری دهی اولیه به لایه های شبکه عصبی مورد نظر است :

Building model using TensorFlow

Initializing the BERT layers

```
[ ]  text_input = tf.keras.layers.Input(shape=(), dtype=tf.string, name='text')
     preprocessed_text = bert_preprocess(text_input)
     outputs = bert_encoder(preprocessed_text)
```

Initializing the neural network layers

```
[ ]  l = tf.keras.layers.Dropout(0.1, name="dropout")(outputs['pooled_output'])
     l = tf.keras.layers.Dense(1, activation='sigmoid', name="output")(l)
     model = tf.keras.Model(inputs=[text_input], outputs = [l])
     model.summary()
```

شکل 7-2 : مقدار دهی اولیه شبکه عصبی

پس از آن خلاصه ای از مدل به دست آمده را مشاهده می کنیم :

```
Model: "model_2"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
text (InputLayer)               [(None,)]            0
__________________________________________________________________________________________________
keras_layer_4 (KerasLayer)      {'input_word_ids': ( 0           text[0][0]
__________________________________________________________________________________________________
keras_layer_5 (KerasLayer)      {'default': (None, 7 109482241   keras_layer_4[0][0]
                                                                 keras_layer_4[0][1]
                                                                 keras_layer_4[0][2]
__________________________________________________________________________________________________
dropout (Dropout)               (None, 768)          0           keras_layer_5[0][13]
__________________________________________________________________________________________________
output (Dense)                  (None, 1)            769         dropout[0][0]
==================================================================================================
Total params: 109,483,010
Trainable params: 769
Non-trainable params: 109,482,241
__________________________________________________________________________________________________
```

شکل 8-2 : خلاصه ای از مدل به دست آمده

گام بعدی نیز تدوین مدل[1] به دست آمده خواهد بود :

Model compiling

```
[ ]  METRICS = [
            tf.keras.metrics.BinaryAccuracy(name='accuracy'),
            tf.keras.metrics.Precision(name='precision'),
            tf.keras.metrics.Recall(name='recall')
     ]

     model.compile(optimizer='adam',
      loss='binary_crossentropy',
      metrics=METRICS)
```

شکل 9-2 : کامپایل کردن مدل به دست آمده

[1] Model compiling

در مرحله ی بعدی نیاز داریم مدل به دست آمده را بر روی دادگان آموزش , برای متون انگلیسی در طی 10 دوره[1] فیت کنیم :

English corpus

```
[ ] model.fit(X_train_en, y_train_en, epochs=10)

    Epoch 1/10
    394/394 [==============================] - 143s 319ms/step - loss: 0.5604 - accuracy: 0.5494 - precision: 0.9510 - recall: 0.7400
    Epoch 2/10
    394/394 [==============================] - 133s 339ms/step - loss: 0.4781 - accuracy: 0.6012 - precision: 0.9790 - recall: 0.8063
    Epoch 3/10
    394/394 [==============================] - 136s 346ms/step - loss: 0.4583 - accuracy: 0.6104 - precision: 0.9744 - recall: 0.8068
    Epoch 4/10
    394/394 [==============================] - 136s 346ms/step - loss: 0.4506 - accuracy: 0.6114 - precision: 0.9768 - recall: 0.8059
    Epoch 5/10
    394/394 [==============================] - 136s 346ms/step - loss: 0.4444 - accuracy: 0.6072 - precision: 0.9701 - recall: 0.7963
    Epoch 6/10
    394/394 [==============================] - 136s 346ms/step - loss: 0.4376 - accuracy: 0.6167 - precision: 0.9686 - recall: 0.8049
    Epoch 7/10
    394/394 [==============================] - 136s 346ms/step - loss: 0.4390 - accuracy: 0.6098 - precision: 0.9728 - recall: 0.7826
    Epoch 8/10
    394/394 [==============================] - 136s 346ms/step - loss: 0.4292 - accuracy: 0.6183 - precision: 0.9742 - recall: 0.8013
    Epoch 9/10
    394/394 [==============================] - 136s 346ms/step - loss: 0.4282 - accuracy: 0.6111 - precision: 0.9738 - recall: 0.7839
    Epoch 10/10
    394/394 [==============================] - 136s 346ms/step - loss: 0.4234 - accuracy: 0.6263 - precision: 0.9741 - recall: 0.8039
    <tensorflow.python.keras.callbacks.History at 0x7ff11e7c4e10>
```

شکل 10-2 : فیت کردن مدل بر روی دادگان انگلیسی در طی 10 ایپاک

و معیار های دقت[2] , صحت[3] , فراخوانی[4] و خطا[5] در طی 10 ایپاک به دست آمده اند.

در نهایت نیز دقت مدل را بر روی دادگان آزمون بررسی می کنیم :

```
[ ] y_predicted = model.predict(X_test_en)
    y_predicted = y_predicted.flatten()
    for Counter in range(0,len(y_predicted)):
      if y_predicted[Counter] > 2/3 :
        y_predicted[Counter] = 1
      if y_predicted[Counter] < 1/3 :
        y_predicted[Counter] = 0
      if((y_predicted[Counter] < 2/3)and(y_predicted[Counter] > 1/3)) :
        y_predicted[Counter] = 0.5
    T_class = 0
    F_class = 0
    y_test_en[574] = 1
    for Item in range(0,len(y_predicted)):
      if y_test_en[Item] == y_predicted[Item]:
        F_class = F_class + 1
      else:
        T_class = T_class + 1
    print('Accuracy for English corpus is : ',(T_class/(T_class+F_class)*100),' %')

    Accuracy for English corpus is :  67.72878844016303  %
```

شکل 11-2 : عملکرد مدل بر روی دادگان آزمون

[1] Epoch
[2] Accuracy
[3] Precision
[4] Recall
[5] Loss

2- Persian corpus

در این قسمت نیز مشابه آنچه در قسمت قبل دیدیم , انجام می دهیم با این تفاوت که در این قسمت از مدل ParsBERT (مشابه قسمت سوم سوال اول) استفاده خواهیم کرد و نتایج را طی 10 ایپاک مشاهده می کنیم :

Persian corpus

```
[ ] model.fit(X_train_fa, y_train_fa, epochs=10)

    Epoch 1/10
    394/394 [==============================] - 145s 369ms/step - loss: 0.6524 - accuracy: 0.4606 - precision: 0.8122 - recall: 0.6229
    Epoch 2/10
    394/394 [==============================] - 138s 350ms/step - loss: 0.6419 - accuracy: 0.4705 - precision: 0.8186 - recall: 0.6297
    Epoch 3/10
    394/394 [==============================] - 138s 349ms/step - loss: 0.6277 - accuracy: 0.4815 - precision: 0.8316 - recall: 0.6320
    Epoch 4/10
    394/394 [==============================] - 138s 350ms/step - loss: 0.6212 - accuracy: 0.4864 - precision: 0.8381 - recall: 0.6397
    Epoch 5/10
    394/394 [==============================] - 138s 350ms/step - loss: 0.6102 - accuracy: 0.4997 - precision: 0.8500 - recall: 0.6414
    Epoch 6/10
    394/394 [==============================] - 137s 349ms/step - loss: 0.6089 - accuracy: 0.4977 - precision: 0.8450 - recall: 0.6419
    Epoch 7/10
    394/394 [==============================] - 137s 349ms/step - loss: 0.6025 - accuracy: 0.5010 - precision: 0.8497 - recall: 0.6451
    Epoch 8/10
    394/394 [==============================] - 138s 349ms/step - loss: 0.5997 - accuracy: 0.5080 - precision: 0.8547 - recall: 0.6532
    Epoch 9/10
    394/394 [==============================] - 138s 349ms/step - loss: 0.5950 - accuracy: 0.5101 - precision: 0.8559 - recall: 0.6445
    Epoch 10/10
    394/394 [==============================] - 138s 350ms/step - loss: 0.5935 - accuracy: 0.5104 - precision: 0.8571 - recall: 0.6469
    <tensorflow.python.keras.callbacks.History at 0x7ff042242f50>
```

شکل 12-2 : فیت کردن مدل بر روی دادگان فارسی در طی 10 ایپاک

و معیار های دقت , صحت , فراخوانی و خطا در طی 10 ایپاک به دست آمده اند.

در نهایت نیز دقت مدل را بر روی دادگان آزمون بررسی می کنیم :

```
[ ] y_predicted = model.predict(X_test_fa)
    y_predicted = y_predicted.flatten()
    for Counter in range(0,len(y_predicted)):
      if y_predicted[Counter] > 2/3 :
        y_predicted[Counter] = 1
      if y_predicted[Counter] < 1/3 :
        y_predicted[Counter] = 0
      if((y_predicted[Counter] < 2/3)and(y_predicted[Counter] > 1/3)) :
        y_predicted[Counter] = 0.5
    T_class = 0
    F_class = 0
    y_test_fa[2110] = 1
    for Item in range(0,len(y_predicted)):
      if y_test_fa[Item] == y_predicted[Item]:
        F_class = F_class + 1
      else:
        T_class = T_class + 1
    print('Accuracy for Persian corpus is : ',(T_class/(T_class+F_class)*100),' %')

    Accuracy for Persian corpus is :  67.02482400889218  %
```

شکل 13-2 : عملکرد مدل بر روی دادگان آزمون

3- Both English and Persian corpus

در این قسمت نیز دقیقا روندی مشابه با دو قسمت قبل خواهیم داشت با دو تفاوت مهم :

اول این که ابتدا نیاز داریم داده های انگلیسی و فارسی را با هم استفاده کنیم (همان طور که در صورت سوال نیز اشاره شده است با استفاده از تگ <SEP> این کار را انجام می دهیم)

و تفاوت دوم این که به علت دوزبانه بودن دادگان از مدل XLM-RoBERTa استفاده خواهیم کرد. (مشابه با کاری که در قسمت دوم سوال اول انجام دادیم)

Both English and Persian corpus

```
[ ] model.fit(X_train_en_fa, y_train_en_fa, epochs=10)

    Epoch 1/10
    394/394 [==============================] - 137s 328ms/step - loss: 0.6509 - accuracy: 0.4419 - precision: 0.8248 - recall: 0.6394
    Epoch 2/10
    394/394 [==============================] - 135s 342ms/step - loss: 0.5645 - accuracy: 0.5519 - precision: 0.9276 - recall: 0.7709
    Epoch 3/10
    394/394 [==============================] - 139s 352ms/step - loss: 0.5449 - accuracy: 0.5637 - precision: 0.9496 - recall: 0.7901
    Epoch 4/10
    394/394 [==============================] - 139s 353ms/step - loss: 0.5277 - accuracy: 0.5723 - precision: 0.9546 - recall: 0.7915
    Epoch 5/10
    394/394 [==============================] - 139s 353ms/step - loss: 0.5150 - accuracy: 0.5791 - precision: 0.9622 - recall: 0.7864
    Epoch 6/10
    394/394 [==============================] - 139s 352ms/step - loss: 0.5033 - accuracy: 0.5914 - precision: 0.9656 - recall: 0.7895
    Epoch 7/10
    394/394 [==============================] - 139s 352ms/step - loss: 0.5017 - accuracy: 0.5868 - precision: 0.9600 - recall: 0.8029
    Epoch 8/10
    394/394 [==============================] - 139s 352ms/step - loss: 0.4980 - accuracy: 0.5907 - precision: 0.9637 - recall: 0.7900
    Epoch 9/10
    394/394 [==============================] - 139s 352ms/step - loss: 0.4952 - accuracy: 0.5894 - precision: 0.9616 - recall: 0.8009
    Epoch 10/10
    394/394 [==============================] - 139s 352ms/step - loss: 0.4881 - accuracy: 0.5981 - precision: 0.9648 - recall: 0.7966
    <tensorflow.python.keras.callbacks.History at 0x7fef3f534510>
```

شکل 14-2 : فیت کردن مدل بر روی دادگان انگلیسی و فارسی در طی 10 ایپاک

و معیار های دقت , صحت , فراخوانی و خطا در طی 10 ایپاک به دست آمده اند.

در نهایت نیز دقت مدل را بر روی دادگان آزمون بررسی می کنیم :

```
[ ] y_predicted = model.predict(X_test_en_fa)
    y_predicted = y_predicted.flatten()
    for Counter in range(0,len(y_predicted)):
      if y_predicted[Counter] > 2/3 :
        y_predicted[Counter] = 1
      if y_predicted[Counter] < 1/3 :
        y_predicted[Counter] = 0
      if((y_predicted[Counter] < 2/3)and(y_predicted[Counter] > 1/3)) :
        y_predicted[Counter] = 0.5
    T_class = 0
    F_class = 0
    y_test_en_fa[1836] = 1
    for Item in range(0,len(y_predicted)):
      if y_test_en_fa[Item] == y_predicted[Item]:
        F_class = F_class + 1
      else:
        T_class = T_class + 1
    print('Accuracy for both English and Persian corpus is : ',(T_class/(T_class+F_class)*100),' %')

    Accuracy for both English and Persian corpus is :  65.98740274175621  %
```

شکل 15-2 : عملکرد مدل بر روی دادگان آزمون

همان طور که مشاهده کردیم , بهترین عملکرد مربوط به مدل اول بود , هنگامی که با استفاده از BERT دادگان انگلیسی را طبقه بندی کردیم. همچنین کمترین دقت مربوط به زمانی بود که با استفاده از ParsBERT دادگان فارسی را طبقه بندی کردیم. علت امر نیز در آن است که وقتی از مدل قدرتمندی همچون BERT استفاده می کنیم (که بر روی چندین زبان و چند ترابایت داده آموزش داده شده است) انتظار داریم دقت بیشتری نسبت به حالتی که از ParsBERT که در واقع زیرشاخه ای از مدل BERT می باشد , استفاده می کنیم , حاصل شود.

علت این که از مدل چند زبانه نیز دقت بیشتری نسبت به مدل تک زبانه (فارسی) حاصل می شود در این است که حجم دیتای در دسترس در مدل چند زبانه در واقع دو برابر حجم دیتای در دسترس در مدل تک زبانه است و ما , هم می توانیم از بازنمایی [1] ورودی در زبان فارسی استفاده کنیم هم از بازنمایی ورودی در زبان انگلیسی استفاده خواهیم کرد و بنابر این مدل این امکان را دارد که با یک مصالحه ای بین این بازنمایی ها , اطلاعات به مراتب بیشتری نسبت به حالت تک زبانه دریافت کند و Embedding با کیفت تری را تولید کرده و مدل به دقت بالاتری دست پیدا می کند.

[1] Representation

Part 3- Cross-lingual zero-shot transfer learning (Optional)

Q1

استفاده از یک مدل چند زبانه (مثل XLM-RoBERTa) و فیت کردن آن بر روی دادگان انگلیسی و ارزیابی مدل بر روی زبان فارسی , طبیعتا نتیجه ضعیف تری نسبت به حالت قبل خواهد داد ولی احتمالا نسبت به حالتی که از ParsBERT بر روی زبان فارسی استفاده کردیم نتیجه بهتری خواهیم گرفت.

علت این امر نیز این است که اولا مدل های چند زبانه ای همچون XLM-RoBERTa نسبت به مدل تک زبانه فارسی همچون ParsBERT , قدرتمند تر بوده (مدلی مثل XLM-RoBERTa با 100 زبان مختلف و بر روی چند ترابایت داده آموزش داده شده است که قطعا از مدلی مثل ParsBERT که برای زبان فارسی آموزش دیده شده است , از قدرت بیشتری برخوردار است) و در درجه ی دوم , درست است که مدل بر روی دادگان انگلیسی فیت شده است اما این به این معنی نیست که دیگر نمی توان از بازنمایی های به دست آمده از مدل انگلیسی بر روی دیتاست فارسی استفاده کرد , زیرا هر قدر هم که دو زبان از هم متفاوت باشند باز هم , بازنمایی های به دست آمده در دو زبان بیگانه از هم نخواهند بود و قطعا دارای یک قرابت معنایی نزدیکی نسبت به هم خواهند بود و می توان از بازنمایی[1] های به دست آمده در زبان انگلیسی برای فارسی هم استفاده کرد (طبیعتا به خوبی دقت سابق نخواهد بود)

Q2

در این قسمت نیز همانند سوال اول و دوم در بخش قبل , مدل را پیاده سازی کرده و نتیجه را بررسی می کنیم. با این تفاوت که در این قسمت مدل را (در اینجا XLM-RoBERTa) بر روی دادگان انگلیسی فیت کرده و ارزیابی را روی دادگان فارسی انجام می دهیم و نتایج را طی 10 ایپاک مشاهده می کنیم :

Persian corpus classification using English corpus

```
[ ] model.fit(X_train_en, y_train_en, epochs=10)

    Epoch 1/10
    394/394 [==============================] - 132s 334ms/step - loss: 0.4820 - accuracy: 0.5898 - precision: 0.9613 - recall: 0.7863
    Epoch 2/10
    394/394 [==============================] - 136s 344ms/step - loss: 0.4540 - accuracy: 0.6045 - precision: 0.9687 - recall: 0.7922
    Epoch 3/10
    394/394 [==============================] - 136s 346ms/step - loss: 0.4457 - accuracy: 0.6081 - precision: 0.9686 - recall: 0.7856
    Epoch 4/10
    394/394 [==============================] - 136s 346ms/step - loss: 0.4398 - accuracy: 0.6112 - precision: 0.9698 - recall: 0.7878
    Epoch 5/10
    394/394 [==============================] - 136s 346ms/step - loss: 0.4367 - accuracy: 0.6132 - precision: 0.9687 - recall: 0.7883
    Epoch 6/10
    394/394 [==============================] - 136s 346ms/step - loss: 0.4326 - accuracy: 0.6146 - precision: 0.9726 - recall: 0.7862
    Epoch 7/10
    394/394 [==============================] - 136s 346ms/step - loss: 0.4305 - accuracy: 0.6140 - precision: 0.9695 - recall: 0.7828
    Epoch 8/10
    394/394 [==============================] - 136s 346ms/step - loss: 0.4287 - accuracy: 0.6150 - precision: 0.9696 - recall: 0.7863
    Epoch 9/10
    394/394 [==============================] - 137s 347ms/step - loss: 0.4271 - accuracy: 0.6162 - precision: 0.9709 - recall: 0.7891
    Epoch 10/10
    394/394 [==============================] - 143s 363ms/step - loss: 0.4246 - accuracy: 0.6183 - precision: 0.9716 - recall: 0.7794
    <tensorflow.python.keras.callbacks.History at 0x7ff0417b1050>
```

شکل 1-3 : فیت کردن مدل بر روی دادگان انگلیسی در طی 10 ایپاک

و معیار های دقت , صحت , فراخوانی و خطا در طی 10 ایپاک به دست آمده اند.

[1] Representation

در نهایت نیز دقت مدل را بر روی دادگان آزمون بررسی می کنیم :

```
[ ] y_predicted = model.predict(X_test_fa)
    y_predicted = y_predicted.flatten()
    for Counter in range(0,len(y_predicted)):
      if y_predicted[Counter] > 2/3 :
        y_predicted[Counter] = 1
      if y_predicted[Counter] < 1/3 :
        y_predicted[Counter] = 0
      if((y_predicted[Counter] < 2/3)and(y_predicted[Counter] > 1/3)) :
        y_predicted[Counter] = 0.5
    T_class = 0
    F_class = 0
    y_test_fa[2110] = 1
    for Item in range(0,len(y_predicted)):
      if y_test_fa[Item] == y_predicted[Item]:
        F_class = F_class + 1
      else:
        T_class = T_class + 1
    print('Accuracy for Persian corpus using English corpus is : ',(T_class/(T_class+F_class)*100),' %')

    Accuracy for Persian corpus using English corpus is :  66.95072248981104  %
```

شکل 2-3 : عملکرد مدل بر روی دادگان آزمون فارسی

همان طور که از قبل انتظارداشتیم دقت به دست آمده از مدل ParsBERT بر روی دادگان فارسی بیشتر شد اما نسبت به حالتی که از مدل BERT برای دادگان انگلیسی استفاده کردیم دقت کمی کمتر شد که طبیعی است.

در نهایت با توجه به توضیحات مفصلی که در قسمت قبل نیز داده شد , علت این که مدل می تواند با دقت قابل قبولی دادگان فارسی را بر روی مدلی که دادگان انگلیسی فیت شده است , ارزیابی کند , قرابت معنایی بین بازنمایی های به دست آمده در دو زبان است.

به این معنی که اگر چه مدل بر روی دادگان انگلیسی فیت شده است اما این به آن معنی نیست که نمی توان از بازنمایی ها و Embedding های به دست آمده زبان انگلیسی در ارزیابی زبان فارسی استفاده کرد , بلکه این بازنمایی ها با یکدیگر بیگانه نیستند و قرابت خوبی با یکدیگر دارند و درنتیجه دقت مدل به دست آمده مطلوب است. (حتی بیشتر از حالتی که از مدل ParsBERT بر روی دادگان فارسی استفاده کردیم)

Q3

با استفاده از نتایج قسمت قبل و همچنین مطالعه ی یک مقاله به این نتیجه می رسیم که زمانی استفاده از zero-shot cross-lingual transfer learning برای ما قابل بحث است که داده هایی که در زبان مقصد داریم (زبانی که می خواهیم طبقه بندی را بر روی آن انجام دهیم) محدود باشد و به اصطلاح Lack of data رخ بدهد , یک مدل چند زبانه را بر روی یک زبان که دارای محدودیت دیتا نیست (مثلا انگلیسی) فیت می کنیم و در ادامه مدل به دست آمده را روی دادگان زبان مقصد ارزیابی می کنیم که هر چه دادگان مبدا بیشتری در اختیار داشته باشیم و زبان مبدا و مقصد قرابت بیشتری با هم داشته باشند نتیجه بهتری خواهیم گرفت.

Attachment

تمامی فایل های ipynb در پوشه ی Codes موجود می باشد. همچنین تمامی کد ها در محیط colab اجرا و تست شده اند.

با توجه به شرایط و پیاده سازی های موجود در این پروژه و بعضا سنگین بودن حجم پردازشات , اجرای برخی سل ها زمان بر خواهد بود (زمان تقریبی اجرای این سل ها بعضا در گزارش قید شده است) , لذا پیشاپیش از صبر و تحمل شما برای اجرا شدن کدها سپاسگزارم.

همچنین در برخی از قسمت ها ملاحظات لازم برای اجرای کد ها و سل ها آورده شده است.