به نام خدا

دانشگاه تهران

دانشکدگان فنی

دانشکده مهندسی برق و کامپیوتر

پردازش زبان‌های طبیعی

تمرین چهارم

نام و نام خانوادگی : حسین سیفی

شماره دانشجویی : ۸۱۰۱۰۰۳۸۶

خرداد ۱۴۰۱

فهرست

سوال ۱- Textual Entailment

بخش ۱

دیتاست مورد نظر که از نظر تقسیم بندی به مجموعه‌های آموزش، تست و اعتبار سنجی کمی عجیب به نظر می‌رسد نیاز به پیش پردازش جزئی دارد. در دیتاست انتخاب شده برای آموزش و تست این سوال برچسب‌هایی با مقادیر نامناسب که در این پروژه طبقه بندی جایی ندارند وجود دارد. برچسب‌هایی مانند - ، xx و ... در مجموعه دادگان تست و آموزش وجود دارند و همچنین داده‌های ستون Sent1 و Sent2 متناظر با آن‌ها نیز دارای مقادیر معتبری نیستند و به نظر نمی‌رسد که قابل اصلاح باشند بنابراین داده‌هایی با برچسب نامعتبر را از دیتاست حذف کردیم.

همچنین پیش از ایجاد و آموزش مدل نیاز داریم تا فرمت ورودی مناسب را برای شبکه طراحی شده ایجاد کنیم. بدین منظور به کمک دستورات زیر جمله اول(Sent1) و دوم(Sent2) را با استفاده از توکن <SEP> به یکدیگر متصل کنیم. اقدامی مشابه با آنچه در دستور زیر برای مجموعه آموزش انجام شده است برای مجموعه‌های تست و اعتبارسنجی نیز انجام می‌گیرد.

```
train['conc'] = [(train.iloc[i]['sent1']+'<SEP>'+train.iloc[i]['sent2']) for i in range(len(train))]
```

سپس باید فرمت برچسب مجموعه‌های داده را به شکل One-hot تبدیل کنیم. بدین منظور ابتدا برچسب‌هایی که از نوع رشته هستند را به اعداد تبدیل می‌کنیم و سپس با استفاده از تابع to_categorical به فرمت مورد نظر تبدیل می‌کنیم. موارد ذکر شده به کمک قطعه کد زیر انجام شده‌اند:

```
encoded_dict = {'c':0, 'n':1, 'e':2}

train['label'] = train.label.map(encoded_dict)

y_train = to_categorical(train['label'])
```

بخش ۲

برای پیاده سازی این بخش نیز همانند دیگر بخش‌های این پروژه از یک معماری با سه لایه که لایه اول، لایه ورودی با اندازه ۱۲۸ و تابع فعالساز ReLu، لایه میانی با اندازه ۳۲ و تابع فعالساز Relu و لایه سوم، لایه خروجی با اندازه ۳ و تابع فعالساز Softmax است استفاده شده است. جزییات این مدل در بخش اول سوال ۲ به صورت کامل قابل مشاهده است. در پیاده سازی این بخش از مدل از پیش آموزش دیده xlm_roberta استفاده شده است که با کمک دستور زیر بارگزاری می‌شود. لازم به ذکر است که فراخوانی مدل از پیش آموزش دیده تنها وجه تفاوت مدل آموزش دیده در بخش‌های مختلف این پروژه می‌باشد.

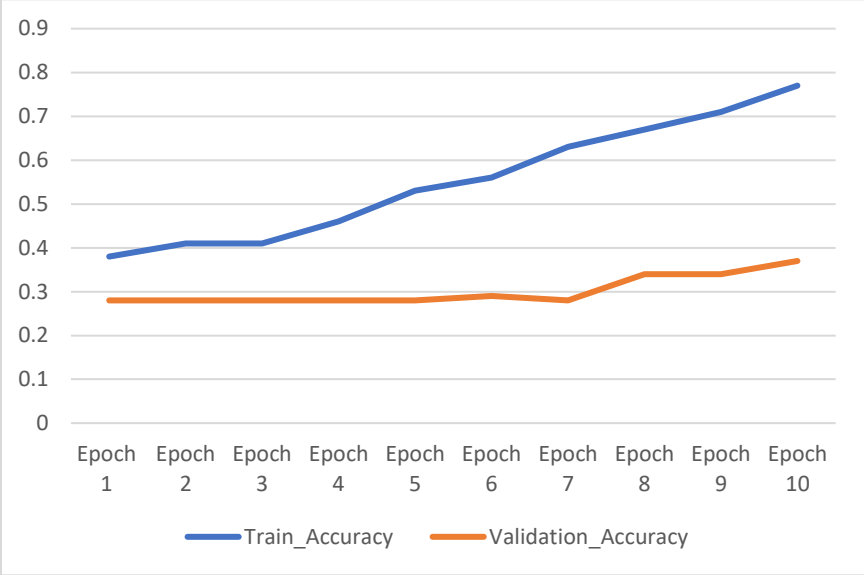
```
from transformers import RobertaTokenizer, RobertaModel, TFXLMRobertaModel

tokenizer = AutoTokenizer.from_pretrained('xlm-roberta-base')

model = TFXLMRobertaModel.from_pretrained("jplu/tf-xlm-roberta-base")
```

این مدل نیز همانند مدل ParsBert آموزش دیده در بخش سوم همین سوال در طول Fine-Tune شدن در هر یک از Epochها نوساناتی در مقدار دقت و Loss دیده می‌شود و در عمل نشانی از آموزش دیدن در این مدل دیده نمی‌شود. در جدول زیر مقادیر معیارهای ارزیابی این مدل بر روی دادگان تست قابل مشاهده است:

	Precision	Recall	F1-score	Support
C	0.33	0.20	0.25	561
N	0.61	0.18	0.28	502
E	0.43	0.84	0.57	610
Accuracy			0.42	1673
Macro Avg	0.46	0.40	0.36	1673
Weighted Avg	0.45	0.42	0.47	1673

همانطور که در جدول فوق دیده می‌شود معیار Accuracy مقداری کمی بهتر از مقدار تصادفی برای سه کلاس دارد و مقدار Precision برای کلاس N و مقدار Recall برای کلاس E نسبت به بقیه کلاس‌ها بالاتر است.

در نمودار زیر مقدار Accuracy بر روی داده‌های تست و اعتبارسنجی در پایان هر Epoch مشخص شده است:



نمودار 1

همانطور که در نمودار فوق مشخص است دقت داده‌های اعتبارسنجی تا پایان Epoch پنجم کاملا برابر است و کوچکترین تغییری نداشته است و پس از Epoch پنجم با رخ دادن شکست در نمودار مقدار دقت اندکی بهتر شد و در نهایت بر روی دادگان تست عملکردی بهتر از داده‌های اعتبارسنجی داشته است. این در حالی است که دقت روی دادگان آموزش به صورت مداوم در حال تغییر بوده است و نشان می‌دهد که این مدل از توانایی عمومی سازی(Generaliztion) ضعیفی بر روی دادگان آموزش برخوردار است و در نتیجه حتی قبل از تست و صرفا با بررسی نمودار فوق می‌توان متوجه شد که انتظار پیش‌بینی با دقت بالا از این مدل نمی‌توان داشت.

بخش ۳

در این بخش از مدل از پیش آموزش دیده Parsbert استفاده شده است. این بخش از سوال ۱ نیز از معماری مشابهی با بقیه بخش‌های سوال ۱ و ۲ بهره می‌برد. تنها تفاوت این معماری با معماری دیگر شبکه‌های عصبی این گزارش وجود لایه ورودی با اندازه ۱۱۹ می‌باشد که به دلیل عدم موفقیت تبدیل دادگان اعتبارسنجی به Tensorهایی با طول ۱۲۸ مجبور به ایجاد این تغییر شدیم. علی رغم اینکه در سوال ۲ دادگان مورد نیاز با صحت کامل به بردارهایی با طول مورد نیاز تبدیل شدند، در این سوال با ایجاد

تغییراتی در مجموعه دادگان نیز موفق به انجام این کار نشدیم و در نهایت با تبدیل اندازه لایه ورودی به ۱۱۹ و کاهش اندازه بردارهای تمامی دادگان به ۱۱۹موفق شدیم شبکه را به صورت کامل Fine-Tune کنیم. لازم به ذکر است که با توجه به انجام Padding و Truncating با این کاهش اندازه هیچ خللی در کار شبکه ایجاد نشده است و هیچ بخشی از دادگان از دسترس مدل برای استفاده دور نمی‌ماند. این مدل علاوه بر لایه ورودی با اندازه ۱۱۹ شامل لایه خروجی به تعداد کلاس‌های موجود در دیتاست یعنی ۳ که هر یک با هدف یکی از برچسب‌های C، N و E ایجاد شده‌اند با تابع فعالساز Softmax و یک لایه میانی با اندازه ۳۲ با تابع فعالساز ReLu نیز در این معماری قرار دارد.

مدل از پیش آموزش دیده به کمک دستورات زیر بارگزاری می‌شوند:

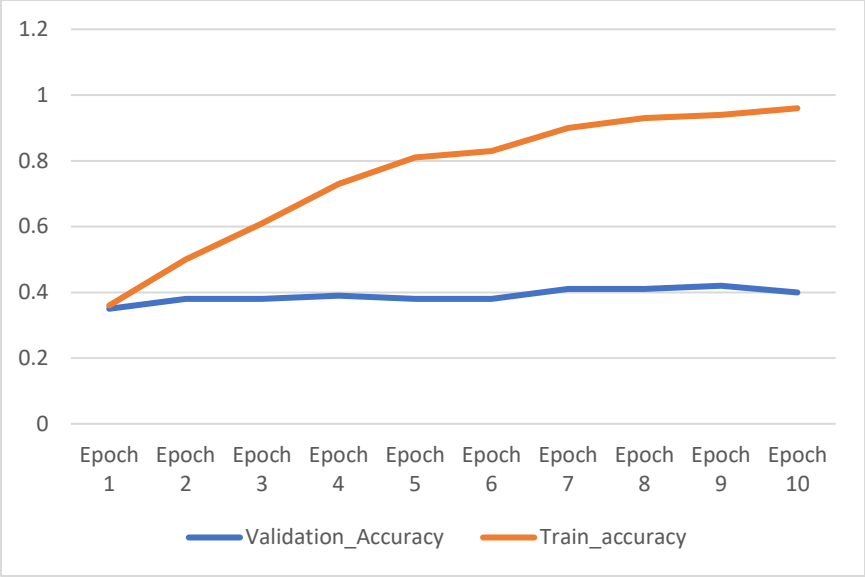
```
from transformers import AutoConfig, AutoTokenizer, TFAutoModel

config = AutoConfig.from_pretrained("HooshvareLab/bert-base-parsbert-uncased")
tokenizer = AutoTokenizer.from_pretrained("HooshvareLab/bert-base-parsbert-uncased")
model = TFAutoModel.from_pretrained("HooshvareLab/bert-base-parsbert-uncased")
```

برای Fine-Tune کردن وظیفه مورد نظر روی این مدل از ۱۰ Epoch به صورت آزمایشی استفاده شد تا نتیجه بررسی شود و در صورت نیاز از تعداد بیشتر یا کمتری Epoch در اجراهای بعدی استفاده شود اما به توجه به تغییرات دقت و Loss در هر Epoch اجرا شده به نظر نمی‌رسد به تعداد بیشتری احتیاج داشته باشیم. در طول اجرای هر Epoch مقادیر Loss و دقت رصد شدند(که متاسفانه قابل به اشتراک گذاشتن نیست) و این مقادیر به صورت نامنظمی نوساناتی را داشتند و تغییر می‌کردند بدین صورت که هر دوی این مقادیر ذکر شده جهش‌های را به بالا و پایین داشتند و نشانی از ساختار در بین این تغییرات دیده نمی‌شد. این تغییرات به گونه‌ای بود که حتی پس از پایان یک Epoch و بررسی مقادیر Accuracy و Loss بر روی داده‌های اعتبار سنجی، پس از اجرای Epoch بعدی و انجام گام‌هایی در جهت بهبود نتیجه، حتی در مواقعی مقدار دقت کاهش نیز می‌یافت. جدول زیر مقدار Accuracy و Loss را برای داده‌های اعتبارسنجی نشان می‌دهد:

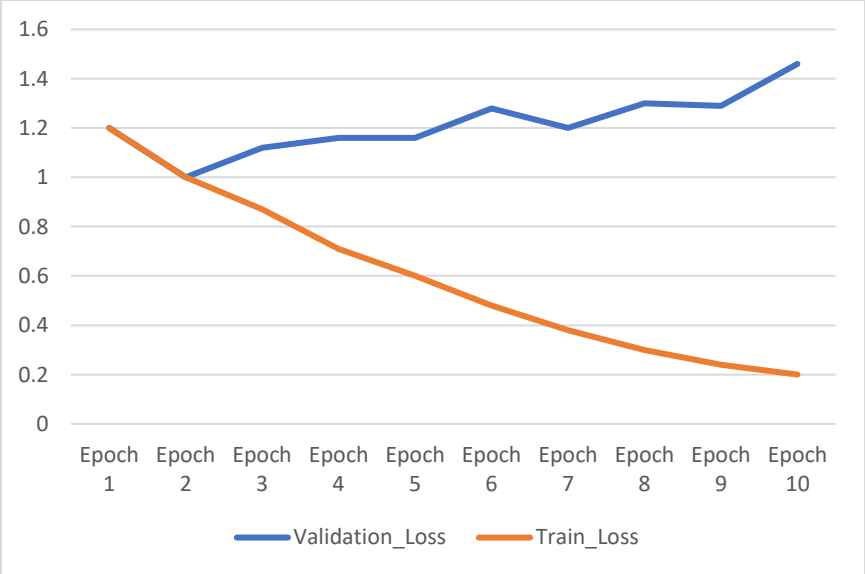
	Precision	Recall	F1-score	Support
C	0.34	0.17	0.22	561
N	0.49	0.43	0.46	502
E	0.49	0.71	0.55	610
Accuracy			0.44	1673
Macro Avg	0.43	0.44	0.41	1673
Weighted Avg	0.43	0.44	0.41	1673

همانطور که در جدول فوق دیده می‌شود دقت نهایی مدل Parsbert کمی بهتر از مدل از پیش آموزش دیده xlm_roberta می‌باشد اما همچنان نتایج کمی بهتر از حالت تصادفی برای سه برچسب می‌باشد و دستیابی به این اعداد و ارقام را نمی‌توان موفقتی در زمینه رده بندی ارتباط بین جملات در پردازش زبان طبیعی دانست.

در نمودارهای زیر مقادیر دقت و Loss را برای داده‌های آموزش و اعتبار سنجی پس از هر Epoch مشاهده می‌کنید:



نمودار 2



نمودار 3

در نمودار دقت مشخص است که با اجرای هر Epoch دقت روی مجموعه آموزش افزایش می‌یابد اما تفاوت دقت روی داده‌های اعتبار سنجی بسیار ناچیز و قابل چشم پوشی است. به عبارتی دیگر مدل هیچگونه پیشرفتی در پیش‌بینی برچسب داده‌هایی که تا به حال ندیده است در طی Epochهای متعدد ندارد و حتی اگر اجرای Fine-tunning را پس از Epoch اول متوقف کنیم چیزی را از دست نخواهیم داد. این موضوع در نمودار Loss شکل جدی تری به خود می‌گیرد و این نمودار نشان می‌دهد که نه تنها این مدل در طی Epochهای متوالی پیشرفتی ندارد، بلکه از نظر مقدار Loss پس از Epoch دوم پسرفت هم خواهد داشت.

کلام آخر در مقایسه دو مدل فوق:

همانگونه در که در قسمت‌های قبل توضیح داده شد، مدل از پیش آموزش دیده xlm_roberta به خوبی برای وظیفه طبقه بندی رابطه بین جملات Fine-Tune نمی‌شود و نتیجه‌ای نسبتا تصادفی ارائه می‌دهد که شباهت کمی به یک مدل آموزش دیده دارد و خطای زیادی در پیش بینی برچسب داده‌ها تست دارد از طرفی دیگر مدل Fine-Tune شده با استفاده از مدل از پیش آموزش دیده ParsBert دقت کمی بهتر است و دقت متوسطی را برای هر یک از کلاس‌ها ارائه می‌دهد هر چند که این مدل نیز در روند Fine-Tune شدن عکس العمل مناسبی از خود نشان نمی‌دهد. بدون شک این وظیفه انتخاب شده(Textual Entailment) وظیفه سنگینی برای شبکه‌های عصبی فعلی محسوب می‌شود که به دلیل عدم وجود دیتاست مناسب و خوب یا عدم تطابق معماری‌های انتخاب شده برای این وظیفه به نتیجه مطلوبی دست نمی‌یابد. البته دلیل دیگر عدم ارائه نتایج مناسب توسط مدل‌های Fine-Tune شده می‌تواند تطابق نداشتن مدل از پیش آموزش دیده باشد و گواه این فرضیه نیز تغییر دقت و بهبود عملکرد مدلی که از Parsbert استفاده می‌کند نسبت به مدل xlm_roberta است که تنها با تغییر مدل Pretrained، عملکرد مدل(هر چند کم) بهبود یافت.

سوال ۲- Multilingual Classification

بخش ۱

برای پیاده سازی رده بندی چند زبانه به وسیله شبکه‌های عصبی عمیق از مدل از پیش آموزش دیده(Pretrained) پایه برت به صورت حساس به حروف بزرگ و کوچک(bert_base_cased) استفاده شده است.

ابتدا نیاز داریم تا مدل از پیش آموزش دیده و توکنایزر مورد نظر را از طریق بسته transformers بارگزاری کنیم. توکنایزر وظیفه تبدیل متن ورودی به توکن‌ها را دارا می‌باشد.

```
tokenizer = AutoTokenizer.from_pretrained('bert-base-cased')
bert = TFBertModel.from_pretrained('bert-base-cased')
```

پیش از آموزش مدل، ابتدا باید داده‌های ورودی به کمک توکنایزر به فرمت داده‌های ورودی مورد نیاز Bert تبدیل شوند. همانطور که در قطعه کد قبل قابل مشاهده است، برای تبدیل متن به فرمت ایده‌آل قصد داریم از توکنایزر Bert_base_cased استفاده کنیم. به کمک کد زیر داده‌های آموزش به این فرمت دلخواه تبدیل می‌شوند همچنین قطعه کدهای مشابهی برای تبدیل داده‌های تست و اعتبارسنجی(Validation) نوشته شده است:

```
x_train = tokenizer(

    text=train.source.tolist(),
    max_length=128,
    truncation=True,
    padding=True,
    return_tensors='tf',
    return_token_type_ids = False,
    return_attention_mask = True,
    verbose = True)
```

همانطور که در سوال خواسته شده است، توکنایزر فوق بردارهایی(Tensors) با حداکثر طول ۱۲۸ ایجاد می‌کند. خروجی تابع فوق همانطور که با مقداردهی به پارامترهای آن مشخص کرده‌ایم، دو بردار attention_mask و input_id برای هر یک از جملات ورودی(ستون source در دیتاست داده شده) است. Attention_mask که برداری از صفر و یک و هم اندازه با بردار input_id ایجاد می‌کند که با ایجاد تمایز بین کلمات موجود در جمله ورودی و حاشیه(padding) ایجاد شده برای یکسان سازی طول بردارها، در واقع به ما کمک می‌کند تا بتوانیم جملاتی با طول متفاوت را به عنوان ورودی به برت بدهیم و input_id شامل شناسه کلمات موجود در جمله ورودی با ترتیب همان جمله به همراه حاشیه(padding) مورد نیاز برای هم اندازه کردن بردارهای جملات می‌باشد. همچنین در پارامترهای توکنایزر مشخص شده است در صورتی که طول جمله از ۱۲۸ بیشتر باشد، آن‌را به بخش‌هایی با طول ۱۲۸ تقسیم کند.(Truncating)

همچنین نیاز داریم تا برچسب داده‌ها را به فرمت One-hot تبدیل کنیم بدین معنی که برچسب هر سطر یک بردار با طول ۳ است که درایه متناظر با کلاس آن برابر ۱ و دیگر درایه‌ها برابر صفر هستند. این کار به کمک قطعه کد زیر برای داده‌های آموزش انجام می‌گیرد و در کد اصلی برای دادگان اعتبار سنجی و تست نیز انجام گرفته است:

```
encoded_dict = {'bible':0, 'quran':1, 'mizan':2}

train['category'] = train.category.map(encoded_dict)

y_train = to_categorical(train.category)
```

در کد زیر معماری لایه‌های مورد استفاده مشخص شده است. ابتدا یک لایه که شامل Bert hidden state است با نام embedding به معماری اضافه می‌شود. در لایه بعدی GlobalMaxPool1D وجود دارد و برای ایجاد لایه‌های و برای ایجاد لایه‌های CNN از آن استفاده می‌کنیم. سپس یک لایه Dense با اندازه ۱۲۸ و تابع فعالساز ReLu، یک لایه Dropout با آستانه ۰.۱ و یک لایه Dense دیگر با اندازه ۳۲ و تابع فعالساز Relu موجود است. در نهایت یک لایه خروجی با اندازه ۳ و تابع فعالساز Softmax تعریف می‌شود و مدل با لایه‌های تعریف شده ایجاد می‌شود.

```
embeddings = bert(input_ids,attention_mask = input_mask)[0]

out = tf.keras.layers.GlobalMaxPool1D()(embeddings)
out = Dense(128, activation='relu')(out)
out = tf.keras.layers.Dropout(0.1)(out)
out = Dense(32,activation = 'relu')(out)
y = Dense(3,activation = 'softmax')(out)
model = tf.keras.Model(inputs=[input_ids, input_mask], outputs=y)
```

در مرحله بعد پارامترهای آموزش را مقداردهی می‌کنیم، توابع مورد استفاده برای Loss و Metric را به ترتیب با Categoricalcrossentropy و CategoricalAccuracy مشخص کردیم و در نهایت مدل را کامپایل می‌کنیم. نرخ آموزش(Learinng Rate) مطابق خواسته سوال با مقدار ۰.۰۰۰۰۳ مقداردهی شده است.

```
optimizer = Adam(

    learning_rate=3e-05,
    epsilon=1e-08,
    decay=0.01,
    clipnorm=1.0
    )

loss = CategoricalCrossentropy(from_logits = True)
metric = CategoricalAccuracy('balanced_accuracy'),

model.compile(
    optimizer = optimizer,
    loss = loss,
    metrics = metric)
```

در این مرحله مدل آماده Fine Tune شدن با استفاده از داده‌های آموزش و اعتبارسنجی به کمک داده‌های مربوطه در ۱۰ Epoch و با Batch_size برابر ۳۲ می‌باشد.

```
train_history = model.fit(

    x ={'input_ids':x_train['input_ids'],'attention_mask':x_train['attention_mask']} ,
    y = y_train,
    validation_data = ({'input_ids':x_valid['input_ids'],'attention_mask':x_valid['attention_mask']}, y_valid),
    epochs=10,
    batch_size=32
)
```

پس از Fine Tune کردن مدل آماده شده، می‌توان به ارزیابی آن پرداخت. ارزیابی به کمک تابع classification_report از کتابخانه sklearn و با استفاده از برچسب‌های پیش بینی شده برای داده‌های تست توسط مدل فوق انجام می‌شود. نتایج ارزیابی به شرح زیر می‌باشد:

	Precision	Recall	F1-score	Support
Bible	0.98	0.99	0.98	900
Quran	0.98	0.98	0.98	900
Mizan	0.99	0.99	0.99	900
Accuracy			0.98	2700
Macro Avg	0.98	0.98	0.98	2700
Weighted Avg	0.98	0.98	0.98	2700

دقت‌های به دست آمده مقادیر بسیار خوبی هستند و دقت مشاهده شده برای یک وظیفه(Task) مرتبط با پردازش متن و طبقه بندی متون مقداری ایده‌آل است که به لطف ابداع برت در سالیان اخیر و پیشرفت شبکه‌های عصبی محقق شده است.

در جدول زیر نیز مقدار Accuracy و Loss مجموعه داده اعتبار سنجی در پایان هر یک از Epoch ها نشان داده شده است:

	Validation_loss	Validation_accuracy
Epoch 1	0.0662	0.9833
Epoch 2	0.0689	0.9848
Epoch 3	0.0684	0.9837
Epoch 4	0.0699	0.9833
Epoch 5	0.0712	0.9841
Epoch 6	0.0728	0.983
Epoch 7	0.0724	0.9822
Epoch 8	0.0738	0.9841
Epoch 9	0.0741	0.9833
Epoch 10	0.0770	0.9837

همانطور که مشاهده می‌شود مقدار Loss تا Epoch چهارم نزولی بوده است و در Epoch پنجم به بعد صعودی شده است. همچنین مقدار Accuracy نیز تا Epoch پنجم صعودی بوده است و پس از آن نزولی شده است. بنابراین در هر لحظه بعد از Epoch پنجم می‌توانستیم توقف زود هنگام(Early stopping) را انجام دهیم اما بنابر خواسته سوال اجازه دادیم تا فرآیند به طور کامل انجام شود.

بخش ۲

در این بخش به طبقه بندی متون فارسی به کمک مدل از پیش آموزش دیده Parsbert می‌پردازیم. تمامی بخش‌های این طبقه بند اعم از معماری شبکه عصبی، تبدیل دیتاست به بردار، آموزش مدل و نحوه بررسی دقت خروجی در این بخش با بخش قبل کاملا مشابه است و تنها تفاوت مربوط به نحوه بارگزاری مدل از پیش آموزش دیده Parsbert است و ورودی این شبکه نیز از ستون source به ستون targets از دیتاست تغییر می‌کند. این مدل به روش زیر بارگزاری می‌شود:

```
from transformers import AutoConfig, AutoTokenizer, TFAutoModel

config = AutoConfig.from_pretrained("HooshvareLab/bert-base-parsbert-uncased")
tokenizer = AutoTokenizer.from_pretrained("HooshvareLab/bert-base-parsbert-uncased")
parsbert = TFAutoModel.from_pretrained("HooshvareLab/bert-base-parsbert-uncased")
```

پس از انجام عمل Fine-Tuning نتایج به دست آمده به شرح زیر می‌باشد:

	Precision	Recall	F1-score	Support
Bible	0.97	0.96	0.97	900
Quran	0.97	0.98	0.97	900
Mizan	0.97	0.96	0.96	900
Accuracy			0.97	2700
Macro Avg	0.97	0.97	0.97	2700
Weighted Avg	0.97	0.97	0.97	2700

همانطور که در جدول فوق دیده می‌شود مطابق آنچه انتظار داشتیم مقدار تمامی معیارهای ارزیابی با استفاده از جملات فارسی نسبت به جملات انگلیسی کاهش یافته‌اند البته قابل ذکر است که دقت به دست آمده به کمک مدل آموزش دیده Parsbert نیز خوب و قابل قبول است اما در شرایطی که دسترسی به داده‌های انگلیسی مربوطه وجود داشته باشد، به نظر می‌رسد که استفاده از آن به نتیجه بهتری منتهی شود. این کاهش دقت مدل فارسی نسبت به مدل انگلیسی می‌تواند به دلیل این باشد که زبان فارسی از نظر ساخت‌واژی و نحوی بسیار پیچیده‌تر از زبانی مثل انگلیسی است و به طبع جمع آوری دیتاست مناسب، انتخاب مدل مناسب و آموزش آن می‌تواند با چالش بزرگتری نسبت به بسیاری از زبان‌های دیگر همراه باشد.

	Validation_loss	Validation_accuracy
Epoch 1	0.0889	0.9685
Epoch 2	0.0929	0.9693
Epoch 3	0.0854	0.9685
Epoch 4	0.0843	0.9715
Epoch 5	0.0873	0.9719
Epoch 6	0.0899	0.9744
Epoch 7	0.0914	0.9719
Epoch 8	0.0963	0.9726
Epoch 9	0.1009	0.9722
Epoch 10	0.1002	0.9722

کاهش دقت در Epoch ششم به بعد و افزایش مقدار Loss بعد از Epoch چهارم در جدول فوق گواه این است که مدل Fine-Tune شده Parsbert نیز بعد از Epoch ششم دچار Overfitting شده است. در این مدل نیز همانند مدل قبل امکان توقف زودهنگام(Early stopping) پس از زمانی که شواهد Overfitting مشاهده شد به صورت دستی یا با استفاده از تابع Early

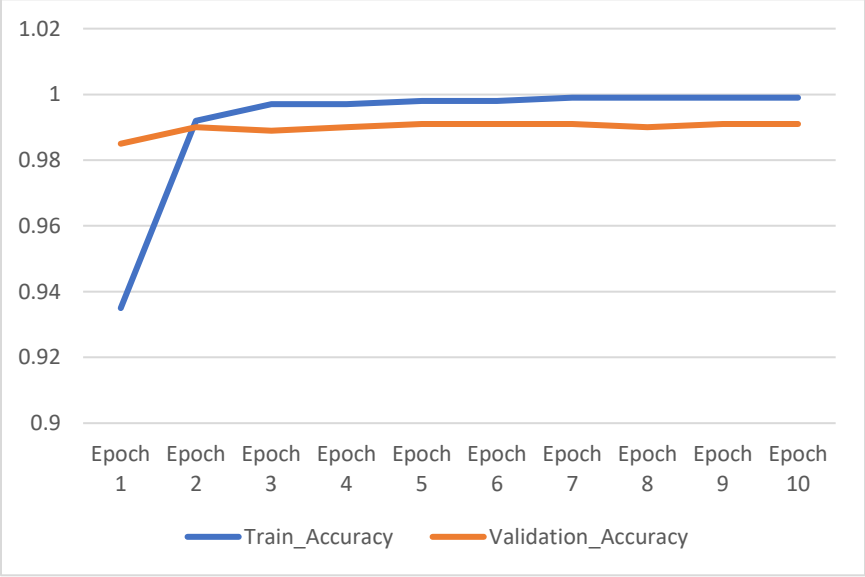
stopping از پکیج Keras وجود داشت. اگرچه این کار با توجه به خواسته سوال انجام نشد و اجازه دادیم که فرآیند مورد نظر تا انتهای Epoch دهم پیش برود.

بخش ۳

در این بخش از سوال ۲ نیز معماری کاملا مشابهی با بخش‌های قبل یعنی یک لایه ورودی با اندازه ۱۲۸، لایه میانی با اندازه ۳۲ و لایه خروجی با اندازه ۳ وجود دارد. به عنوان ورودی این مدل مقادیر ستون source را با یک توکن <SEP> به مقادیر ستون targets الصاق می‌کنیم و به عنوان ورودی به شبکه طراحی شده می‌دهیم. مقدار AUC برای این مدل برابر ۰.۹۹۱ می‌باشد و سایر نتایج ارزیابی محاسبه شده به کمک تابع classification_report به شرح زیر می‌باشد:

	Precision	Recall	F1-score	Support
Bible	0.99	1	1	900
Quran	0.99	0.99	0.99	900
Mizan	0.99	0.99	0.99	900
Accuracy			0.99	2700
Macro Avg	0.99	0.99	0.99	2700
Weighted Avg	0.99	0.99	0.99	2700

پیش از پردازش و تست کردن مدل پیش بینی می‌شد که با الصاق جملات فارسی و انگلیسی و طبقه بندی چند زبانه کیفیت مدل بالا برود و مقدار معیارهای ارزیابی از جمله Accuracy افزایش و در نهایت همین اتفاق رخ داد. طولانی‌تر شدن جملات مرتبط با یک برچسب خاص حتی اگر چند زبانه باشد با استفاده از مدل از پیش آموزش دیده مناسب می‌تواند باعث افزایش دقت مدل در پیش بینی برچسب‌ها شود و مدل فوق این فرضیه را اثبات می‌کند.



نمودار 4

با بررسی نمودار فوق متوجه می‌شویم که مقدار دقت این مدل بر روی داده‌های اعتبارسنجی از Epoch دوم تا دهم تنها ۰.۰۰۰۴ تغییر کرده است بنابراین می‌توان گفت اگرچه مدل دوزبانه زمان بیشتری را صرف پردازش هر نمونه(حدود 50 ms/step بیشتر از مدل‌های یک زبانه) و در نتیجه هر Epoch می‌کند، اما داده‌های ترکیبی انگلیسی و فارسی پس از Epoch دوم به دقت مناسبی

رسیده‌اند(۱ درصد بیش از مدل یک زبانه انگلیسی و ۲ درصد بیشتر از مدل فارسی) و زمان زیادی را با Early stopping نسبت به دیگر مدل‌ها می‌توان صرفه جویی کرد در نتیجه زمان Fine-Tune شدن مدل دو زبانه بسیار کمتر از مدل‌های تک زبانه خواهد بود و همچنین دقت بهتری را با همین رمان کمتر ارائه می‌دهد.

مقایسه نهایی مدل‌های دوزبانه و تک زبانه:

در جدول زیر مواردی از این سه مدل با یکدیگر مقایسه شده‌اند:

	Bilingual	English Monolingual	Persian Monolingual
Accuracy	0.99	0.98	0.97
Epochs	2	5	6
Each step time	974	928	919

با توجه به جدول فوق مشاهده می‌شود که مدل دو زبانه از هر دو جنبه زمان اجرا و دقت بر دو مدل بر دو مدل دیگر برتری دارد که با توجه به طولانی تر شدن جملات هر سطر در این مدل به کمک داده‌های مرتبط، این افزایش دقت منطقی به نظر می‌رسد. همچنین با توجه به پیچیدگی نحوی و ساخت‌واژی زبان فارسی نسبت به زبان انگلیسی و همچنین کمبود مدل‌های از پیش آموزش دیده قدرتمند برای زبان فارسی، ضعف اندک مدل تک زبانه فارسی نسبت به مدل انگلیسی قابل درک است.

سوال ۳- Cross-lingual zero-shot transfer learning

بخش ۱:

انتظار می‌رود که عملکرد این مدل نسبت به مدل‌های دیگر ضعف چشم‌گیری داشته باشد چرا که مدل‌های از پیش آموزش دیده پایه مانند Bert، Roberta، DistilBert و ... به صورت عمومی آموزش دیده‌اند و آنچه این مدل‌ها را خاص می‌کند و باعث عملکرد خوب آن‌ها در کاربردهای خاص می‌شود انجام فرآیند Fine-Tunning است. بنابراین نمی‌توان انتظار داشت که یک مدل را برای کاربردی خاص Fine-Tune کرد و این مدل برای کاربردی دیگر(یا زبانی دیگر از همان کاربرد) به خوبی عمل کند و مقدار بالایی برای معیارهای ارزیابی ارائه دهد چون در این صورت دیگر نیازی به عمل Fine-Tunning وجود نداشت و یک مدل را می‌توانستیم برای تمام کاربردها به صورت مستقیم استفاده کنیم.

بخش ۲:

در این بخش مدل از پیش آموزش دیده xlm-Roberta و معماری استفاده شده در سوال ۱ و ۲ را به کمک داده‌های ستون Source بخش آموزش و اعتبارسنجی دیتاست Fine-Tune می‌کنیم و سپس با استفاده از داده‌های ستون Targets بخش تست دیتاست مدل را ارزیابی می‌کنیم. نتایج ارزیابی مدل به شرح زیر خواهد بود:

	Precision	Recall	F1-score	Support
Bible	0.92	0.53	0.67	900
Quran	0.73	0.87	0.79	900
Mizan	0.79	0.98	0.88	900
Accuracy			0.79	2700
Macro Avg	0.81	0.79	0.78	2700
Weighted Avg	0.81	0.79	0.78	2700

همانطور که در جدول فوق مشاهده می‌شود بر خلاف انتظار مدل Fine-Tune شده با زبان انگلیسی و تست شده با زبان فارسی دقت نسبتا خوب ۷۹ درصد را ارائه می‌دهد. به کمک وجود ترجمه و معادل کلمات بین زبان‌های مختلف در مدل‌های چند زبانه مانند mBERT ما توان استفاده از این قابلیت Cross-lingual zero-shot transfer learning را به دست می‌آوریم و می‌توانیم در مواقعی که کمبود داده برچسب خورده داریم از این تکنیک استفاده کنیم.

بخش ۳

هدف Cross-lingual zero-shot transfer learning این است که مدل هایی را با داده های موجود در یک یا چند زبان مبدا آموزش دهد و از آن‌ها در زبان‌های مقصد دیگر که داده‌ای برای آنها وجود ندارد یا دیتاست مناسب برای آن در دسترس نیست استفاده شوند. ارزش این مدل‌ها برای زبان‌هایی که منابع کمی دارند مشخص می‌شود چرا که به داده‌های برچسب خورده کمتری برای آموزش مدل احتیاج دارد. این روش در کاربردهایی مانند Question Asnwering و طبقه بندی کاربرد دارد. اخیرا مدل‌های از پیش آموزش دیده چند زبانه متعددی مانند mBERT و XLM-Roberta در دسترس قرار گرفته‌اند که برای کاربردهای فوق می‌توانند مورد استفاده قرار گیرند.