Assignment 4

NLP

PREPARED BY

Mohsen Fayyaz - 810100524

Spring 2022

فهرست مطالب

ParsiNLU Dataset Classification 1 **3**

1) در ابتدا تحلیلی بر روی داده های train داشته باشید. آیا پیش پردازش خاصی لازم دارد؟(در صورتی که جواب شما منفی است دلیلتان را توضیح دهید و در غیر اینصورت موارد مورد نظر را پیاده سازی کرده و دلایلتان را شرح دهید). 3

2) شبکه عصبی عمیقی طراحی کنید که به کمک آن بتوانید داده ها را طبقه بندی کنید. به منظور اعمال Word-embedding از مدل Multilingual BERT به نام XLM-RoBERTa استفاده نمایید. این مدل در Hugging Face به آدرس زیر منتشر شده است: https://huggingface.co/xlm-roberta-base 5

3) بار دیگر شبکه طراحی شده در قسمت قبل را با ParsBET پیاده سازی کنید و آنها را با هم مقایسه کنید. 10

Multilingual classification 2 **12**

1) شبکه عصبی عمیقی طراحی کنید که به کمک آن بتوانید داده های انگلیسی را طبقه بندی کنید (می توانید از یکی از مدل های BERT، DistilRoBERTa، RoBERTa، DistilBERT و یا ... استفاده کنید). 12

2) شبکه عصبی عمیقی طراحی کنید که به کمک آن بتوانید داده های فارسی را طبقه بندی کنید (از مدلParsBERT استفاده کنید).15

3) شبکه عصبی عمیقی طراحی کنید که به کمک آن بتوانید داده های انگلیسی و فارسی را به همراه هم (راهنمایی: داده های انگلیسی و ترجمه شده فارسی آن را با یک تگ<SEP>به هم بچسبانید) طبقه بندی کنید (می توانید از یکی از مدل های چند زبانه مثل XLM-RoBERTa و یا... استفاده کنید). 16

در نهایت تحلیل خود را بر اساس نتایج به دست آمده بر روی داده های test گزارش دهید. 17

آیا مورد سوم (استفاده از مدل های چند زبانی بر روی داده های چند زبانی) باعث بالا بردن دقت شبکه خواهد شد؟ 17

Cross-lingual zero-shot transfer learning (3 امتیازی **18**

1) انتظار شما از Performance مدل بر روی داده های test زبان فارسی، قبل از اجرای این مدل چیست؟ 18

2) بعد از اجرای این مدل آیا انتظارات پیشین شما برآورده شده است؟دلیل این Performance ای که گرفته اید چیست؟ 19

3) در چه مواقعی از Cross-lingual zero-shot transfer learning استفاده می کنیم در واقع کاربرد آن را توضیح دهید. 21

ParsiNLU Dataset Classification 1

1) در ابتدا تحلیلی بر روی داده های train داشته باشید. آیا پیش پردازش خاصی لازم دارد؟(در صورتی که جواب شما منفی است دلیلتان را توضیح دهید و در غیر اینصورت موارد مورد نظر را پیاده سازی کرده و دلایلتان را شرح دهید).

ابتدا داده ها را بارگذاری می‌کنیم و چند مورد از آن را نمایش می‌دهیم.

```
pd.set_option('display.max_colwidth', -1)
pd.DataFrame(dataset["train"][0:10])
```

```
C:\Users\Mohsen\Anaconda3\lib\site-packages\ipykernel_launcher.py:1: FutureWarning: Passing a negative integer is deprecat
ed in version 1.0 and will not be supported in future version. Instead, use None to not limit the column width.
  """Entry point for launching an IPython kernel.
```

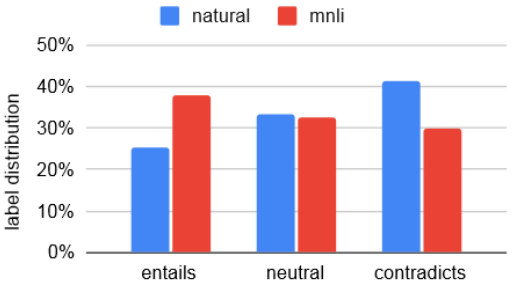
	sent1	sent2	category	label
0	زدان به قدری بخش بزرگی از نیروی کار را تشکیل می دهند که به سختی می توان باور داشت که اگر این امر در مورد زدان صادق .نباشد ، این امر می تواند صادق باشد	.مردان بخش عظیمی از نیروی کار هستند بنابراین تنها افراد مهم هستند	translation-train	c
1	.سالها است که کندگره در تلاش است تا اثربخشی مدیریت اطلاعات و فناوری را در دولت فدرال بالا ببرد	کندگره یوندجه ویژه ای برای مدیریت اطلاعات و فناوری در دولت فدرال .دارد	translation-train	n
2	.سرامیک‌های زیست خنثی پس از قرارگیری در بدن میزبان خواص فیزیکی و مکانیکی خود را حفظ می‌کند	.خواص فیزیکی سرامیک‌ها قابل اندازه گیری است	natural-wiki	n
3	دولت از هیچ قانونی که منجر به کاهش چشمگیر توانایی کشورمان در استفاده از زغال سنگ به عنوان منبع اصلی برق فعلی و آینده شود ، .پشتیبانی نخواهد کرد	قانونی که باعث کاهش استفاده از زغال سنگ به عنوان منبع انرژی شود ، .توسط دولت پشتیبانی نمی شود	translation-train	e
4	(approximate algorithms) و الگوریتم‌های تقریبی (exact) روش‌ها و الگوریتم‌های بهینه‌سازی به دو دسته الگوریتم‌های دقیق .تقسیم‌بندی می‌شوند	.آمار در دروس مدیریتی نقش مهمی را بازی میکند	natural-wiki	n
5	گرچه بسیاری از اختلافات یاد شده در بین فقها برسر بحث سفر صلاتی بوده است، اما اختلافات آنها در بحث کمّیت پوشش برای تمام به .بحث کمّیت پوشش از دگاه نیز سرایت کرده است	در داشتن حجاب به هنگام نماز در یک اتاق خالی و بدون وجود نامحرم چه فلسفه ای نهفته است؟	natural-wiki	n
6	.واقع شده‌است Koçarlı یک منطقهٔ مسکونی در ترکیه است که در (Tığıllar, Koçarlı :به لاتین) تیقیلر، کچارلی	.کوچه لوه سو سیمیشم یک اهدگ ترکی قدیمی می باشد	natural-wiki	n
7	وزیر کشور عراق در ادامه با بیان اینکه ایران رنج‌های زیادی بابت تحریم‌های اقتصادی مواجه شد اظهار کرد: اما ایران رهبر فرزانه و .حکیم دارد که توانست از همه مشکلات عبور کند	به گفته وزیر کشور عراق ایران توانست مشکلات تحریم را پشت سر بگذارد	natural-miras	e
8	.لیدگان چشم دوخت	.مرد چشمک زد	translation-dev	c
9	.سیف" کتب و مقالات بسیاری در زمینهٔ اقتصاد، سیاست و تاریخ مشروطه به رشتهٔ تحریر درآورده یاترجمه کرده است"	.فعالیتهای آقای "سیف" بیشتر معطوف بر نوشتار و ترجمه بود	natural-wiki	e

شکل 1 - بخشی از دیتای آموزش

همانطور که دیده می‌شود دیتاست تمیز است و همچنین چون از مدل‌های جدید مانند xlm-roberta قرار است استفاده کنیم که از BPE برای توکنایز کردن استفاده می‌کنند و همچنین contextual هستند و همچنین توزیع داده پیش‌آموزششان مشابه بوده مانند ویکیپدیا مشکلی از نظر پیش پردازش نداریم. مثلا اگر می‌خواستیم مدلی ساده مثل naive bayes داشته باشیم اهمیت پیدا می‌کرد که دقیقا چطور داده‌ها شکسته شوند و توکنایز شوند و شمرده شوند. ولی در مدل BERT مشکلی از این جهات نیست و در هر صورت که داده باشد امبدینگ مناسب آن ساخته شده است و تاثیر شدیدی ندارد ضمنا که داده ها هم خودشان تمیز هستند.

برای اطمینان بیشتر، مقاله این دیتاست در https://arxiv.org/pdf/2012.06154.pdf را نیز بررسی می‌کنیم. در بخش 3.2.4 که این دیتاست توضیح داده می‌شود گفته شده که داده‌ها دو حالت داشته‌اند.

یا به صورت طبیعی از Miras,10 Persian Wikipedia and VOA corpus برداشته شده‌اند و به annotator ها داده شده‌اند که طبیعتا دیتاست های بسیار تمیزی هستند (همچنین wikipedia احتمال زیاد با داده پیش‌آموزش مدلهایی که استفاده می‌کنیم هم اشتراک دارد). یا حالت دوم این بوده که از دیتاست معروف MNLI در انگلیسی استفاده و ترجمه شده است که طبق مقاله ترجمه‌ها بررسی و صحیح شده‌اند و به همین دلیل مشکل خاصی از نظر تمیزی دیتاست نداریم.
در آخر توزیع داده‌ها هم در مقاله آمده که در زیر می‌آوریم.



شکل 2 - توزیع لیبل داده‌ها

ضمنا باید اشاره کرد که این دو دسته طبیعی و ترجمه از MNLI در ستون category دیتاست در اختیار ما گذاشته شده است. تنها پیش پردازشی که لازم بود انجام شود روی لیبل ها بود که بعضی مشکل داشتند و با استفاده از تابع فیلتر دیتاست این کار انجام شد.

```
for data_set in ["train", "validation", "test"]:
#     print(np.unique(dataset[data_set]["label"], return_counts=True))
    bad_idxs = [i for i in range(len(dataset[data_set])) if dataset[data_set][i]["label"] not in ["c", "e", "n"]]
    print(dataset[data_set][bad_idxs])
    print()

dataset = dataset.filter(lambda example: example['label'] in ["c", "e", "n"])
dataset
```

```
Loading cached processed dataset at C:\Users\Mohsen\.cache\huggingface\datasets\persiannlp___parsinlu_reading_comprehensio
n\parsinlu-repo\1.0.0\a99ad431d1637751f03fa9da3de66a41e93983d11116a0918a6efaaa72343d2d\cache-ddbade114b5e3de9.arrow
Loading cached processed dataset at C:\Users\Mohsen\.cache\huggingface\datasets\persiannlp___parsinlu_reading_comprehensio
n\parsinlu-repo\1.0.0\a99ad431d1637751f03fa9da3de66a41e93983d11116a0918a6efaaa72343d2d\cache-def594aad0f0a1fa.arrow
Loading cached processed dataset at C:\Users\Mohsen\.cache\huggingface\datasets\persiannlp___parsinlu_reading_comprehensio
n\parsinlu-repo\1.0.0\a99ad431d1637751f03fa9da3de66a41e93983d11116a0918a6efaaa72343d2d\cache-e57186a74fccface.arrow
```

```
{'sent1': ['xx'], 'sent2': ['xx'], 'category': ['natural-miras'], 'label': ['xx']}

{'sent1': [], 'sent2': [], 'category': [], 'label': []}

{'sent1': ['آنها برنامه هایی را برای افزایش کارآیی و به کارگیری منابع به طور مؤثر تثبیت کردند.', 'این هل نوع جدیدی از هل های فوق مشاره و کوچک را به نمایش می گذار.'], 'sent2': ['برنامه های افزایش بهره وری تحقق شدند ، زیرا آنها عمیقاً به کارایی اهمیت می دهد.', 'که فضای اداری عمومی را به افرادی که از راه دور کار میکنند، می فروشد'], 'category': ['translation-dev', 'translation-dev'], 'label': ['-', '-']}


DatasetDict({
    train: Dataset({
        features: ['sent1', 'sent2', 'category', 'label'],
        num_rows: 754
    })
    test: Dataset({
        features: ['sent1', 'sent2', 'category', 'label'],
        num_rows: 1673
    })
    validation: Dataset({
        features: ['sent1', 'sent2', 'category', 'label'],
        num_rows: 270
    })
})
```

شکل 3 - فیلتر لیبلهای نامناسب

2) شبکه عصبی عمیقی طراحی کنید که به کمک آن بتوانید داده ها را طبقه بندی کنید. به منظور اعمال Word-embedding از مدل Multilingual BERT به نام XLM-RoBERTa استفاده نمایید. این مدل در Hugging Face به آدرس زیر منتشر شده است: https://huggingface.co/xlm-roberta-base

ابتدا باید اشاره کرد که مدل XLM-RoBERTa همانطور که از اسمش پیداست بر اساس روبرتا ساخته شده است که pre-training objective متفاوت، هایپرپارامترهای متفاوت و ضمنا دیتای بیشتر برای پیش‌آموزش نسبت به برت داشته است.

همانطور که در مقاله آمده است، برای evaluation تسک textual entailment از accuracy استفاده می‌شود. برای این کار این متریک بارگذاری می‌کنیم و تابع محاسبه آن را برای معرفی به ترینر آماده می‌کنیم. همچنین دیتاست را با استفاده از توکنایزر توکنایز می‌کنیم که ستون جدید input_ids که همان عدد نسبت داده شده به هر توکن در آن است و ورودی مدل است اضافه می‌شود. در این تسک که دو جمله داریم ترتیب به این شکل می‌شود که ابتدا توکن CLS که به شکلی همان بازنمایی کل جمله می‌توان دانست می‌آید، سپس جمله اول و سپس توکنایزر خودش عدد معادل توکن SEP که جدا کننده است را می‌گذارد و سپس جمله دوم که این دو را از هم جدا کند. ضمنا attention mask هم ساخته می‌شود که نشان می‌دهد در کجا جمله و در کجا پدینگ قرار گرفته است. البته در اینجا کل وکتور برابر 1 است چون پدینگ را اینجا اعمال نمیکنیم. خود ترینر هاگینگ فیس در ادامه به صورتی که بچ ها از جملات نسبتا هم اندازه تشکیل شده باشد (group_by_length = True) پدینگ ها را بچ بچ اعمال می‌کند تا سرعت آموزش بالا رود. (چون سرعت اجرا به ماکسیمم اندازه جمله در هر بچ بستگی دارد و بخاطر یک جمله به طول 100 ممکن است تمام بقیه جملات داخل بچ هم به 100 پد شوند و اجرای بچ را خیلی کندتر می‌کند.) همچنین در این مرحله چون لیبل ها به صورت استرینگ بودند، آن ها را نیز تبدیل به عدد می‌کنیم.

```python
metric = load_metric("accuracy")
def compute_metrics(eval_pred):
    logits, labels = eval_pred
    predictions = np.argmax(logits, axis=-1)
    return metric.compute(predictions=predictions, references=labels)

str_to_int = {"e": 0, "n": 1, "c": 2}
def tokenize_function(examples):
    tokenized_batch = tokenizer(examples["sent1"], examples["sent2"], truncation=True)
    tokenized_batch["label"] = [str_to_int[label] for label in examples["label"]]
    return tokenized_batch

tokenized_datasets = dataset.map(tokenize_function, batched=True)
tokenized_datasets
```

شکل 4 - متریک و توکنایزر

در ادامه پس از تستهای متعدد هایپرپارامترهای مناسب را پیدا کرده و مدل را با آن ها آموزش دادیم که در زیر نتیجه آن آمده است.

```python
training_args = TrainingArguments(
    output_dir="xlm-r",
    evaluation_strategy="epoch",
    save_strategy = "epoch",
    logging_steps = 20,
    learning_rate=1e-5,
    num_train_epochs=10,
    per_device_train_batch_size=8,
    per_device_eval_batch_size=8,
    load_best_model_at_end=True,
    save_total_limit = 1,
    metric_for_best_model="accuracy"
)
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=tokenized_datasets["train"],
    eval_dataset=tokenized_datasets["validation"],
    tokenizer=tokenizer,
    compute_metrics=compute_metrics,
)
trainer.train()
```

[950/950 08:22, Epoch 10/10]

Epoch	Training Loss	Validation Loss	Accuracy
1	1.108500	1.145660	0.288889
2	1.090500	1.124437	0.288889
3	1.064100	1.127263	0.344444
4	1.006900	1.096134	0.403704
5	0.898900	1.075101	0.440741
6	0.861600	1.060790	0.474074
7	0.647300	1.103528	0.470370
8	0.576400	1.081326	0.496296
9	0.568000	1.104497	0.514815
10	0.478500	1.109849	0.511111

```python
trainer.evaluate(tokenized_datasets["test"])
```

```
The following columns in the evaluation set  don't have
orward` and have been ignored: category, sent2, sent1.
eClassification.forward`,  you can safely ignore this m
***** Running Evaluation *****
  Num examples = 1673
  Batch size = 8
```

[210/210 00:09]

```
{'eval_loss': 1.1014331579208374,
 'eval_accuracy': 0.5331739390316796,
 'eval_runtime': 9.212,
 'eval_samples_per_second': 181.612,
 'eval_steps_per_second': 22.796,
 'epoch': 10.0}
```

شکل 5 - آموزش مدل xlm-r و نتایج آن

همانطور که دیده می‌شود لاس ترین کم می‌شود و لاس ولیدیشن هم همچنین کم می‌شود تا یکجا که دیگر افزایشی می‌شود و چون در ترینر مشخص کردیم که بهترین مدل در آخر بارگذاری شود مشکلی نیست و خودش این کار را انجام می‌دهد. در انتها نیز با استفاده از evaluate روی داده تست دقت را می‌سنجیم که برابر 53.31 درصد است. (البته در بهترین حالت. در خیلی از مواقع بین 47 تا 53 متغیر است و به هایپرپارامترها بسیار بستگی دارد.) قابل ذکر است که در مقاله برای مدل پارس برت آمده که برای بخش طبیعی دقت 51.8 و برای بخش ترجمه شده دقت 53.9 ثبت شده است. بنابراین این دقتی که به دست آوردیم مناسب است و مدل به خوبی آموزش دیده است.

در مورد مدل بهتر است بیشتر توضیح دهیم.

برای بارگذاری مدل از AutoModelForSequenceClassification استفاده می‌کنیم که ورودی آن مدل و

تعداد لیبل خروجی است و لایه های مورد نیاز را به بالای بازنمایی CLS در آخرین اضافه می‌کند و به تعدادی که داده ایم نورون خروجی می‌گذارد که لوجیت‌ها را خروجی دهند. جدای از راحت‌تر بودن استفاده از این کلاس برای این کار، مدلهای مختلف، لایه های متفاوتی را برای classification انتهای مدل پیشنهاد می‌دهند. مثلا یک مدل یکی از لایه های denseش داخل خود مدل پیاده شده است و باید فقط یک لایه dense دیگر بالای آن قرار گیرد و بعضی مدل‌ها هر دو لایه باید خارج باشد یا بعضی مدل‌ها از مقادیر خاصی برای dropout یا activation function های متفاوت مثل RELU یا LeakyRELU استفاده می‌کنند که پیشنهادی مقاله آن مدل بوده است. استفاده از این تابع دقیقا همان حالتی که پیشنهاد شده را خودش اضافه می‌کند.

```
model_checkpoint = "xlm-roberta-base"
tokenizer = AutoTokenizer.from_pretrained(model_checkpoint)
model = AutoModelForSequenceClassification.from_pretrained(model_checkpoint, num_labels=3)
model
```

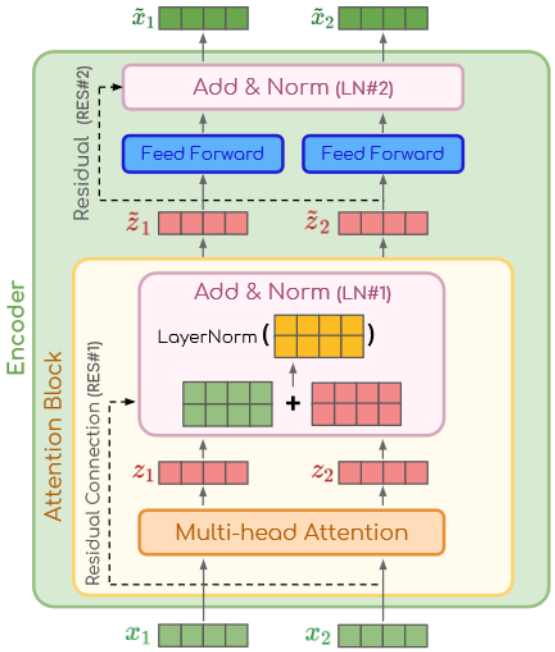
```
Some weights of the model checkpoint at xlm-roberta-base were not used when initializing XLMR
on: ['lm_head.dense.weight', 'roberta.pooler.dense.weight', 'lm_head.dense.bias', 'lm_head.la
coder.weight', 'lm_head.bias', 'roberta.pooler.dense.bias', 'lm_head.layer_norm.bias']
- This IS expected if you are initializing XLMRobertaForSequenceClassification from the check
nother task or with another architecture (e.g. initializing a BertForSequenceClassification m
g model).
- This IS NOT expected if you are initializing XLMRobertaForSequenceClassification from the c
expect to be exactly identical (initializing a BertForSequenceClassification model from a Ber
del).
Some weights of XLMRobertaForSequenceClassification were not initialized from the model check
are newly initialized: ['classifier.out_proj.bias', 'classifier.out_proj.weight', 'classifier
se.weight']
You should probably TRAIN this model on a down-stream task to be able to use it for predictio
```

```
XLMRobertaForSequenceClassification(
  (roberta): RobertaModel(
    (embeddings): RobertaEmbeddings(
      (word_embeddings): Embedding(250002, 768, padding_idx=1)
      (position_embeddings): Embedding(514, 768, padding_idx=1)
      (token_type_embeddings): Embedding(1, 768)
      (LayerNorm): LayerNorm((768,), eps=1e-05, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
    (encoder): RobertaEncoder(
      (layer): ModuleList(
        (0): RobertaLayer(
          (attention): RobertaAttention(
            (self): RobertaSelfAttention(
              (query): Linear(in_features=768, out_features=768, bias=True)
              (key): Linear(in_features=768, out_features=768, bias=True)
              (value): Linear(in_features=768, out_features=768, bias=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
            (output): RobertaSelfOutput(
              (dense): Linear(in_features=768, out_features=768, bias=True)
              (LayerNorm): LayerNorm((768,), eps=1e-05, elementwise_affine=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
          )
```

شکل 6 - قسمت ابتدایی مدل

در تصویر بالا نحوه ساخت مدل و لایه های ابتدایی آن دیده می‌شود. همانطور که می‌دانیم لایه 0 همان لایه امبدینگ است که از وان هات ووکب به بازنمایی 768 (در مدل های بیس) بعدی می‌برد. در اینجا چون مدل xlm-r است و چند زبانه است تعداد ووکب همانطور که دیده می‌شود خیلی بزرگ و حدود 250 هزار است در حالیکه مدل های یک زبانه خیلی کوچکترند. همین موضوع باعث افزایش پارامترهای این مدل و سنگین تر شدنش نسبت به مدلهای تک زبانه هم می‌شود.

در ادامه اولین لایه از 12 لایه encoder را می‌بینیم که از بخشهای self attention و Add&Norm و فیدفوروارد و یک Add&Norm دیگر تشکیل شده است. در بخش اول با استفاده از روش attention که در مقاله transformer توضیح داده شده است به شکل استفاده از q, k, v توجه هر توکن به توکنهای دیگر را به دست می‌آوریم. در بخشهای بعدی دو residual connection داریم که ورودی را با خروجی جمع می‌کنند که نشان داده شده است به مدل کمک می‌کند و همچنین از مشکل vanishing gradients جلوگیری می‌کند. همچنین مدل dense هم داریم که اگر ترکیب خاصی لازم بود بتواند یاد بگیرد و اجرا کند که تفسیرپذیری سخت تری دارد.



شکل 7 - بخش encoder ترنسفورمرها (برگرفته از https://arxiv.org/pdf/2205.03286.pdf)

در شکل بالا بخشهایی که توضیح داده شد از شکل مقاله https://arxiv.org/pdf/2205.03286.pdf آمده است.

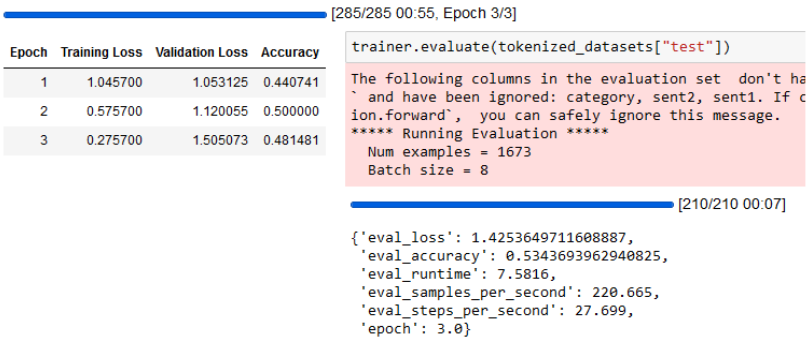
```
      (11): RobertaLayer(
        (attention): RobertaAttention(
          (self): RobertaSelfAttention(
            (query): Linear(in_features=768, out_features=768, bias=True)
            (key): Linear(in_features=768, out_features=768, bias=True)
            (value): Linear(in_features=768, out_features=768, bias=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
          (output): RobertaSelfOutput(
            (dense): Linear(in_features=768, out_features=768, bias=True)
            (LayerNorm): LayerNorm((768,), eps=1e-05, elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
        (intermediate): RobertaIntermediate(
          (dense): Linear(in_features=768, out_features=3072, bias=True)
          (intermediate_act_fn): GELUActivation()
        )
        (output): RobertaOutput(
          (dense): Linear(in_features=3072, out_features=768, bias=True)
          (LayerNorm): LayerNorm((768,), eps=1e-05, elementwise_affine=True)
          (dropout): Dropout(p=0.1, inplace=False)
        )
      )
    )
  )
  (classifier): RobertaClassificationHead(
    (dense): Linear(in_features=768, out_features=768, bias=True)
    (dropout): Dropout(p=0.1, inplace=False)
    (out_proj): Linear(in_features=768, out_features=3, bias=True)
  )
)
```

شکل 8 - بخش انتهایی مدل

در بخش انتهایی مدل همانطور که گفته شد بعد از 12 لایه انکودر (0 تا 11) یک classification head قرار داده شده که خاص این مدل است. مثلا در این مدل می‌بینیم که داخل خود مدل لایه ای به نام RobertaOutput هست که یک لایه linear دارد و به همین دلیل در هدی که اضافه گذاشته شده فقط یک لایه دیگر هست در حالی که در مدلهای دیگر این مورد می‌تواند متفاوت باشد. همچنین باید دقت کرد 768 بعد یک بازنمایی و به طور خاص بازنمایی CLS در لایه آخر است و 3 در خروجی همان تعداد لیبل است که خودمان در کد نوشتیم تا به آن تعداد کلاس خروجی داشته باشیم.

3) بار دیگر شبکه طراحی شده در قسمت قبل را با ParsBET پیاده سازی کنید و آنها را با هم مقایسه کنید.

در این قسمت تنها کافیست همان قسمت قبل را صرفا با استفاده از مدل HooshvareLab/bert-base-parsbert-uncased اجرا کنیم.



شکل 6 - آموزش مدل parsbert و نتایج آن

همانطور که دیده می‌شود دقت مدل روی داده تست برابر 53.43 درصد است (در این مورد هم بسته به هایپرپارامترها بین 47 تا 53 می‌توان مقادیر متفاوتی دید) که همانطور که گفته شد نسبت به دقتهای گزارش شده در پیپر درست است. برای مقایسه دقت مدل‌ها در کارهای پژوهشی هوش مصنوعی معمولا باید اجراها با سیدهای مختلف مثلا 5 سید اجرا شوند و در آخر میانگین و انحراف از معیار آن گزارش شود و ضمنا در صورت لزوم confidence interval هم محاسبه شود تا بتوان از نظر آماری به یقین رسید که دو مدل در نتیجه به مقداری که significant باشد تفاوت دارند. در این تمرین صرفا دو عدد داریم و شرایط هایپر پارامترها نیز خاص است و نمیتوان بر اساس اختلاف کمی که دیده شد لزوما حکم داد که کدام مدل بهتر از دیگری است.

اما با این وجود، همین که دو مدل توانستند به خوبی این دیتاست را بیاموزند نشانه پیش آموزش مناسب آنها و آشنایی آنها با فارسی است. در این مورد خاص پارس برت کمی بهتر عمل کرد که می‌توان آن را به علت این دانست که تمام وزنهای این مدل برای فارسی آموزش دیده اند و تمرکز آن روی یک زبان بوده، ولی xlm-r تعداد زبان بسیاری را پشتیبانی می‌کند و به همین دلیل در مدل خود باید گنجایش تمام 88 زبانی که آموزش دیده را داشته باشد و به همین دلیل شاید نتواند به خوبی یک مدل که تمرکزش یک زبان بوده عمل کند. این مورد در خود انگلیسی هم دیده شده به این صورت که اگر مقاله آن https://arxiv.org/pdf/1911.02116.pdf مرور شود این نکته در مقایسه این مدل با روبرتا دیده می‌شود که روبرتا در انگلیسی بهتر عمل کرده. (ضمنا جالب است که زبان فارسی پنجمین زبان پر منبع بین

88 زبانی که این مدل روی آنها آموزش دیده شده بوده است.) اما با این وجود اختلاف اندک است و شاید بتوان یکی از خوبی های این مدل را آن دانست که روی انگلیسی هم به خوبی آموزش دیده و به صورت غیر مستقیم شاید بتواند روی بخشی از دیتاست که از mnli ترجمه شده بهتر عمل کند. ضمنا تفاوت دیگر پارس برت و xlm-r در اندازه ووکب آنها است که xlm-r چون 88 زبان را پشتیبانی می‌کند، ووکب خیلی بزرگتری نسبت به پارس برت دارد و باز از پیچیدگی هایی است که مدل باید در وزنهای خود بگنجاند.

ضمنا باید اشاره کرد بخاطر ووکب خیلی بزرگ xlm-r که در بخشهای قبل هم توضیح داده شد، مدل بزرگ تر و کندتر از پارس برت است و این خودش را در آموزش هم نشان می‌دهد و در کل پارس برت برای جایی که سرعت مهم است می‌تواند گزینه بهتری باشد.

Multilingual classification 2

1) شبکه عصبی عمیقی طراحی کنید که به کمک آن بتوانید داده های انگلیسی را طبقه بندی کنید (می توانید از یکی از مدل های DistilRoBERTa، RoBERTa، DistilBERT، BERT و یا ... استفاده کنید).

ابتدا داده‌ها را بارگذاری می‌کنیم.

```
In [3]: df = pd.read_excel("Question2_Data/train.xlsx")
        df
```

Out[3]:

	source	targets	category
0	When news is brought to one of them, of (the b...	...و چون یکی از آنان را به [ولادت] دختر مژده دهند	quran
1	After them repaired Zadok the son of Immer ove...	...و چون دشمنان ما شنیدند که ما آگاه شده‌ایم و خد	bible
2	And establish regular prayers at the two ends ...	...و نماز را در دو طرف روز و ساعات نخستین شب برپا	quran
3	And it came to pass, that, when I was come aga...	...و فرمود تا مدعیانش نزد تو حاضر شوند؛ و از او ب	bible
4	Ah woe, that Day, to the Rejecters of Truth!	!وای در آن روز بر تکذیب کنندگان	quran
...	...	...	...
12595	Women impure are for men impure, and men impur...	...زنان پلید برای مردان پلید و مردان پلید برای زن	quran
12596	I don't want any silly dance given in my honour.'	...بدابراین حالا هم میل ندارم جشنی به افتخار من د	mizan
12597	And the Earth will shine with the Glory of its...	...و زمین به نور پروردگارش روشن می‌شود، و کتاب [ا	quran
12598	Then lifted I up mine eyes, and saw, and behol...	...گفتم: «این چیست؟» او جواب داد: «این است آن اِیف	bible
12599	His soul was dried up.	.روح خشکیده بود	mizan

12600 rows × 3 columns

```
In [4]: dataset = DatasetDict()
        for data_set in ["train", "valid", "test"]:
            dataset[data_set] = Dataset.from_pandas(pd.read_excel(f"Question2_Data/{data_set}.xlsx"))
        dataset
```

Out[4]:
```
DatasetDict({
    train: Dataset({
        features: ['source', 'targets', 'category'],
        num_rows: 12600
    })
    valid: Dataset({
        features: ['source', 'targets', 'category'],
        num_rows: 2700
    })
    test: Dataset({
        features: ['source', 'targets', 'category'],
        num_rows: 2700
    })
})
```

شکل 7 - بارگذاری داده‌ها

برای مدل از electra استفاده می‌کنیم که pre-training objective شبیه به GAN دارد و نشان داده شده که می‌تواند نتایج بهتری نسبت به برت داشته باشد. برای بارگذاری و آموزش آن مشابه سوال قبل عمل می‌کنیم.

```
model_checkpoint = "google/electra-base-discriminator"
tokenizer = AutoTokenizer.from_pretrained(model_checkpoint)
model = AutoModelForSequenceClassification.from_pretrained(model_checkpoint, num_labe
model
```

```
          (intermediate): ElectraIntermediate(
            (dense): Linear(in_features=768, out_features=3072, bias=True)
            (intermediate_act_fn): GELUActivation()
          )
          (output): ElectraOutput(
            (dense): Linear(in_features=3072, out_features=768, bias=True)
            (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
      )
    )
  )
  (classifier): ElectraClassificationHead(
    (dense): Linear(in_features=768, out_features=768, bias=True)
    (dropout): Dropout(p=0.1, inplace=False)
    (out_proj): Linear(in_features=768, out_features=3, bias=True)
  )
)
```

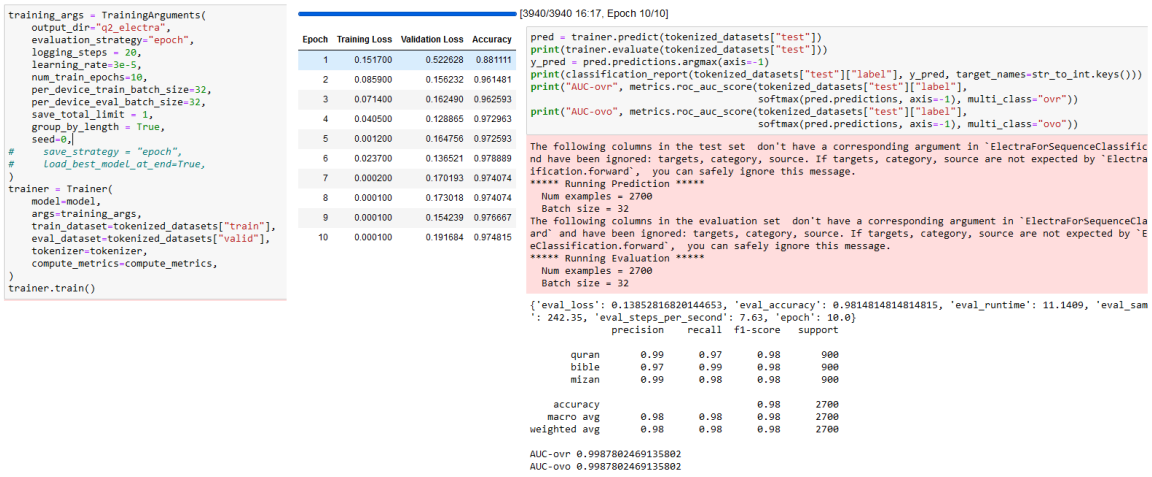
```
metric = load_metric("accuracy")
def compute_metrics(eval_pred):
    logits, labels = eval_pred
    predictions = np.argmax(logits, axis=-1)
    return metric.compute(predictions=predictions, references=labels)

str_to_int = {"quran": 0, "bible": 1, "mizan": 2}
def tokenize_function(examples):
    tokenized_batch = tokenizer(examples["source"], truncation=True, max_length=128)
    tokenized_batch["label"] = [str_to_int[label] for label in examples["category"]]
    return tokenized_batch

tokenized_datasets = dataset.map(tokenize_function, batched=True)
print(tokenized_datasets["train"][0])
tokenized_datasets
```

شکل 8 - بارگذاری مدل و توکنایز کردن داده

در این بخش مدل را بارگذاری کردیم . همچنین متریک accuracy را برای اینکه هر ایپاک روی بخش ولیدیشن محاسبه شود نوشتیم. ضمنا داده ها را توکنایز کردیم و همانطور که گفته شده بود حداکثر طول را 128 گذاشتیم و همچنین لیبل ها را از حالت متنی به عددی در آوردیم و در فیچر label که ترینر انتظار دارد لیبل در آن باشد گذاشتیم. با این کار input_ids و attention_mask که در بخشهای قبلی هم توضیح داده شده بود اضافه می‌شود. همچنین مشابه توضیحات سوال اول، بخش classificationHead اضافه شده است که ورودی 768 بعد بازنمایی CLS لایه آخر (بعد از گذر از یک لایه Dense) و خروجی 3 حالت لیبل ها است که در مسئله داریم.



```
training_args = TrainingArguments(
    output_dir="q2_electra",
    evaluation_strategy="epoch",
    logging_steps = 20,
    learning_rate=3e-5,
    num_train_epochs=10,
    per_device_train_batch_size=32,
    per_device_eval_batch_size=32,
    save_total_limit = 1,
    group_by_length = True,
    seed=0,
#     save_strategy = "epoch",
#     load_best_model_at_end=True,
)
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=tokenized_datasets["train"],
    eval_dataset=tokenized_datasets["valid"],
    tokenizer=tokenizer,
    compute_metrics=compute_metrics,
)
trainer.train()
```

[3940/3940 16:17, Epoch 10/10]

Epoch	Training Loss	Validation Loss	Accuracy
1	0.151700	0.522628	0.881111
2	0.085900	0.156232	0.961481
3	0.071400	0.162490	0.962593
4	0.040500	0.128865	0.972963
5	0.001200	0.164756	0.972593
6	0.023700	0.136521	0.978889
7	0.000200	0.170193	0.974074
8	0.000100	0.173018	0.974074
9	0.000100	0.154239	0.976667
10	0.000100	0.191684	0.974815

```
pred = trainer.predict(tokenized_datasets["test"])
print(trainer.evaluate(tokenized_datasets["test"]))
y_pred = pred.predictions.argmax(axis=-1)
print(classification_report(tokenized_datasets["test"]["label"], y_pred, target_names=str_to_int.keys()))
print("AUC-ovr", metrics.roc_auc_score(tokenized_datasets["test"]["label"],
                                       softmax(pred.predictions, axis=-1), multi_class="ovr"))
print("AUC-ovo", metrics.roc_auc_score(tokenized_datasets["test"]["label"],
                                       softmax(pred.predictions, axis=-1), multi_class="ovo"))
```

```
The following columns in the test set  don't have a corresponding argument in `ElectraForSequenceClassific
nd have been ignored: targets, category, source. If targets, category, source are not expected by `Electra
ification.forward`,  you can safely ignore this message.
***** Running Prediction *****
  Num examples = 2700
  Batch size = 32
The following columns in the evaluation set  don't have a corresponding argument in `ElectraForSequenceCla
and` and have been ignored: targets, category, source. If targets, category, source are not expected by `E
eClassification.forward`,  you can safely ignore this message.
***** Running Evaluation *****
  Num examples = 2700
  Batch size = 32
{'eval_loss': 0.13852816820144653, 'eval_accuracy': 0.9814814814814815, 'eval_runtime': 11.1409, 'eval_sam
': 242.35, 'eval_steps_per_second': 7.63, 'epoch': 10.0}
              precision    recall  f1-score   support

       quran       0.99      0.97      0.98       900
       bible       0.97      0.99      0.98       900
       mizan       0.99      0.98      0.98       900

    accuracy                           0.98      2700
   macro avg       0.98      0.98      0.98      2700
weighted avg       0.98      0.98      0.98      2700

AUC-ovr 0.9987802469135802
AUC-ovo 0.9987802469135802
```

شکل 8 - آموزش مدل انگلیسی و تست آن

همانطور که دیده می‌شود مدل را با هایپرپارامترهای گفته شده اجرا کردیم. در انتها نیز با استفاده از تابه predict لوجیتها را به دست آوردیم و بعد از گرفتن argmax به classification_reportد دادیم که نتایجش مشخص است و حدود 98 درصد است که خیلی بالاست. اما برای auc که خواسته شده، این مورد برای مسائل دو کلاسه به کار می‌رود. همچنان که حالت عادی تابع برای محاسبه آن در حالت چند کلاسه ارور دادن است! اما با استفاده از ورودی multi_class و مقدار دهی آن به One-vs-rest و One-vs-one که مسئله را به همان حالتهای دو کلاسه تبدیل می کند و سپس میانگین می گیرد این کار را انجام دادیم. ضمنا باید اشاره کرد که ورودی برای محاسبه auc معادل احتمال باید باشد و به همین لوجیتها را در بعد آخر سافتمکس گرفته ایم تا مجموعش 1 شود. این مقادیر نیز حدود 99.8 درصد هستند که خیلی بالا است و مشخصا مدل می‌تواند به راحتی این تسک را انجام دهد.

2) شبکه عصبی عمیقی طراحی کنید که به کمک آن بتوانید داده های فارسی را طبقه بندی کنید (از مدل ParsBERT استفاده کنید).

در این بخش همان موارد بالا را صرفا با استفاده از مدل HooshvareLab/bert-base-parsbert-uncased اجرا می‌کنیم. ضمنا در بخش توکنایز کردن به جای استفاده از ستون source که انگلیسی بود از تارگت که متون فارسی است استفاده می‌کنیم.

```
model_checkpoint = "HooshvareLab/bert-base-parsbert-uncased"
tokenizer = AutoTokenizer.from_pretrained(model_checkpoint)
model = AutoModelForSequenceClassification.from_pretrained(model_checkpoint, num_labels=3)

metric = load_metric("accuracy")
def compute_metrics(eval_pred):
    logits, labels = eval_pred
    predictions = np.argmax(logits, axis=-1)
    return metric.compute(predictions=predictions, references=labels)

str_to_int = {"quran": 0, "bible": 1, "mizan": 2}
def tokenize_function(examples):
    tokenized_batch = tokenizer(examples["targets"], truncation=True, max_length=128)
    tokenized_batch["label"] = [str_to_int[label] for label in examples["category"]]
    return tokenized_batch

tokenized_datasets = dataset.map(tokenize_function, batched=True)
print(tokenized_datasets["train"][0])
tokenized_datasets

training_args = TrainingArguments(
    output_dir="q2_parsbert",
    evaluation_strategy="epoch",
    logging_steps = 20,
    learning_rate=3e-5,
    num_train_epochs=10,
    per_device_train_batch_size=32,
    per_device_eval_batch_size=32,
    save_total_limit = 1,
#     load_best_model_at_end=True,
#     save_strategy = "epoch",
#     metric_for_best_model="accuracy",
    group_by_length = True,
    seed=0,
)
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=tokenized_datasets["train"],
    eval_dataset=tokenized_datasets["valid"],
    tokenizer=tokenizer,
    compute_metrics=compute_metrics,
)
trainer.train()
```

[3940/3940 18:17, Epoch 10/10]

Epoch	Training Loss	Validation Loss	Accuracy
1	0.103600	0.127444	0.954074
2	0.077000	0.122845	0.969259
3	0.021500	0.179542	0.963333
4	0.000400	0.161452	0.974074
5	0.000100	0.165337	0.973704
6	0.005600	0.173197	0.974074
7	0.000000	0.174485	0.975556
8	0.000800	0.174245	0.975926
9	0.000000	0.180969	0.975926
10	0.000000	0.176583	0.975556

```
{'eval_loss': 0.1851060390472412, 'eval_accuracy': 0.9740740740740741, 'eval_runtime': 12.23, 'eval_samples_per_second':
20.769, 'eval_steps_per_second': 6.95, 'epoch': 10.0}
              precision    recall  f1-score   support

       quran       0.98      0.98      0.98       900
       bible       0.97      0.97      0.97       900
       mizan       0.97      0.97      0.97       900

    accuracy                           0.97      2700
   macro avg       0.97      0.97      0.97      2700
weighted avg       0.97      0.97      0.97      2700

AUC-ovr 0.9979840534979423
AUC-ovo 0.9979840534979424
```

شکل 9 - آموزش مدل فارسی و تست آن

در این نتایج نیز دقتهای حدود 97 , 98 درصد و auc حدود 99.79 است. دقتها باز هم خیلی بالا هستند و نشان می‌دهد که مدل فارسی نیز به خوبی از عهده این تسک بر می‌آید. تنها اندکی این مدل از مدل انگلیسی بدتر شد که می‌توان آن را بخاطر بهتر بودن pre-training objective مدل الکترا در بخش قبل دانست که نسبت به برت انگلیسی نشان داده شده بهتر است و در این بخش هم پارس برت براساس همان برت اولیه است و نباید انتظار زیادی نسبت به الکترا از آن داشت. (ضمنا موارد دیگری مثل اندازه بزرگتر دیتای انگلیسی، مورفولوژی کمتر آن و ... هم ذکر کرد.)

3) شبکه عصبی عمیقی طراحی کنید که به کمک آن بتوانید داده های انگلیسی و فارسی را به همراه هم (راهنمایی: داده های انگلیسی و ترجمه شده فارسی آن را با یک تگ<SEP>به هم بچسبانید) طبقه بندی کنید (می توانید از یکی از مدل های چند زبانه مثل XLM-RoBERTa و یا... استفاده کنید).

در این بخش نیز مثل بخش قبل کار می‌کنیم با این تفاوت که مدل را xlm-roberta-base قرار می‌دهیم. ضمنا به توکنایزر هر دوی سورس و تارگت را می‌دهیم که سپس خودش با استفاده از توکن SEP این دو را به هم وصل می‌کند.همچنین باید اشاره کرد که به علت محدودیت منابع (8 گیگابایت حافظه گرافیکی سیستم) امکان اجرای این مدل با بچ سایز 16 نبود. علت این اتفاق نسبت به مدلهای قبل این است که این مدل چندزبانه است و ووکب آن اندازه ای حدود 200 هزار دارد. همین باعث می‌شود که از 200هزار لازم باشد وزن های امبدینگ داشته باشیم به وکتور های 768 که تعداد خیلی بیشتری وزن نسبت به مدلهای قبلی که ووکبشان مخصوص یک زبان و خیلی کوچکتر بود است. این موضوع در وزن مدل هم دیده می‌شود که حدودا دو برابر وزن مدلهای قبلی در زمان دانلود است. به همین دلیل امکان اجرای آن روی سیستم من نبود. برای حل مشکل بچ سایز به 8 کاهش پیدا کرد و برای اینکه تعداد step های آموزش برابر بماند تعداد ایپاک از 10 به 5 تغییر داده شد. (ضمنا آموزش مدل بزرگتر طبیعتا کندتر است و این تغییر برای سرعت بیشتر آزمایش هم خوب است.)

```
model_checkpoint = "xlm-roberta-base"
tokenizer = AutoTokenizer.from_pretrained(model_checkpoint)
model = AutoModelForSequenceClassification.from_pretrained(model_checkpoint, num_labels=3)

metric = load_metric("accuracy")
def compute_metrics(eval_pred):
    logits, labels = eval_pred
    predictions = np.argmax(logits, axis=-1)
    return metric.compute(predictions=predictions, references=labels)

str_to_int = {"quran": 0, "bible": 1, "mizan": 2}
def tokenize_function(examples):
    tokenized_batch = tokenizer(examples["source"], examples["targets"], truncation=True, max_length=128)
    tokenized_batch["label"] = [str_to_int[label] for label in examples["category"]]
    return tokenized_batch

tokenized_datasets = dataset.map(tokenize_function, batched=True)
print(tokenized_datasets["train"][0])
tokenized_datasets

training_args = TrainingArguments(
    output_dir="q2_parsbert",
    evaluation_strategy="epoch",
    logging_steps = 20,
    learning_rate=3e-5,
    num_train_epochs=5,
    per_device_train_batch_size=16,
    per_device_eval_batch_size=16,
    save_total_limit = 1,
#     load_best_model_at_end=True,
#     save_strategy = "epoch",
#     metric_for_best_model="accuracy",
    group_by_length = True,
    seed=0,
)
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=tokenized_datasets["train"],
    eval_dataset=tokenized_datasets["valid"],
    tokenizer=tokenizer,
    compute_metrics=compute_metrics,
)
trainer.train()
```

[3940/3940 23:27, Epoch 5/5]

Epoch	Training Loss	Validation Loss	Accuracy
1	0.171500	0.116700	0.975926
2	0.003900	0.109969	0.982963
3	0.094700	0.103452	0.984074
4	0.000700	0.064465	0.989630
5	0.000100	0.052393	0.991852

```
{'eval_loss': 0.06184544786810875, 'eval_accuracy': 0.9918518518518519,
': 150.138, 'eval_steps_per_second': 9.397, 'epoch': 5.0}
              precision    recall  f1-score   support

       quran       0.99      0.99      0.99       900
       bible       0.99      1.00      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

AUC-ovr 0.9992831275720165
AUC-ovo 0.9992831275720165
```

شکل 10 - آموزش مدل چندزبانه و تست آن

این مدل هم به خوبی جواب می‌دهد و دقتهای حدود 99 درصد گرفته و auc نیز حدود 99.92 درصد است. تحلیل در ادامه می‌آید.

- **در نهایت تحلیل خود را بر اساس نتایج به دست آمده بر روی داده های test گزارش دهید.**

طبق نتایجی که دیدیم به صورت نسبی پارس برت از همه بدتر، سپس الکترای انگلیسی و بهترین مدل xlm-r چندزبانه بود.

در مورد الکترا و پارس برت قبلا توضیح داده شد که الکترا pre-training objective متفاوت دارد و از برت انگلیسی بهتر عمل می‌کند در حالیکه پارس برت بر اساس همان برت است و ضمنا گفتیم که انگلیسی دیتای بیشتری نسبت به فارسی دارد (هر چند فارسی هم کم دیتا ندارد) و همچنین طبق درس از نظر مورفولوژی ضعیف تر است که کار مدل را نسبت به فارسی کمی راحت تر می‌کند.

- **آیا مورد سوم (استفاده از مدل های چند زبانی بر روی داده های چند زبانی) باعث بالا بردن دقت شبکه خواهد شد؟**

در مورد سوم که از مدل چندزبانی استفاده کردیم بهترین نتایج را توانستیم بگیریم. بنابراین استفاده از مدل چند زبانی بر روی داده چند زبانی می‌تواند باعث بالا بردن دقت شود. ابتدایی ترین علت این است که این مدل به دو برابر اطلاعات نسبت به دو مدل قبلی دسترسی داشته. و ایده‌ای که هست این است که مثلا شاید در این دیتاست خاص، قرآن در زبان فارسی ویژگی های بارزتری برای جداسازی از دو کتاب دیگر داشته باشد و مثلا در انگلیسی کتاب دیگری این خاصیت را داشته باشد و نثر و استفاده از کلمات خاصی داشته باشد که ترجمه فارسی دیده نشود. بنابراین این دو با این که ترجمه هستند اما در هر کدام اطلاعاتی هست که در دیگری نیست و استفاده از مدل چندزبانه که از هر دو استفاده کند همانطور که دیده شد، می‌تواند بهترین نتایج را داشته باشد. البته باید توجه داشت که طبق سوال قبلی که دیدیم، مدل چندزبانه نسبت به یک زبان خاص، معمولا نمی‌تواند نتایج بهتری داشته باشد چون در خودش خیلی از زبانها را گنجانده. اما در مسائلی که ورودی می‌تواند از چند زبان باشد قطعا استفاده از آنها می‌تواند نسبت به مدل های تک زبانه برتری ایجاد کند.

3) امتیازی Cross-lingual zero-shot transfer learning

1) انتظار شما از Performance مدل بر روی داده های test زبان فارسی، قبل از اجرای این مدل چیست؟

اولا باید گفت که چون داریم به صورت cross-lingual کار می‌کنیم قطعا دقت کمتر از حالت ترین و تست روی یک زبان خواهد شد. اما باید اشاره کرد که مدلهای چندزبانه مثل XLM-R یاد گرفته‌اند که از کلمات و مفاهیم مشابه در زبان های مختلف، بازنمایی های مشابهی بسازند. بنابراین اگر مدل بتواند روی یکی از این زبان‌ها دسته بندی را انجام دهد، به علت همین شباهتی که مدل یاد گرفته می‌توانیم روی هر زبان دیگری هم احتمالا این کار را انجام دهیم. بنابراین باید انتظار دقت نسبتا خوبی را داشته باشیم اما طبیعتا نه به اندازه مدلی که کاملا ترین و تستش In Distribution بوده است. (اتفاقا در مقاله https://aclanthology.org/2022.acl-long.144/ که اخیرا منتشر شده است نیز به پروبینگ همین مدل پرداخته شده و نشان داده شده است که قابلیت cross-lingual عمل کردن را به خوبی در بازنمایی های خود دارد.)

2) بعد از اجرای این مدل آیا انتظارات پیشین شما برآورده شده است؟دلیل این Performance ای که گرفته اید چیست؟

در این بخش مدل را مانند سوال قبل اجرا می‌کنیم تنها تفاوت این است که دیگر هم سورس هم تارگت را به مدل نمی‌دهیم. بلکه فقط سورس که انگلیسی است را می‌دهیم و سپس بر روی بخش تارگت که فارسی است تست می‌کنیم. در تصویر زیر، آموزش مدل و تغییرات لاس و دقت در طول ایپاک ها آمده است.

```
model_checkpoint = "xlm-roberta-base"
tokenizer = AutoTokenizer.from_pretrained(model_checkpoint)
model = AutoModelForSequenceClassification.from_pretrained(model_checkpoint, num_labels=3)

metric = load_metric("accuracy")
def compute_metrics(eval_pred):
    logits, labels = eval_pred
    predictions = np.argmax(logits, axis=-1)
    return metric.compute(predictions=predictions, references=labels)

str_to_int = {"quran": 0, "bible": 1, "mizan": 2}
def tokenize_function(examples):
    tokenized_batch = tokenizer(examples["source"], truncation=True, max_length=128)
    tokenized_batch["label"] = [str_to_int[label] for label in examples["category"]]
    return tokenized_batch

tokenized_datasets = dataset.map(tokenize_function, batched=True)
print(tokenized_datasets["train"][0])
tokenized_datasets

training_args = TrainingArguments(
    output_dir="q2_parsbert",
    evaluation_strategy="epoch",
    logging_steps = 20,
    learning_rate=3e-5,
    num_train_epochs=5,
    per_device_train_batch_size=16,
    per_device_eval_batch_size=16,
    save_total_limit = 1,
#     load_best_model_at_end=True,
#     save_strategy = "epoch",
#     metric_for_best_model="accuracy",
    group_by_length = True,
    seed=0,
)
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=tokenized_datasets["train"],
    eval_dataset=tokenized_datasets["valid"],
    tokenizer=tokenizer,
    compute_metrics=compute_metrics,
)
trainer.train()
```

[3940/3940 18:47, Epoch 5/5]

Epoch	Training Loss	Validation Loss	Accuracy
1	0.268000	0.150862	0.958889
2	0.109400	0.129471	0.971481
3	0.081300	0.173883	0.967037
4	0.022600	0.110631	0.980000
5	0.053800	0.124389	0.980000

شکل 11 - آموزش مدل چندزبانه و تست آن

پس از این مرحله که آموزش مدل تکمیل شد، حالا دقت روی فارسی را می‌سنجیم.

```
{'eval_loss': 0.10795055329799652, 'eval_accuracy': 0.9825925925925926,
': 237.46, 'eval_steps_per_second': 14.863, 'epoch': 5.0}
ID
              precision    recall  f1-score   support

       quran       0.99      0.97      0.98       900
       bible       0.97      0.99      0.98       900
       mizan       0.98      0.98      0.98       900

    accuracy                           0.98      2700
   macro avg       0.98      0.98      0.98      2700
weighted avg       0.98      0.98      0.98      2700

AUC-ovr 0.9987145061728396
AUC-ovo 0.9987145061728396
```

```
{'eval_loss': 1.515942931175232, 'eval_accuracy': 0.7925925925925926,
213.567, 'eval_steps_per_second': 13.368, 'epoch': 5.0}
OOD
              precision    recall  f1-score   support

       quran       0.79      0.76      0.78       900
       bible       0.82      0.64      0.72       900
       mizan       0.77      0.98      0.86       900

    accuracy                           0.79      2700
   macro avg       0.80      0.79      0.79      2700
weighted avg       0.80      0.79      0.79      2700

AUC-ovr 0.9203460905349795
AUC-ovo 0.9203460905349794
```

شکل 12 - مقایسه تست ID و OOD

همانطور که دیده می‌شود، دقتها وقتی روی تست انگلیسی باشیم که ID محسوب می‌شود حدود 98 و auc حدود 0.9987 است که خیلی بالا است و مناسب است.

اما نکته جالب این است که وقتی روی تارگت که فارسی است و OOD محسوب می‌شود به دقت حدود 78 تا 86 می‌رسیم و auc حدود 0.9203.
در اینجا می‌بینیم که مشخصا دقت خیلی کمتر از ID و حالتهای قبلی فارسی است. اما باید توجه داشت که همین دقتها برای سه کلاس بسیار خوب است. براساس np.unique می‌بینیم که

(array([0, 1, 2]), array([900, 900, 900], dtype=int64))

یعنی لیبل های تست بالانس هستند بنابراین دقت رندوم حدود 33 درصد می‌شود. اینکه توانستیم بدون اینکه حتی روی بخش فارسی آموزش دهیم، به دقت بالای 78 رسیدیم یعنی خوب عمل شده. علت آن نیز در قبل هم توضیح داده شده که این مدل از تمام زبانها در فضای یکسان بازنمایی ساخته و امیدوار هم هستیم که مفاهیم مشابه نزدیک هم باشند و بنابراین دسته بندی بر اساس یک زبان و مرزبندی این فضا در یک زبان باید تا حدی بشود به بقیه زبان‌ها نیز تعمیم یابد که این را دیدیم. بنابراین دقیقا مطابق انتظار، دقت خوب است اما طبیعتا نه به خوبی وقتی که روی خود فارسی آموزش داده شود.

3) در چه مواقعی از Cross-lingual zero-shot transfer learning استفاده می کنیم در واقع کاربرد آن را توضیح دهید.

مهم‌ترین کاربرد cross-lingual zero-shot وقتی است که دیتای کافی در یک زبان نداریم و در زبان‌های دیگر داریم. به همین دلیل با استفاده از زبانهایی که دیتای بیشتری داریم مدل را می‌توانیم آموزش دهیم و سپس روی زبان low-resource از آن استفاده کنیم. این روش به این علت کار می‌کند که احتمالا در هر دو زبان courpus خام داشتیم و مدل با آنها پیش‌آموزش دیده و توانسته مفاهیم دو طرف را در یک فضا embed کند ولی در یک تسک خاص که دیتا برای یک زبان خاص نداریم با این کار از این تشابه سازی انجام شده بهره می‌بریم و آن را از یک زبان با منبع به تسک خاص خود که دیتای کافی ندارد در یک زبان کم منبع تعمیم می‌دهیم. در مجموع آموزش مدل‌های عمیق برای تسکها نیازمند دیتای تگ خورده بزرگ است که در اکثر زبان‌ها کمیاب است. بنابراین استفاده از تسکهای زبانهای غنی تر مثل انگلیسی (و در حالت بهتر کمی هم از همان زبان low resource) می‌تواند نتایج مناسبی دهد که با دیتای کم همان زبان لزوما شاید به دست نمی‌آمد.