# Assignment 6

# NLP


PREPARED BY

**Samin Mehdizadeh - 810100526**

**Mohsen Fayyaz - 810100524**


Summer 2022

# فهرست مطالب

# QA 1

## Abstract And Introduction

این پروژه در مورد آموزش یا fine-tune کردن مدل پارس برت و آلبرت فارسی روی چند دیتاست سوال و جواب به صورت extractive است که یعنی باید جواب سوال را در یک context پیدا کرد. در این تمرین باید در مجموع 2 مدل و 4 دیتاست را آزمایش می‌کردیم که می شود 8 حالت.

برای این کار ابتدا داده ها بارگذاری و توکنایز شدند و سپس به صورت یک تسک پرسش و پاسخ با استفاده از کتابخانه هاگینگ فیس آموزش داده شد. برای ارزیابی مدل نیز از معیارهای F1 Score و Exact Match استفاده کردیم.

مدل های آموزش دیده در https://huggingface.co/mohsenfayyaz هستند و نتایج مناسب بر اساس معیارهای بررسی شده به دست آمد. به طور خلاصه، یکسان بودن توزیع داده آموزش با تست و بیشتر بودن تعداد نمونه‌های دیتاست باعث بهبود نتایج می‌شود و از نظر مدل، برت که وزنهای خیلی بیشتری نسبت به وزنهای مشترک آلبرت دارد می‌تواند بهتر عمل کند.

در ادامه توضیح کامل و با جزئیات از پیاده سازی، آموزش، ارزیابی و نتایج می‌آید.

## Implementation

در ابتدا کتابخانه های مورد نیاز مانند transformers و datasets را نصب و سپس وارد می‌کنیم.

در اولین قدم یک کلاس DatasetLoader نوشتیم تا 4 نوع دیتاست خواسته شده را به صورت یکپارچه بارگذاری کند و از حالت json به دیتاست تبدیل کند که آماده استفاده برای آموزش مدل باشد.

در اینجا به توضیح کد می‌پردازیم اما خود کد در کنار این فایل ارسال شده است.
پس از خواندن دیتاست ها از فایل json آنها را در یک DataFrame از Pandas ذخیره و سپس با استفاده از datasets.Dataset.from_pandas آنرا تبدیل به آبجکت دیتاست کردیم. ضمنا هر بخش از train, validation, test را که یک دیتاست هستند در یک datasets.DatasetDict قرار دادیم. همانطور که در تلگرام اطلاع داده شد، برای تمام دیتاست‌ها از بخش تست دیتاست pquad استفاده کردیم و دیتاست هایی که بخش val نداشتند، از تستشان به عنوان dev استفاده کردیم.

در ادامه به توکنایز کردن دیتاست پرداختیم. برای این کار از لینکهای زیر کمک گرفتیم.

https://huggingface.co/docs/transformers/tasks/question_answering
https://huggingface.co/tasks/question-answering
https://huggingface.co/course/chapter7/7

برای توکنایز کردن دیتاست از توکنایزر هاگینگ فیس استفاده کردیم. به این ترتیب که سوال و کانتکست را به عنوان ورودی می‌دهیم و آرگومان های دیگر را مشخص می‌کنیم و توکنایز انجام می‌شود. در این روند، توکنهای خاصی نیز اضافه می‌شوند و جمله به شکل زیر می‌شود.

[CLS] Question [SEP] Context [SEP]

که CLS برای classification و SEP به معنای seperator یا جدا کننده است که سوال را از کانتکست جدا کرده است. در آخر مهمترین بخش خروجی توکنایزر input_ids است که بعد از توکنایز کردن، عدد هر توکن را نوشته است، و ورودی مدل همین اعداد است. (به همراه attention mask و باقی خروجی های توکنایزر که توضیح بیشتر در پروژه های قبلی داده شده است)

برای توکنایز کردن "truncation="only_second قرار می‌دهیم که یعنی فقط از تیکه دوم که context است زده شود و نه از سوال. ضمنا return_offsets_mapping=True قرار می‌دهیم تا

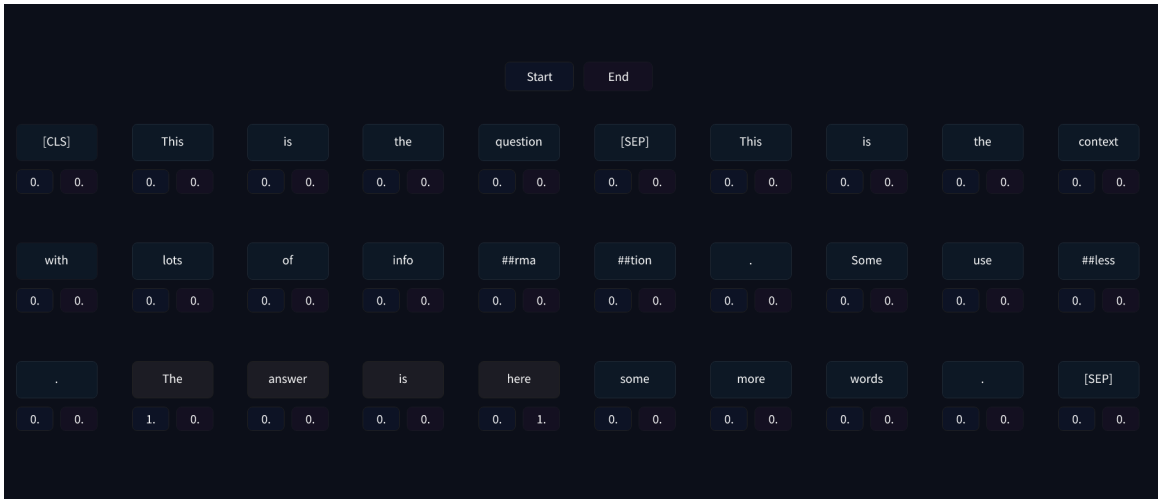
بعد از توکنایز شدن، محل هر توکن روی ورودی اولیه را داشته باشیم که به شکل [(start_char, end_char), ...] است و در ادامه از آن برای پیدا کردن بخشی که در دیتاست مشخص شده است استفاده می‌کنیم. حداکثر طول را 400 قرار می‌دهیم که در ادامه می‌بینیم که حدود طول جملات همینقدر است حداکثر و مقدار مناسبی است. ضمنا از return_overflowing_tokens=True را نیز می‌توانیم استفاده کنیم که هر چقدر از 400 بیشتر شد را به عنوان یک نمونه جدید داشته باشیم و چیزی از دست نرود. (این می‌تواند با stride هم ترکیب شود که مقدار overlap در صورت شکسته شدن را نشان می‌دهد.)

همانطور که گفته شد در ادامه، با استفاده از کاراکتر شروع که در دیتاست مشخص شده است، تعداد کاراکتر متن جواب، و offset هایی که قبلا برگردانده شد، محل دقیق توکن شروع و پایان را به دست می‌آوریم و در خروجی توکنایزر می‌گذاریم.

تا اینجای کار، آماده سازی دیتاست انجام شد و به سراغ آموزش مدل می‌رویم. در ایتدای کد آموزش، مدل و توکنایزر و دیتاست را بارگذاری می‌کنیم و از AutoModelForQuestionAnswering استفاده می‌کنیم.

```
AlbertForQuestionAnswering(
  (albert): AlbertModel(
    (embeddings): AlbertEmbeddings(
      (word_embeddings): Embedding(80000, 128, padding_idx=0)
      (position_embeddings): Embedding(512, 128)
      (token_type_embeddings): Embedding(2, 128)
      (LayerNorm): LayerNorm((128,), eps=1e-12, elementwise_affine=True)
      (dropout): Dropout(p=0, inplace=False)
    )
    (encoder): AlbertTransformer(
      (embedding_hidden_mapping_in): Linear(in_features=128, out_features=768, bias=True)
      (albert_layer_groups): ModuleList(
        (0): AlbertLayerGroup(
          (albert_layers): ModuleList(
            (0): AlbertLayer(
              (full_layer_layer_norm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
              (attention): AlbertAttention(
                (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)
                (attention_dropout): Dropout(p=0, inplace=False)
                (output_dropout): Dropout(p=0, inplace=False)
                (dense): Linear(in_features=768, out_features=768, bias=True)
                (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
              )
              (ffn): Linear(in_features=768, out_features=3072, bias=True)
              (ffn_output): Linear(in_features=3072, out_features=768, bias=True)
              (activation): NewGELUActivation()
              (dropout): Dropout(p=0, inplace=False)
            )
          )
        )
      )
    )
  )
  (qa_outputs): Linear(in_features=768, out_features=2, bias=True)
)
```

این کار یک لایه خروجی بالای مدل اضافه می‌کند که همانطور که دیده می‌شود یک لایه با ورودی 768 که ابعاد representation است را می‌گیرد و به ابعاد 2 می‌برد که اولی نشان می‌دهد این توکن شروع جواب هست یا نه، و دومی نشان می‌دهد این توکن پایانی است یا نه. (در بالا چون آلبرت را آوردیم و این مدل وزن لایه ها را share می‌کند بین لایه ها، تنها یک لایه چاپ شده است که در اصل 12 بار اجرا می‌شود. اگر برت چاپ می شد 12 لایه مجزا دیده می‌شد.)




و آموزش مدل هم به همین شکل است که باید دقیقا روی توکن شروع بعد اول 1 شود و روی توکن آخر بعد دوم 1 یک شود و باقی خروجی ها همگی 0 باشند.

برای آموزش مدل از آرگومان های زیر استفاده کردیم.

```
training_args = TrainingArguments(
      output_dir="./results",
      evaluation_strategy="epoch",
      learning_rate=2e-5,
      per_device_train_batch_size=16,
      per_device_eval_batch_size=16,
      num_train_epochs=3,
      weight_decay=0.01,
      group_by_length=True,
      logging_steps=20
)
```

این مقادیر همان مقادیری هستند که در لینکهای قبلی پیشنهاد شده بودند. نرخ یادگیری 2e-5 و تعداد ایپاک 5. البته در بخش ablation تغییراتی هم ایجاد می‌کنیم و تاثیر را می‌بینیم. ضمنا group

by length را قرار دادیم که باعث می‌شود بچ ها را سعی کند از نمونه های هم اندازه بسازد تا نیاز کمتری به padding باشد که مستقیما با زمان اجرا رابطه دارد.

برای ارزیابی مدل‌ها از دو معیار f1 و exact match استفاده می‌کنیم.

معیار exact match معیاری است که بر اساس تطابق دقیق کاراکتر پاسخ پیش‌بینی‌شده و پاسخ درست است. برای پاسخ هایی که به درستی پیش بینی شده اند، تطابق دقیق 1 خواهد بود. حتی اگر فقط یک کاراکتر متفاوت باشد، تطابق دقیق 0 خواهد بود.

معیار F1-Score در صورتی مفید است که هر دو مثبت کاذب و منفی کاذب را به یک اندازه ارزش گذاری کنیم. امتیاز F1 بر روی هر کلمه در ترتیب پیش بینی شده در مقابل پاسخ صحیح محاسبه می شود.

این دو معیار در متریک هایی که برای دیتاست squad قرار دارند ("load_metric("squad

محاسبه می‌شوند و به همین دلیل از آن استفاده می‌کنیم. نمونه ای از آن در زیر دیده می‌شود.

```
metric.compute(predictions=[{'id': '1719422', 'prediction_text': '۱۸۹۹'}],
               references=[{'id': '1719422', 'answers': [
                   {'answer_start': 11, 'text': '۱۸۹۹ سال'},
                   {'answer_start': 15, 'text': '۱۸۹۹'},
               ]}])
```

```
{'exact_match': 100.0, 'f1': 100.0}
```

همانطور که دیده می‌شود در داده های تست، رفرنس ها بعضا چندین جواب دارند که با دادن همه آنها به این متریک، خودش بهترین مچ را در نظر می‌گیرد. برای محاسبه این مقدار، نمونه هایی که جوابی ندارند را کنار گذاشتیم. (در squad v2 این موارد هم در نظر گرفته شدند، اما برای کاهش پیچیدگی و چون در سوال خواسته نشده بود همان هایی که جواب داشتند بسنده کردیم.)

برای به دست آوردن خروجی مدل از pipeline هاگینگ فیس استفاده می‌کنیم.

```
pipeline("question-answering", model=self.model,
                tokenizer=self.tokenizer, device=0)
```

به این ترتیب می‌توان به این پایپلاین مجموعه ای از سوالات و context ها داد و خروجی را که به شکل زیر است گرفت.

{'score': 0.8518245816230774, 'start': 14, 'end': 19, 'answer': ' ۱۸۹۹'}

(این کار را می‌توان به صورت دستی هم انجام داد و با استفاده از تابع predict در trainer خروجی خام مدل را برای تست گرفت و سپس خودمان ماکسیمم شروع و خروجی را به دست آوریم و بعد خروجی را از روی آن اعداد به صورت رشته بسازیم. اما چون شناخت از هاگینگ فیس داریم، این کار را راحت تر و سریعتر با پایپلاین انجام دادیم.)
مقدار score میزان اطمینان مدل را از جوابش نشان می‌دهد که همان میانگین احتمال توکن شروع و احتمال توکن خروجی است.

سپس این ها را به همراه جواب های reference به متریک دادیم و نتایج را به دست آوردیم که در ادامه می‌آید.

## Results/Analysis

### 1. PQuAD

https://huggingface.co/mohsenfayyaz/bert-base-parsbert-uncased_pquad

در زیر نتایج در طول آموزش دیده می‌شود.

[1110/1110 11:30, Epoch 3/3]

| Epoch | Training Loss | Validation Loss |
|---|---|---|
| 1 | 1.345300 | 1.514629 |
| 2 | 0.711100 | 1.520786 |
| 3 | 0.570400 | 1.619204 |

و در زیر نتیجه نهایی روی بخش تست دیده می‌شود.

{'exact_match': 57.025920873124 15, 'f1': 78.45450893098601}

در خود مقاله https://arxiv.org/pdf/2202.06219.pdf مقادیر 61.5 و 79.8 آمده است که می‌بینیم بنابراین آموزش ما با این منابع محدود مناسب پاسخ داده است و توانستیم مقادیر مناسبی را نسبت به بیس لاین مقاله کسب کنیم.

https://huggingface.co/mohsenfayyaz/albert-fa-base-v2_pquad

در زیر نتایج در طول آموزش دیده می‌شود.

[1110/1110 11:59, Epoch 3/3]

| Epoch | Training Loss | Validation Loss |
|---|---|---|
| 1 | 1.313200 | 1.596055 |
| 2 | 0.773700 | 1.573966 |
| 3 | 0.574700 | 1.694198 |

و در زیر نتیجه نهایی روی بخش تست دیده می‌شود.

{'exact_match': 49.931787175989086, 'f1': 73.16670892152278}

تفاوت دقت برت و آلبرت که آلبرت کمتر شده است طبیعی است بخاطر اشتراک گذاری وزنها آلبرت حجم کل حدود 100 مگابایت دارد و برت حدود 600 مگابایت است. بنابراین از نظر تئوری گنجایش برت بیشتر است. (مقاله آلبرت https://arxiv.org/abs/1909.11942 هم ادعای بهتر شدن در این حالت را ندارد و در حالتهای بزرگتر این ادعا را داشته است.)

نمونه خروجی:

تیم آینتراخت فرانکفورت در چه سالی تاسیس شد؟

این تیم در سال ۱۸۹۹ تأسیس شد و تا به حال موفق به کسب یک عنوان قهرمانی در لیگ آلمان، ۵ عنوان قهرمانی در جام حذفی آلمان به‌همراه یک نایب قهرمانی در جام باشگاههای اروپا و همچنین یک قهرمانی در جام یوفا شده‌است. از سال ۱۹۲۵ به بعد، ورزشگاه اختصاصی این باشگاه ورزشگاه والداستادیون بوده که در سال ۲۰۰۵ نام آن به کومرتسبانک آرنا تغییر یافت.

{'score': 0.9836439490318298, 'start': 14, 'end': 19, 'answer': ' ۱۸۹۹'}

## 2. PersianQA

https://huggingface.co/mohsenfayyaz/bert-base-parsbert-uncased_persian_qa

در زیر نتایج در طول آموزش دیده می‌شود.

[1185/1185 16:39, Epoch 3/3]

| Epoch | Training Loss | Validation Loss |
|---|---|---|
| 1 | 2.806000 | 2.328210 |
| 2 | 1.895900 | 2.149322 |
| 3 | 1.571000 | 2.220667 |

و در زیر نتیجه نهایی روی بخش تست دیده می‌شود.

{'exact_match': 43.38335607094134, 'f1': 68.27823270914114}

https://huggingface.co/mohsenfayyaz/albert-fa-base-v2_persian_qa

در زیر نتایج در طول آموزش دیده می‌شود.

[1185/1185 18:10, Epoch 3/3]

| Epoch | Training Loss | Validation Loss |
|---|---|---|
| 1 | 2.454100 | 2.311774 |
| 2 | 1.865600 | 2.161770 |
| 3 | 1.396800 | 2.273769 |

و در زیر نتیجه نهایی روی بخش تست دیده می‌شود.

{'exact_match': 40.791268758526606, 'f1': 66.9052405879214}

بدتر شدن دقت نسبت به قسمت قبل به این علت است که دیگر داده تست با داده آموزش از یک توزیع نبودن و اصطلاحا In Distribution نیست و به سمت OOD یا Out of Distribution رفتیم و طبیعتا دقت کمی بدتر شده است. اما هنوز هم مقادیر نسبتا خوبی به دست آمده است.

نمونه خروجی:

تیم آینتراخت فرانکفورت در چه سالی تاسیس شد؟

این تیم در سال ۱۸۹۹ تأسیس شد و تا به حال موفق به کسب یک عنوان قهرمانی در لیگ آلمان، ۵ عنوان قهرمانی در جام حذفی آلمان به‌همراه یک نایب قهرمانی در جام باشگاههای اروپا و همچنین یک قهرمانی در جام یوفا شده‌است. از سال ۱۹۲۵ به بعد، ورزشگاه اختصاصی این باشگاه ورزشگاه والداستادیون بوده که در سال ۲۰۰۵ نام آن به کومرتسبانک آرنا تغییر یافت.

{'score': 0.5821070671081543, 'start': 7, 'end': 19, 'answer': ' ۱۸۹۹ در سال'}

### 3. ParSQuAD

https://huggingface.co/mohsenfayyaz/bert-base-parsbert-uncased_parsquad

در زیر نتایج در طول آموزش دیده می‌شود.




| Epoch | Training Loss | Validation Loss |
|---|---|---|
| 1 | 1.261600 | 1.856759 |
| 2 | 0.995900 | 1.820132 |
| 3 | 0.700200 | 2.048670 |

و در زیر نتیجه نهایی روی بخش تست دیده می‌شود.

{'exact_match': 52.796725784447474, 'f1': 69.86877786355724}

https://huggingface.co/mohsenfayyaz/albert-fa-base-v2_parsquad

در زیر نتایج در طول آموزش دیده می‌شود.




| Epoch | Training Loss | Validation Loss |
|---|---|---|
| 1 | 1.276000 | 1.817525 |
| 2 | 0.993400 | 1.833020 |
| 3 | 0.769300 | 2.000509 |

و در زیر نتیجه نهایی روی بخش تست دیده می‌شود.

{exact_match': 45.839017735334245, 'f1': 67.19522968538176'}

تفاوت کم دقت و بهتر شدنش نسبت به قبلی این است که این دیتاست بزرگتر بوده و طبیعتا نمونه های بیشتر باعث می‌شود مدل توزیع بزرگتری را آموزش ببیند و بتواند بهتر عمل کند.

### 4. PQuAD and PersianQA

https://huggingface.co/mohsenfayyaz/bert-base-parsbert-uncased_pquad_and_persian_qa

در زیر نتایج در طول آموزش دیده می‌شود.

[2292/2292 28:08, Epoch 3/3]

| Epoch | Training Loss | Validation Loss |
|---|---|---|
| 1 | 1.815100 | 1.675753 |
| 2 | 1.030300 | 1.679921 |
| 3 | 0.583000 | 1.822376 |

و در زیر نتیجه نهایی روی بخش تست دیده می‌شود.

{'exact_match': 59.34515688949522, 'f1': 79.58011430218735}

https://huggingface.co/mohsenfayyaz/albert-fa-base-v2_pquad_and_persian_qa

در زیر نتایج در طول آموزش دیده می‌شود.

[2292/2292 30:36, Epoch 3/3]

| Epoch | Training Loss | Validation Loss |
|---|---|---|
| 1 | 1.744200 | 1.695394 |
| 2 | 1.068300 | 1.660845 |
| 3 | 0.640600 | 1.813202 |

و در زیر نتیجه نهایی روی بخش تست دیده می‌شود.

{'exact_match': 51.841746248294676, 'f1': 75.72671789957992}

این مورد بهترین نتایج را در تمام آزمایشها گرفت.

نمونه خروجی:
تیم آینتراخت فرانکفورت در چه سالی تاسیس شد؟

این تیم در سال ۱۸۹۹ تأسیس شد و تا به حال موفق به کسب یک عنوان قهرمانی در لیگ آلمان، ۵ عنوان قهرمانی در جام حذفی آلمان به‌همراه یک نایب قهرمانی در جام باشگاههای اروپا و همچنین یک قهرمانی در جام یوفا شده‌است. از سال ۱۹۲۵ به بعد، ورزشگاه اختصاصی این باشگاه ورزشگاه والداستادیون بوده که در سال ۲۰۰۵ نام آن به کومرتسبانک آرنا تغییر یافت.

{'score': 0.9521518349647522, 'start': 15, 'end': 19, 'answer': '۱۸۹۹'}

## Conclusion

| Model | Dataset | F1 | Exact Match |
|---|---|---|---|
| **parsbert** | **pquad** | 78.45 | 57.02 |
| **parsbert** | **persian_qa** | 68.27 | 43.38 |
| **parsbert** | **parsquad** | 69.86 | 52.79 |
| **parsbert** | **pquad_and_persian_qa** | **79.58** | **59.34** |
| **albert** | **pquad** | 73.16 | 49.93 |
| **albert** | **persian_qa** | 66.90 | 40.79 |
| **albert** | **parsquad** | 67.19 | 45.83 |
| **albert** | **pquad_and_persian_qa** | 75.72 | 51.84 |

همانطور که دیدیم مدلی که روی ترکیبی از pquad و persianQA آموزش دید توانست بهترین نتایج را بگیرد. علت این موضوع این است که این ترکیب هم مقدار داده بیشتری دارد که برای مدل مفید است، و ضمنا داده آموزش pquad که از تستش برای ارزیابی استفاده کردیم هم حضور دارد و مدل با توزیع نهایی که برای تست استفاده می‌کنیم آشنایی دارد.

ضمنا همین موضوع را دیدیم که وقتی pquad باشد دقتها خیلی بیشتر می‌شوند ولی وقتی نباشد بدتر.

همچنین دیتاست parsquad که بزرگتر بود توانست نسبت به persian_qa نتایج بهتری بگیرد.

و در آخر از نظر مقایسه مدل‌ها، چون آلبرت تعداد وزنهای یکتای خیلی کمتری دارد به علت اشتراک گذاری وزنها، قدرت کمتری نیز در سایز معادل برت بیس دارد و نتایج نشان می‌دهد که به خوبی برت عادی عمل نکرده است. (البته که آلبرت با همان وزن از نظر حافظه خیلی مدل بزرگتری است و زمان inference بیشتری نیاز دارد ولی می‌تواند برت را پشت سر بگذارد که در اینجا به علت محدودیت منابع سراغ مدلهای بزرگتر نرفتیم.)

## Ablation Study

در این بخش تغییر تعداد اپیاک و نرخ یادگیری را می‌سنجیم. به علت محدودیت منابع و تستهای بسیار زیاد این پروژه، به تغییر همین دو بسنده می‌کنیم.

ابتدا یادآوری می‌کنیم که مدل برت با 3 اپیاک و نرخ یادگیری 2e-5 روی pquad نتیجه زیر را داشت.

{'exact_match': 57.02592087312415, 'f1': 78.45450893098601}

https://huggingface.co/mohsenfayyaz/bert-base-parsbert-uncased_pquad_1epoch

در زیر نتایج در طول آموزش دیده می‌شود.

[370/370 03:44, Epoch 1/1]

| Epoch | Training Loss | Validation Loss |
|---|---|---|
| 1 | 1.550000 | 1.695953 |

و در زیر نتیجه نهایی روی بخش تست دیده می‌شود.

{'exact_match': 54.70668485675307, 'f1': 74.21113530614612}

همانطور که دیده می‌شود، 3 اپیاک قبلی بهتر بوده و مدل این 3 بار حرکت روی دیتاست را برای یادگیری بهتر واقعا نیاز داشته است.

https://huggingface.co/mohsenfayyaz/bert-base-parsbert-uncased_pquad_lr1e-5

در زیر نتایج در طول آموزش دیده می‌شود.

[1110/1110 11:30, Epoch 3/3]

| Epoch | Training Loss | Validation Loss |
|---|---|---|
| 1 | 1.674000 | 1.779795 |
| 2 | 1.015000 | 1.637098 |
| 3 | 0.983600 | 1.632841 |

و در زیر نتیجه نهایی روی بخش تست دیده می‌شود.

{exact_match': 54.979536152796726, 'f1': 76.21874003041727'}

در اینجا هم کاهش نرخ یادگیری و ثابت نگه داشتن تعداد اپیاک باعث بدتر شدن دقت شد که نشان می‌دهد همان نرخ یادگیری اولیه مناسب بوده است.

البته که برای پیدا کردن بهترین هایپرپارامترها بهتر است از روشهای خودکار استفاده کرد که فضای هایپرپارامترها را جستجو می‌کند و بهترین ها را پیشنهاد می‌دهد و هاگینگ فیس هم این قابلیت را دارد. اما بخاطر محدودیت منابع سراغ این روش نرفتیم.

همچنین برای داشتن نتایج قطعی و قابل استناد در تحقیقات ماشین لرنینگ بهتر است تا یک آزمایش با seed های مختلف تکرار شود و میانگین و std آن گزارش شود که باز هم بخاطر محدودیت منابع این قابلیت وجود نداشت.

اما با وجود همه محدودیت ها توانستیم دقتهایی خیلی نزدیک به خود مقاله به دست آوریم.

# NLU 2

## Abstract And Introduction

در این بخش سعی شده است تا به کمک دیتاست MASSIVE برای فهم زبان یا همان NLU مدلی آموزش داده شود که بتوان به کمک آن با داشتن یک utterance تسک های مربوط به slot filling و intent prediction انجام شود.برای آموزش مدل تنها از دیتاست فارسی استفاده شده است و نتایج به کمک xlm-roberta-base به دست آمده است.

در انتها نیز مدل آموزش داده شده بر روی داده های تست مورد ارزیابی قرار گرفته و نتایج آن با نتایج به دست آمده از مقاله MASSIVE[1] مقایسه شده است.

### بررسی داده ها

همه ی داده های مربوط به زبان فارسی در فایل fa-IR.jsonl وجود دارند.در این فایل تمامی داده های مربوط به train-test-evaluation به کمک برچسب partition مشخص شده اند.به طور کی این دیتاست شامل 60 intent و 56 slot است. که لیبل های مربوط به هر کدام در ادامه آمده است:

Intents type:

```
'iot_hue_lightup', 'alarm_query', 'qa_factoid', 'transport_traffic',
'music_likeness', 'recommendation_movies', 'email_query', 'qa_stock',
'play_game', 'calendar_set', 'transport_ticket', 'qa_maths',
'audio_volume_mute', 'music_dislikeness', 'play_radio', 'social_query',
'music_query', 'email_querycontact', 'transport_taxi',
'recommendation_locations', 'takeaway_order', 'audio_volume_down',
'general_joke', 'iot_hue_lightdim', 'general_quirky', 'cooking_query',
'calendar_query', 'email_addcontact', 'datetime_convert',
'music_settings', 'iot_cleaning', 'play_music', 'general_greet',
'social_post', 'play_podcasts', 'weather_query',
'recommendation_events', 'email_sendemail', 'alarm_set',
'iot_hue_lighton', 'audio_volume_other', 'cooking_recipe', 'iot_coffee',
'iot_hue_lightoff', 'iot_wemo_on', 'datetime_query', 'takeaway_query',
'audio_volume_up', 'transport_query', 'lists_query', 'calendar_remove',
'lists_remove', 'qa_currency', 'iot_hue_lightchange',
'lists_createoradd', 'alarm_remove', 'iot_wemo_off', 'play_audiobook',
'qa_definition', 'news_query'
```

[1] https://arxiv.org/pdf/2204.08582.pdf

Slots type:

```
'app_name',        'person',        'change_amount',        'artist_name',
'weather_descriptor',    'playlist_name',    'radio_name',    'list_name',
'business_type',    'currency_name',    'transport_name',    'timeofday',
'player_setting',      'relation',      'music_album',      'place_name',
'house_place',    'time_zone',    'date',    'device_type',    'meal_type',
'event_name',     'podcast_name',     'coffee_type',     'business_name',
'order_type',        'color_type',        'ingredient',        'game_name',
'transport_descriptor',    'drink_type',    'food_type',    'game_type',
'transport_type',   'definition_word',   'news_topic',   'audiobook_name',
'alarm_type',       'transport_agency',       'Other',       'media_type',
'personal_info',    'podcast_descriptor',    'music_genre',    'joke_type',
'song_name',   'general_frequency',   'audiobook_author',   'sport_type',
'email_folder',  'cooking_type',  'time',  'movie_name',  'email_address',
'music_descriptor',  'movie_type'
```

علاوه بر این در هر کدام از داده های train-test-dev مشخص شده اند که از هر کدام از این برچسب ها چه قدر استفاده شده و تعداد آن ها در داده ها به چه صورت بوده است:

Intents:


```
"test": {
  "alarm_remove": 21,
  "music_settings": 6,
  "qa_stock": 26,
  "iot_cleaning": 26,
  "transport_taxi": 23,
  "email_querycontact": 26,
  "audio_volume_up": 13,
  "audio_volume_other": 6,
  "transport_query": 51,
  "general_quirky": 169,
```

```
"train": {
  "alarm_remove": 78,
  "music_settings": 51,
  "qa_stock": 152,
  "iot_cleaning": 93,
  "transport_taxi": 100,
  "email_querycontact": 127,
  "audio_volume_up": 110,
  "audio_volume_other": 18,
  "transport_query": 227,
  "general_quirky": 555,
  "datetime_query": 350,
  "iot_hue_lightchange": 125,
```

```
"dev": {
  "alarm_remove": 14,
  "music_settings": 8,
  "qa_stock": 24,
  "iot_cleaning": 19,
  "transport_taxi": 27,
  "email_querycontact": 16,
  "audio_volume_up": 12,
  "audio_volume_other": 0,
  "transport_query": 36,
  "general_quirky": 105,
```

Slots:

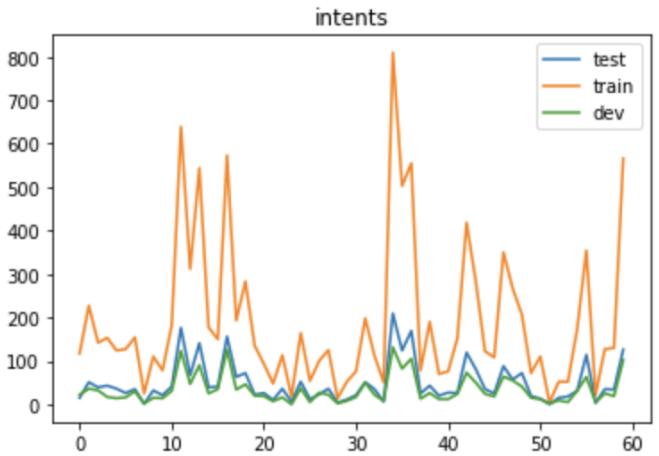

```
"test": {
  "ingredient": 6,
  "music_genre": 49,
  "order_type": 19,
  "food_type": 69,
  "personal_info": 14,
  "email_address": 9,
  "transport_type": 65,
  "alarm_type": 3,
  "movie_type": 3,
  "person": 215,
  "music_album": 1,
  "joke_type": 11,
  "color_type": 26,
  "date": 413,
```

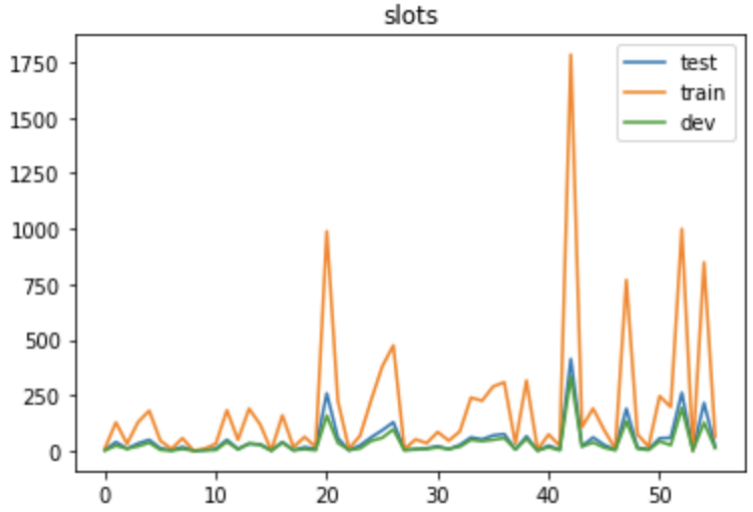
```
"train": {
  "ingredient": 18,
  "music_genre": 180,
  "order_type": 106,
  "food_type": 289,
  "personal_info": 74,
  "email_address": 31,
  "transport_type": 316,
  "alarm_type": 9,
  "movie_type": 10,
  "person": 848,
  "music_album": 1,
  "joke_type": 34,
  "color_type": 93,
  "date": 1784,
```

```
"dev": {
  "ingredient": 3,
  "music_genre": 36,
  "order_type": 19,
  "food_type": 49,
  "personal_info": 9,
  "email_address": 3,
  "transport_type": 54,
  "alarm_type": 2,
  "movie_type": 1,
  "person": 126,
  "music_album": 0,
  "joke_type": 8,
  "color_type": 14,
  "date": 334,
```


داده های کامل در کولب(فایل NLU.ipynb) موجود هستند. همچنین می توان دید که بیشترین Intent در داده ها مربوط به calendar_set بوده است. بیشترین slot نیز مربوط به date بوده است.

نمودار مربوط به داده های intent و slot در تصاویر زیر موجود است. محور افقی بیانگر شماره مربوط به intent یا slot است(شماره ها به ترتیب ذکر شده در گزارش هستند)






مشاهده می شود که تقریبا توزیع داده ها در سه partition تقریبا شبیه به هم هستند.

## آموزش مدل

در ابتدا تمام داده های فارسی به کمک دستور زیر به داده ها به دسته ی train,test,validation شکسته شده اند و فایل های مربوط به intents و slots به دست آمده اند:

```
!python massive/scripts/create_hf_dataset.py -d 1.0/data -o data
```

که در این جا فولدر دیتا تنها شامل داده های زبان فارسی است.

جهت آموزش مدل از فایل train_config.yml استفاده شده است.جزئیات مربوط به مدل استفاده شده در این فایل وجود دارد اما در این جا به طور مختصر راجع به مدل پیاده سازی شده صحبت می شود.

جهت پیاده سازی این قسمت از مدل xlm-roberta-base استفاده است و تمامی فایل های مربوط به وزن های مدل و vocab آن از hugging_face دانلود شده است. همچنین سایز batch در نظر گرفته شده برابر با ۱۲۸ بوده و optimizer استفاده شده برای یادگیری وزن ها adam بوده است. در این جا از learning_scheduler نیز استفاده شده است. این پارامتر می‌تواند روش برنامه ریزی learning rate را مشخص کند. معمولا در ماشین لرنینگ از لرنینگ ریت های بیشتر شروع می‌کنیم تا به نقطه حدودی مینیمم لاس نزدیک شویم ولی هر چه نزدیکتر شویم اگر لرنینگ ریت همان عدد اولیه باشد ممکن است مشکل over shooting داشته باشیم که به همین دلیل بهتر است کم کم لرنینگ ریت در طول آموزش کم شود.تعداد epoch ها نیز برای آموزش مدل برابر با ۴۰ در نظر گرفته

شده است. همچنین جهت تمام نشدن حافظه هنگام ذخیره checkpoint ها در هر لحظه تنها یک checkpoint ذخیره می شود و نتایج تست نیز بر روی آخرین checkpoint ذخیره شده به دست آمده است.

تنظیمات مدل roberta نیز به شکل زیر است:


```
Model config RobertaConfig {
  "architectures": [
    "XLMRIntentClassSlotFill"
  ],
  "attention_probs_dropout_prob": 0.0,
  "bos_token_id": 0,
  "classifier_dropout": null,
  "eos_token_id": 2,
  "head_intent_pooling": "max",
  "head_layer_dim": 2048,
  "head_num_layers": 1,
  "hidden_act": "gelu",
  "hidden_dropout_prob": 0.45,
  "hidden_layer_for_class": 11,
  "hidden_size": 768,
  "initializer_range": 0.02,
  "intermediate_size": 3072,
  "layer_norm_eps": 1e-05,
  "max_position_embeddings": 514,
  "model_type": "roberta",
  "num_attention_heads": 12,
  "num_hidden_layers": 12,
  "output_past": true,
  "pad_token_id": 1,
  "position_embedding_type": "absolute",
  "slot_loss_coef": 4.0,
  "torch_dtype": "float32",
  "transformers_version": "4.20.1",
  "type_vocab_size": 1,
  "use_cache": true,
  "use_crf": false,
  "vocab_size": 250002
}
```

نتایج بر روی داده های آموزش و پس از 40 ایپاک به صورت زیر است:

'training_epoch': 40.0,

'eval_fa-IR_loss': 1.878304362297058,

'eval_fa-IR_intent_acc': 0.8484997540580423,

'eval_fa-IR_intent_acc_stderr': 0.007951770551569794,

'eval_fa-IR_slot_micro_f1': 0.7365283191928456,

'eval_fa-IR_slot_micro_f1_stderr': 0.0015447670985724244

## ارزیابی مدل

برای ارزیابی از مدل آموزش داده شده طی 40 ایپاک استفاده شده است و سپس بر روی داده های تست اعمال شده است. فایل مربوط به پیش بینی های انجام شده در آدرس NLU/preds.jsonl موجود است. همچنین تمامی کانفیک های لازم برای تست نیز در test_config.yml نوشته شده است. در این فایل آدرس مدل،داده های مربوط به intent و slot ها، محل ذخیره پیش بینی انجام شده و یا اطلاعاتی مانند سایز padding داده شده است.

یک نمونه از خروجی ایجاد شده به صورت زیر است:

{"id": "7730", "locale": "fa-IR", "utt": ["بزن", "علامت", "را", "بعدی", "دوشنبه"],
"pred_intent": "calendar_set", "pred_slots": [["دوشنبه", "date"]],
"pred_annot_utt": "[date : دوشنبه]بزن علامت را بعدی"}

ارزیابی بر روی داده های تست نیز منجر به نتایج زیر شد:

{ 'test_all_ex_match_acc': 0.6106254203093476,

'test_all_ex_match_acc_stderr': 0.008941301829524101,

'test_all_intent_acc': 0.8473436449226631,

'test_all_intent_acc_stderr': 0.0065950296218791156,

'test_all_loss': 1.9632301330566406,

'test_all_runtime': 6.2451,

'test_all_samples_per_second': 476.214,

'test_all_slot_micro_f1': 0.7216965742251223,

'test_all_slot_micro_f1_stderr': 0.0009685005411938702,

'test_all_steps_per_second': 3.843,

'training_epoch': None,

'training_global_step': 0}

مشاهده می شود که دقت بر روی intent به ۸۴ درصد رسیده است و مقدار f1 برا اسلات ها حدودا ۷۲ است. این اعداد به مقادیر به دست آمده از مقاله نیز نزدیک هستند (در مقاله برای زبان فارسی دقت intent حدود ۸۶ و مقدار f1 برای اسلات ها حدود ۷۶ به دست آمده است)که نشان می دهد آموزش مدل به خوبی انجام شده است و نتایج قابل قبول هستند.