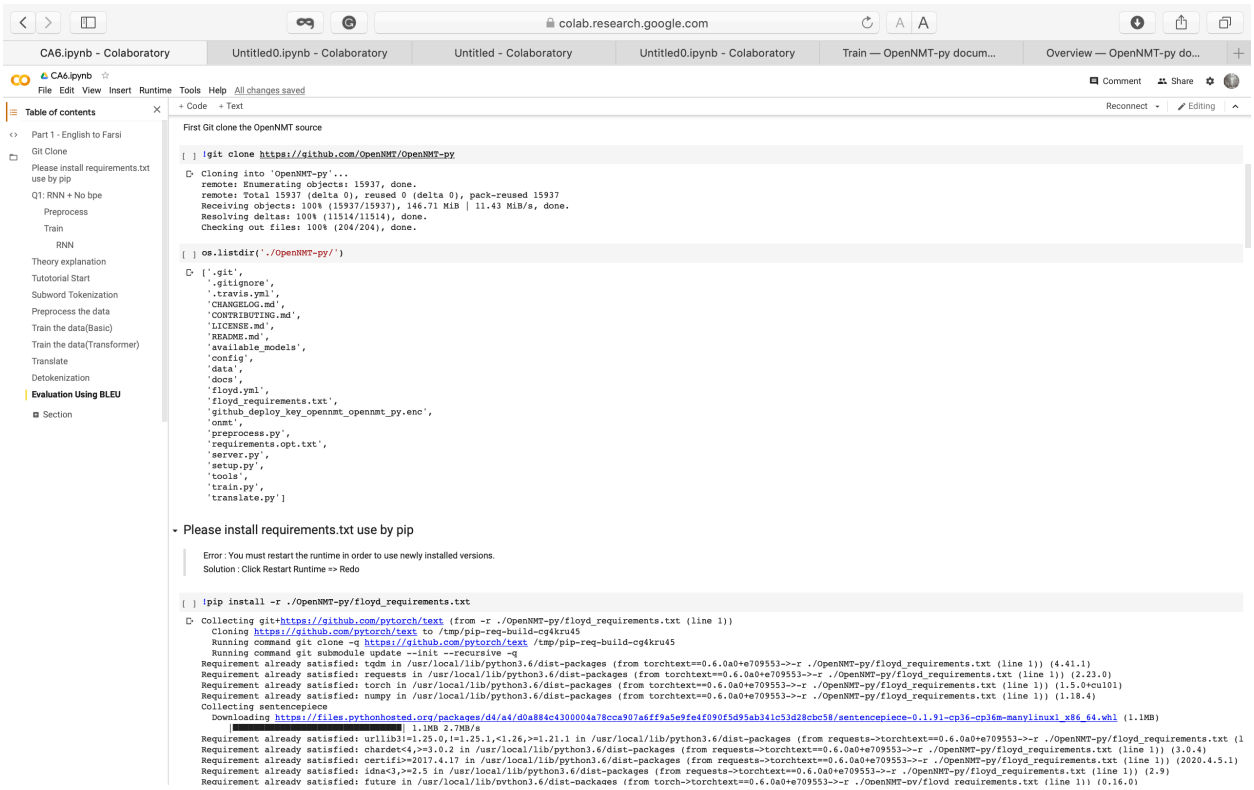
آموزش سیستم ترجمه انگلیسی به فارسی

بخش ۱

ابتدا برای اینکه بتوانم از فایل Clone شده استفاده کنم با توجه به راهنمایی‌های [1] به پوشه‌ای در drive خودم cd کردم و سپس مراحل نصب OpenNMT و نیازمندی‌های آن را انجام دادم (شکل ۱.۱ این گام‌ها را نمایش می‌دهد).



شکل ۱.۱: نصب کتابخانه‌ی مورد نظر

برای انجام پیش‌پردازش از دستور زیر استفاده می‌کنیم:

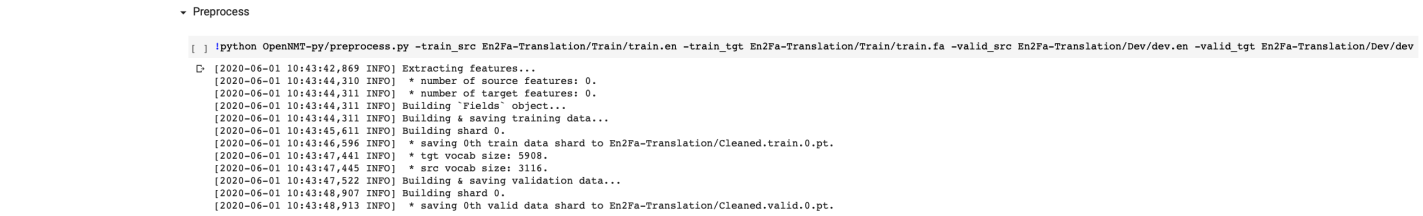
```
!python OpenNMT-py/preprocess.py -train_src En2Fa-Translation/
Train/train.en -train_tgt En2Fa-Translation/Train/train.fa -
valid_src En2Fa-Translation/Dev/dev.en -valid_tgt En2Fa-
Translation/Dev/dev.fa -save_data En2Fa-Translation/Cleaned
```

پارامترهای این دستور به این شکل است که برای داده‌های آموزش و validation آدرس فایل‌های مبدا و مقصد (چیزی که باید ترجمه شود و ترجمه‌ی آن) را دریافت کرده و محل ذخیره و پیش واژه‌ی مورد استفاده برای ذخیره‌ی آن‌ها را نیز دریافت می‌کند. در فایل‌های ورودی انتظار دارد که هر جمله در یک خط مجزا باشد که جملات ما به این شکل هستند. خروجی‌های تولید شده فایل‌های به شکل زیر هستند:

```
'Cleaned',
'Cleaned.train.0.pt',
```

```
'Cleaned.vocab.pt',
'Cleaned.valid.0.pt'
```

نتیجه‌ی اجرای دستور بالا نیز در شکل ۱.۲ نمایش داده شده است.



شکل ۱.۲: پیش‌پردازش

حال به کمک دستور زیر مدل را آموزش می‌دهیم. پارامترها به این شکل است که نوع RNN مورد استفاده را LSTM قرار می‌دهیم (علت این انتخاب این است که خواسته شده است مدل بر مبنای RNN باشد و طبق درس LSTM نیز نوعی RNN است) و تعداد لایه‌های مورد استفاده را برابر با ۴ قرار می‌دهیم. تعداد مراحل آموزش نیز طبق صورت سوال ۵۰۰۰۰ است. تعداد نرون‌های لایه‌ی مخفی را نیز برابر با ۵۱۲ قرار می‌دهیم. برای ذخیره‌ی مدل‌های مختلف در حین آموزش نیز به صورت پیش فرض هر ۵۰۰۰ ایپاک ذخیره سازی انجام می‌شود که این پیش‌فرض را فعلا تغییری نمی‌دهیم. نتیجه در شکل ۱.۳ نمایش داده شده است.

```
!python OpenNMT-py/train.py -data En2Fa-Translation/Cleaned -
save_model En2Fa-Translation/rnn_model -world_size 1 -gpu_rank 0
--rnn_size 512 --layers 4 --rnn_type LSTM -train_steps 50000
```

```
[2020-06-01 14:18:51,709 INFO] Step 49050/50000; acc:  92.99; ppl:  1.26; xent: 0.23; lr: 1.00000; 5135/5179 tok/s;   6282 sec
[2020-06-01 14:18:55,868 INFO] Loading dataset from En2Fa-Translation/Cleaned.train.0.pt
[2020-06-01 14:18:56,206 INFO] number of examples: 26142
[2020-06-01 14:18:58,914 INFO] Step 49100/50000; acc:  92.64; ppl:  1.27; xent: 0.24; lr: 1.00000; 4767/4773 tok/s;   6289 sec
[2020-06-01 14:19:05,197 INFO] Step 49150/50000; acc:  93.75; ppl:  1.22; xent: 0.20; lr: 1.00000; 5284/5195 tok/s;   6296 sec
[2020-06-01 14:19:12,146 INFO] Step 49200/50000; acc:  91.98; ppl:  1.31; xent: 0.27; lr: 1.00000; 5036/5019 tok/s;   6303 sec
[2020-06-01 14:19:18,070 INFO] Step 49250/50000; acc:  93.07; ppl:  1.25; xent: 0.23; lr: 1.00000; 4817/5230 tok/s;   6309 sec
[2020-06-01 14:19:23,990 INFO] Step 49300/50000; acc:  93.26; ppl:  1.25; xent: 0.22; lr: 1.00000; 5114/5110 tok/s;   6315 sec
[2020-06-01 14:19:29,970 INFO] Step 49350/50000; acc:  93.52; ppl:  1.24; xent: 0.22; lr: 1.00000; 5125/5216 tok/s;   6320 sec
[2020-06-01 14:19:36,206 INFO] Step 49400/50000; acc:  93.57; ppl:  1.23; xent: 0.21; lr: 1.00000; 5076/5378 tok/s;   6327 sec
[2020-06-01 14:19:42,781 INFO] Step 49450/50000; acc:  92.75; ppl:  1.27; xent: 0.24; lr: 1.00000; 5397/5362 tok/s;   6333 sec
[2020-06-01 14:19:48,122 INFO] Loading dataset from En2Fa-Translation/Cleaned.train.0.pt
[2020-06-01 14:19:48,452 INFO] number of examples: 26142
[2020-06-01 14:19:49,826 INFO] Step 49500/50000; acc:  92.80; ppl:  1.27; xent: 0.24; lr: 1.00000; 4609/4675 tok/s;   6340 sec
[2020-06-01 14:19:56,145 INFO] Step 49550/50000; acc:  93.31; ppl:  1.24; xent: 0.22; lr: 1.00000; 5152/5103 tok/s;   6347 sec
[2020-06-01 14:20:03,362 INFO] Step 49600/50000; acc:  91.99; ppl:  1.31; xent: 0.27; lr: 1.00000; 5213/5036 tok/s;   6354 sec
[2020-06-01 14:20:09,344 INFO] Step 49650/50000; acc:  92.98; ppl:  1.26; xent: 0.23; lr: 1.00000; 4860/5306 tok/s;   6360 sec
[2020-06-01 14:20:15,143 INFO] Step 49700/50000; acc:  92.85; ppl:  1.27; xent: 0.24; lr: 1.00000; 5006/5055 tok/s;   6366 sec
[2020-06-01 14:20:21,236 INFO] Step 49750/50000; acc:  93.43; ppl:  1.24; xent: 0.22; lr: 1.00000; 5189/5226 tok/s;   6372 sec
[2020-06-01 14:20:27,087 INFO] Step 49800/50000; acc:  93.89; ppl:  1.22; xent: 0.20; lr: 1.00000; 5053/5410 tok/s;   6378 sec
[2020-06-01 14:20:34,206 INFO] Step 49850/50000; acc:  92.73; ppl:  1.26; xent: 0.23; lr: 1.00000; 5394/5354 tok/s;   6385 sec
[2020-06-01 14:20:40,318 INFO] Loading dataset from En2Fa-Translation/Cleaned.train.0.pt
[2020-06-01 14:20:40,661 INFO] number of examples: 26142
[2020-06-01 14:20:40,956 INFO] Step 49900/50000; acc:  92.72; ppl:  1.27; xent: 0.24; lr: 1.00000; 4554/4646 tok/s;   6391 sec
[2020-06-01 14:20:47,330 INFO] Step 49950/50000; acc:  93.09; ppl:  1.25; xent: 0.22; lr: 1.00000; 5187/5087 tok/s;   6398 sec
[2020-06-01 14:20:54,086 INFO] Step 50000/50000; acc:  92.69; ppl:  1.28; xent: 0.24; lr: 0.50000; 5111/5100 tok/s;   6405 sec
[2020-06-01 14:20:54,086 INFO] Loading dataset from En2Fa-Translation/Cleaned.valid.0.pt
[2020-06-01 14:20:54,091 INFO] number of examples: 276
[2020-06-01 14:20:54,646 INFO] Validation perplexity: 484.835
[2020-06-01 14:20:54,646 INFO] Validation accuracy: 44.0111
[2020-06-01 14:20:54,679 INFO] Saving checkpoint En2Fa-Translation/rnn_model_step_50000.pt
```

شکل ۱.۳: چند مرحله از آموزش مدل RNN

الف

همانطور که بالاتر نشان دادیم مدل آموزش داده شده است. حال با استفاده از این مدل برای داده‌های تست پیش‌بینی را انجام می‌دهیم و سپس معیار BLEU را برای آن محاسبه می‌کنیم. دو دستور زیر به ترتیب کارهای گفته شده را انجام می‌دهند.

```
!python OpenNMT-py/translate.py -model En2Fa-Translation/
rnn_model_step_50000.pt -src En2Fa-Translation/Test/test.en -
output En2Fa-Translation/pred.txt -replace_unk -verbose
```

```
!perl  OpenNMT-py/tools/multi-bleu.perl En2Fa-Translation/Test/
test.fa0 En2Fa-Translation/Test/test.fa1 En2Fa-Translation/Test/
test.fa2 En2Fa-Translation/Test/test.fa3 < En2Fa-Translation/
pred.txt
```

خروجی این بخش در شکل ۱.۴ نمایش داده شده است و مقدار محاسبه شده برابر با ۲۰.۲۷ است.

```
[14] !perl  OpenNMT-py/tools/multi-bleu.perl En2Fa-Translation/Test/test.fa0 En2Fa-Translation/Test/test.fa1 En2Fa-Translation/Test/t

  BLEU = 20.27, 60.0/26.9/15.1/8.7 (BP=0.944, ratio=0.946, hyp_len=2412, ref_len=2551)
```

شکل ۱.۴: مقدار BLEU روی داده‌های تست با ۴ مرجع

ب

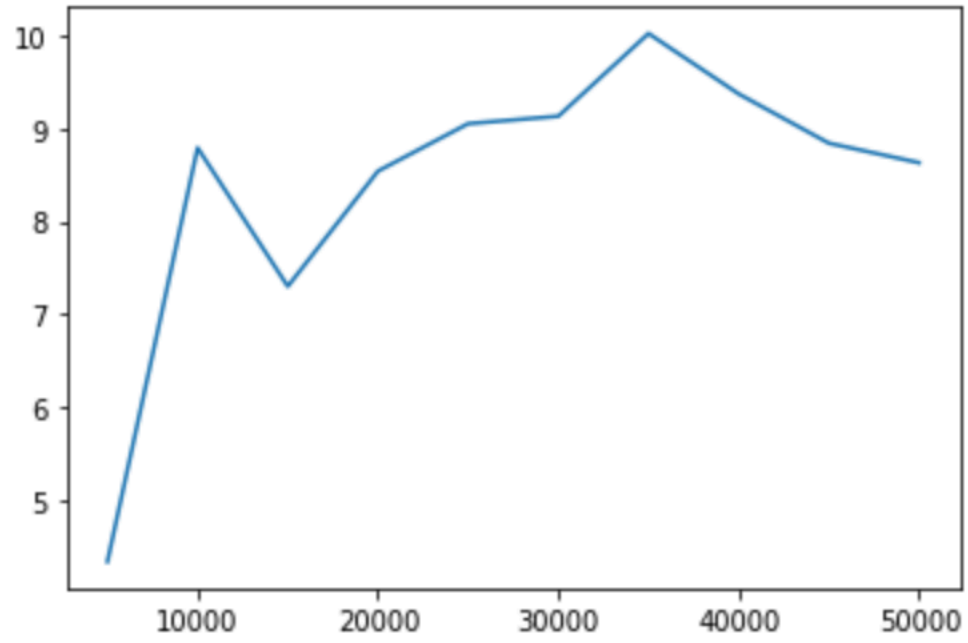
برای این بخش با استفاده از کد نشان داده شده در شکل ۱.۵ مقدار BLEU را برای هر یک از checkpointها که از هم فاصله‌ی ۵۰۰۰ تایی دارند را روی داده‌های dev محاسبه می‌کنیم. شکل ۱.۶ تغییرات این معیار را نمایش می‌دهد.

```
[19] values = []
     iter_num = 5000
     while iter_num <= 50000:
       command = "python OpenNMT-py/translate.py -model En2Fa-Translation/rnn_model_step_{}.pt -src En2Fa-Translation/Dev/dev.en -output En2Fa
       _ = !eval $command
       out = !eval "perl OpenNMT-py/tools/multi-bleu.perl En2Fa-Translation/Dev/dev.fa < En2Fa-Translation/Dev/pred.txt"
       values.append(out)
       iter_num += 5000
```

شکل ۱.۵: محاسبه‌ی مقدار معیار BLEU در checkpoint های مختلف

با توجه به نمودار ۱.۶ این طور به نظر می‌رسد که حدود گام ۳۵۰۰۰ام بهترین مدل را داشته‌ایم و پس از آن مدل شروع به overfit شدن کرده است و عدد مطلوب برای آموزش مدل ۳۵۰۰۰ گام بوده است. البته این با توجه به پارامترهای داده شده به مدل و تعداد لایه و نرون و ... نشان داده شده در بالا است و شاید اگر این پارامترها را بهتر یا متفاوت انتخاب می‌کردیم، نتیجه‌ی دیگری می‌گرفتیم.

```
[<matplotlib.lines.Line2D at 0x7f6375808eb8>]
```



شکل ۱.۶: تغییر معیار BLEU روی داده‌های dev

پ

یکی از جملات ترجمه شده به اشتباه جمله‌ی زیر است:

```
['the', 'plane', 'arrives', 'at', 'Hanover', 'at', 'a',
'quarter', 'past', 'eleven', '.']
```

جمله‌ی تولید شده توسط سیستم ما برای این جمله عبارت است از:

- آن در حقیقت ساعت یازده و ربع طول میکشد

و ترجمه‌ی مرجع به شکل زیر است:

علت این خطا می‌تواند این باشد که مدل با کلمه‌ی خاصی مثل Hanover آشنا نیست و این مسئله سبب می‌شود که نتواند آن را ترجمه کند. از طرفی ممکن است کلمه‌ی plane را خیلی در داده‌های آموزش ندیده باشد و در نتیجه ترجمه‌ی آن را یاد نگرفته باشد. مشخص است که در ترجمه‌ی ساعت‌ها مشکلی ندارد ولی این احتمال وجود دارد که داده‌های آموزش به اندازه‌ی کافی نمونه برای آموزش مدل در زمینه‌ی خاص پرواز هواپیما نداشته بوده باشد. به علاوه به نظر می‌رسد مدل ترتیب کلمات را هم نتوانسته است به درستی تغییر درد به صورتی که بتواند جمله‌ی مناسبی تولید کند.

یکی دیگر از جملات ترجمه شده به نادرست، عبارت زیر است:

```
['then', 'everything', 'is', 'arranged', '.',
'goodbye', '.']
```

ترجمه‌ی مدل در این نمونه به شرح زیر است:

```
. پس به این دلیل هم همینطور . خداحافظ
```

و ترجمه‌ی مرجع:

پس همه چیز مرتب شده است . خداحافظ .

علت این خطا می‌تواند راجع نبودن این فرم جمله در داده‌های آموزش باشد.

افزایش تعداد داده‌ها و اطمینان از representative بودن آن‌ها می‌تواند یک راه حل برای این مسئله باشد. یک راه کار دیگر معیاری است که به ترتیب کلمات و به عبارت دیگر روانی جمله اهمیت بیشتری بدهد تا جملات روان‌تری تولید شوند. هم چنین آموزش برای زمان طولانی تر نیز می‌تواند مفید باشد چون مقدار پیش‌فرض مدل برای تعداد دفعات آموزش ۱۰۰۰۰۰ است که از تعداد مورد استفاده‌ی ما خیلی بیشتر است.

ت

با توجه به [2] این پارامتر برای رسیدگی به توکن‌های unknown است. به این شکل که در زمان ترجمه برای توکن‌هایی که unknown هستند و مدل آن‌ها را نمی‌شناسد سیاست اجرایی را مشخص می‌کند. اگر این پارامتر را استفاده کنیم مدل تلاش می‌کند این توکن را با توکنی که بالاترین وزن attention را دارد جایگزین کند. اگر از آن استفاده نکنیم مدل توکن‌های ناآشنا را تنها در خروجی تکرار می‌کند و تلاش نمی‌کند نزدیک‌ترین کلمه‌ای که می‌شناسد را بیاورد.

ث

از بین پارامترهای معرفی شده در لینک به نظر من پارامترهای زیر می‌تواند از سایر پارامترها تاثیر بیشتری داشته باشد:
۱) ترکیب تعداد لایه‌های موجود و تعداد نرون‌های موجود در هر لایه: این تعداد روی میزان اطلاعاتی که مدل موفق می‌شود از داده‌ها استخراج کند و یادبگیرد تاثیر خواهد داشت چون تعداد اطلاعاتی که می‌تواند کد کند متفاوت خواهد بود.
۲) تابع بهینه‌سازی استفاده شده: این تابع روی چگونگی بهینه کردن و در نتیجه روی مسیر رسیدن به و تعریف ما از جواب مطلوب تاثیر خواهد داشت.
۳) نرخ یادگیری: نرخ یادگیری میزان تاثیر هر داده‌ی جدید روی میزان دانش مدل و میزان تغییر اطلاعات قبلی را مشخص می‌کند که اگر خیلی زیاد باشد مدل سریع دانش قبلی خود را با اطلاعات جدید جایگزین خواهد کرد و اگر خیلی کم باشد خیلی طول خواهد کشید تا مدل اطلاعات جدید را در خود کد کند.
۴) نوع لایه‌هایی که استفاده می‌کنیم: نوع RNNهای مورد استفاده و یا نوع لایه‌های استفاده شده در encoder و decoder همگی قابل تنظیم هستند که هر لایه نکات مثبت خود را می‌تواند داشته باشد و انتخاب ما می‌تواند تاثیر بسزایی در نتیجه‌ی نهایی داشته باشد.
۵) میزان کوچک شدن نرخ یادگیری در حین آموزش: یا به عبارت دیگر learning rate decay مشخص می‌کند که هر اپاک نرخ یادگیری چقدر کوچک شود که این مقدار روی سرعت یادگیری و سرعت کاهش یادگیری تاثیر خواهد داشت که مدل را تحت تاثیر قرار می‌دهد.

ج

در این بخش به مقایسه‌ی چند عامل مختلف می‌پردازیم. مورد (۴) یعنی نوع لایه‌ها را تغییر نمی‌دهیم (به عنوان مثال به CNN یا سایر مدل‌های موجود) چون در صورت سوال خواسته شده است که مدل مبتنی بر RNN باشد. اما به بررسی چند پارامتر دیگر می‌پردازیم. به علاوه به دلیل زیاد بودن تعداد مدل‌ها و طولانی بودن مدت زمان اجرا هر یک را برای ۱۰۰۰۰ step (و نه ۵۰۰۰۰ تای قبلی) آموزش داده و سپس مقایسه می‌کنیم. در هر بخش سایر پارامترها یکسان بوده و تنها همان یک پارامتر بیان شده را تغییر می‌دهیم.

* بررسی تاثیر rnn_size-

این مورد مربوط به مورد (۱) بیان شده در بخش قبل می‌باشد که تعداد نرون‌های موجود در لایه‌های میانی را مشخص می‌کند.

RNN Size	Accuracy	Duration
1024	79.27	3384.54374527931
512	69.38	1322.9168176651

می‌توان دید که با بیشتر شدن این پارامتر قدرت مدل بیشتر می‌شود اما از طرفی تعداد پارامترهایی که باید آموزش داده شوند نیز بیشتر می‌شود در نتیجه سرعت یادگیری کاهش پیدا می‌کند.

* بررسی تاثیر learning_rate-

این مورد مربوط به مورد (۳) بیان شده در بخش قبل می‌باشد.

Learning Rate	Accuracy	Duration
1	69.38	1322.9168176651
0.1	43.37	1328.03911423683

می‌توان دید که با کاهش نرخ یادگیری در تعداد ایپاک برابر مدل اطلاعات کمتری را یادمی‌گیرد. البته باید به این نکته نیز توجه کردن که برای توابع بهینه‌ساز مختلف مقدار نرخ یادگیری متفاوتی پیشنهاد شده است. زمان یادگیری تغییری نمی‌کند چون محاسبات و تعداد پارامترها تغییری نکرده است و تنها ضریب در برخی از روابط فرمول متفاوتی دارد.

* بررسی تاثیر optim-

این مورد مربوط به مورد (۲) بیان شده در بخش قبل می‌باشد.

Optim	Accuracy	Duration
sgd	69.38	1322.9168176651
Adam	2.63	1442.04293203354

می‌توان دید که با ثابت بودن سایر پارامترها تابع بهینه‌سازی sgd به طور چشمگیری بهتر عمل می‌کند.

بخش ۲

برای این بخش عمده‌ی دستورات مشابه حالت قبل است با این تفاوت که دو گام به مراحل ما اضافه می‌شوند. گام اول یادگیری و سپس اعمال bpe است که این گام قبل از سایر گام‌ها (قبل از پیش‌پردازش) انجام می‌شود و گام دیگر decode کردن آن پس از انجام ترجمه است که قبل از محاسبه‌ی BLEU انجام خواهد شد. دستورات این بخش‌ها به شرح زیر است.

ابتدا با دو دستور زیر برای ورودی و خروجی مدل bpe را یادمی‌گیریم.

```
!python OpenNMT-py/tools/learn_bpe.py -i En2Fa-Translation/
Train/train.fa -o En2Fa-Translation/Train/BPEtrain.fa -s 10000


!python OpenNMT-py/tools/learn_bpe.py -i En2Fa-Translation/
Train/train.en -o En2Fa-Translation/Train/BPEtrain.en -s 10000
```

سپس با تکرار دستور زیر برای مبدا و مقصدهای Train و Dev و مبدا Test تبدیل را انجام می‌دهیم.

```
!python OpenNMT-py/tools/apply_bpe.py -c En2Fa-Translation/
Train/BPEtrain.fa -i En2Fa-Translation/Train/train.fa -o En2Fa-
Translation/Train/Q2train.fa
```

در نهایت نیز پس از پیش‌پردازش، آموزش مدل و انجام ترجمه مرحله‌ی زیر را انجام می‌دهیم.

```
!sed -i "s/@@ //g"  En2Fa-Translation/BPEpred.txt
```

الف

تغییرات به شکل توضیح داده شده در بالا خواهند بود. نقشی که این کار روی آموزش دارد این است که این امکان را می‌دهد که به جای یادگیری فقط در سطح کلمه و کد کردن و پیدا کردن representation برای کلمه‌ها بتوانیم برای subwordها نیز این کار را انجام دهیم که این کار این امکان را به ما می‌دهد که زمانی که کلمه‌ای را نمی‌شناسیم بتوانیم آن را به بخش‌های کوچکتر قابل شناسایی که یک روش نمایش برای آن‌ها دهیم بشکنیم و کلمه را با مجموعه‌ای از این بخش‌ها نمایش دهیم.

ب

با انجام این کار دقت مدل از ۹۲.۶۷ در حالت قبل به ۹۱.۵۶ تغییر می‌کند. ولی معیار BLEU در این حالت ۲۳.۴۳ خواهد بود در حالی که در حالت قبلی ۲۰.۲۷ بود. در نتیجه به نظر می‌رسد که عملکرد مدل بهبود پیدا کرده است.

ج

یک حالت که مدل بهبود عملکرد داشته است نمونه‌ی زیر است: (جمله‌ی بالا برای حالت بدون bpe است)

```
SENT 245: ['unfortunately', 'I', 'did', 'not',
'understand', 'that', '.']
PRED 245: متاسفانه من آن را مخالفتی ندارم .

SENT 245: ['unfortunately', 'I', 'did', 'not',
'understand', 'that', '.']
PRED 245: متاسفانه من متوجه نشدم .
```

علت این بهبود می‌تواند این باشد که مدل کلمه‌ی understand را خیلی در مثال‌ها ندیده‌ایم در و برای همین نمی‌توانیم آن را ترجمه کنیم ولی زمانی که این کلمه را به بخش‌های کوچکتر می‌شکنیم و اجازه‌ی استفاده از bpe را به سیستم می‌دهیم دقت بهتر می‌شود.

زمانی دقت بدتر می‌شود که مدل تلاش می‌کند اسامی خاص مانند نام هتل یا نام مکان‌ها را به بخش‌های کوچکتر شکسته و ترجمه کند چون این کلمات الزاما معنی خاصی ندارد و ما دوست داریم همان‌ها در ترجمه تکرار شوند.
به عنوان مثال شکستن نام هتل به شکل زیر مطلوب نیست:

```
'I', 'have', 'already', 'booked', 'two', 'rooms', 'at',
'the', 'G@@', 'r@@', '"@@', 'un@@', 'sch@@', 'na@@',
'be@@',
```

آموزش سیستم نویسه گردانی فارسی به انگلیسی

به طور کلی مراحل آموزش مدل مشابه بخش ۱ در حالت قبل است. یک تفاوت در این سیستم این است که در این سیستم قبل از آموزش باید آماده سازی بیان شده را انجام دهیم که در شکل ۲.۱ این کار برای یکی از فایل‌ها نشان داده شده است. این کار را برای داده‌های train و dev نیز تکرار می‌کنیم.

```
[9]  fixed_values = []
     with open('Transliteration/test.fa', 'r') as fp:
       content = fp.read().split('\n')
       for this_sent in content:
         this_sent_fixed = []
         for this_char in this_sent:
           if this_char == ' ':
             this_sent_fixed.append('<b>')
           else:
             this_sent_fixed.append(this_char)
           this_sent_fixed.append(' ')
         this_sent_joined = ' '.join(this_sent_fixed)
         this_sent_joined = this_sent_joined.strip()
         fixed_values.append(this_sent_joined)

     with open('Transliteration/fixed_test.fa', 'w') as fp:
       for v in fixed_values:
         fp.write(v+'\n')
```

شکل ۲.۱: شکستن جمله به حروف جدا از هم

الف

دقت مدل روی داده های آموزش پس از تمامی مراحل برابر با ۹۷.۹۵ است. معیار BLEU نیز برابر است با: ۲۶.۳۸

ب

این معیار برای تسک نویسه‌گردانی مناسب نیست. علت این است که با اینکه یکی از ایده‌های اصلی پشت این معیار این است که هر چه ترجمه به ترجمه‌ی انسان نزدیکتر باشد بهتر است اما در محاسبه‌ی P موجود در فرمول تعداد را می‌شماریم که اجازه‌ی جابجایی به کلمات بدهیم و مدل را مجبور نکنیم که حتما کلمات را به ترتیب مشخصی قرار دهد چون ترجمه می‌تواند تا حد خوبی سلیقه‌ای باشد. اما در این تسک ترتیب کلمات تا حد خوبی مشخص است و قرار نیست ترتیب کلمات را جابجا کنیم ولی این آزادی عملی که این معیار به ما می‌دهد می‌تواند باعث شود که فکر کنم مدل بهتر از چیزی است که واقعا هست.

ج

از آنجا که در این تسک کار ترجمه را تقریبا به صورت حرف به حرف انجام می‌دهیم یک معیار برای مقایسه می‌تواند edit distance باشد که در آن تعداد کاراکترهایی که نیاز است تغییر کنند تا به جمله‌ی مطلوب برسیم را محاسبه کنیم. برای پیاده سازی از کتابخانه‌ی editdistance==0.3.1 کمک می‌گیریم. در این صورت کد به شکل نشان داده شده در تصویر ۲.۲ خواهد بود و متوسط این معیار برای تمامی جملات تست برابر با ۳۱.۰۳۴ خواهد بود، این درحالی است که جملات زیادی وجود دارند که ۰ تغییر نیازدارند و میانه‌ی تعداد تغییرات برابر با

۱۶.۰ است. به علاوه باید دقت شود که اعداد در سطح کاراکتر هستند و نه کلمه. به نظر من این معیار، معیار بهتری است چون در اینجا قرار نیست به ترتیب‌های متفاوت کلمات امتیازی داده شود و در نتیجه انتظار داریم که دقیقا کاراکترها به همان ترتیب مورد نظر قرار بگیرند و بجز در موارد خاص چند حالت مختلف برای معادل یک کلمه وجود ندارد. به همین دلیل چک کردن یکی بودن کاراکترها معیار مناسبی است و دقیقا به ما نشان می‌دهد که چقدر اشتباه کرده‌ایم.

```
[22] import editdistance
```

```
[24] goal_sentences, predicted_sentences = [], []
     distance_values = []
     with open('Transliteration/test.en', 'r') as fp:
       goal_sentences = fp.read().split('\n')
     with open('Transliteration/Transliteration_pred.txt', 'r') as fp:
       predicted_sentences = fp.read().split('\n')

     for sent_id in range(len(goal_sentences)):
       s1 = goal_sentences[sent_id]
       s2 = predicted_sentences[sent_id]
       d = editdistance.eval(s1, s2)
       distance_values.append(d)
```

شکل ۲.۲: چگونگی محاسبه‌ی معیار edit distance

د

خیر این روش مفید نخواهد بود. علت این مسئله این است که در اینجا داریم به تک تک حروف نگاه می‌کنیم و در نتیجه امکان اینکه حرفی در دادّه‌ای آموزش دیده نشده باشد و در نتیجه اطلاعات یا representation ای از آن نداشته باشیم کم است و در داده‌های تست ممکن نیست به حالتی بر بخوریم که نتوانیم آن را کد کنیم. به علاوه بخش کوچکتری برای شکستن حرف و آن و محاسبه‌ی اطلاعات در آن سطح وجود ندارد در نتیجه این روش مفید نخواهد بود.

مراجع

[1] https://stackoverflow.com/questions/51234981/cant-view-github-repos-after-cloning-them-into-colaboratory

[2] https://opennmt.net/OpenNMT-py/options/translate.html?highlight=replace_unk