

# Assignment 5



NLP

## PREPARED BY

**Samin Mehdizadeh - 810100526**

**Mohsen Fayyaz - 810100524**

Spring 2022

## فهرست مطالب

4

### 1 ترجمه ماشینی

1) همانطور که می دانیم در تسک NMT نیازمند پیش پردازش هستیم، یکی از روش های پیش پردازش اعمال bpe بر پیکره آموزشی است به نظر شما چه پیش پردازش دیگری برای ترجمه ماشینی مبتنی بر شبکه عصبی مفید است؟ به دلخواه خود دو روش پیش پردازش دیگر را، با ذکر دلیل، انتخاب کرده و نام ببرید.

5

2) به نظر شما در زبان فارسی کدام یک از این 3 نوع پیش پردازش از اهمیت بیشتری برخوردار است؟ چرا؟ در زبان انگلیسی چطور؟

5

3) همانطور که می دانید معماری های متفاوتی برای آموزش مدل های sequence to sequence وجود دارد. در این تمرین قصد داریم تا مدلی بر مبنای معماری transformer آموزش دهیم.

7

4) در این مرحله متناسب با ابزارهایی که انتخاب کرده اید، 10 تا از پارامترهای آموزش مدل را شرح دهید (در هر ابزار به شکل جداگانه). ابزار مورد نظر خود را نام برد و ابتدا توضیح دهید که این پارامترها چه کاری انجام می دهند و سپس توضیح دهید چه پارامترهایی را لازم است تا نسبت به حالت پیش فرض تغییر دهید.

8

5) پارامتری که فکر می کنید در کیفیت خروجی یک مدل ترجمه ماشینی تاثیر گذار هستند ولی امکان تغییر یا تنظیم آنها با توجه به منابع موجود (سخت افزاری و محدودیت های ابزار) وجود ندارد را نام بده و اهمیت هر یک را مختصرا توضیح دهید. (در هر ابزار به شکل جداگانه)

6) حال می خواهیم 2 مدل مختلف را با استفاده از 2 ابزار انتخابی آموزش دهیم که جزیات این بخش به شرح زیر است:

15

ابتدا bpe و دو روش پیش پردازش دیگر را بر روی دادگان آموزش اعمال کنید

15

چگونگی و جزئیات پیاده سازی این بخش را در گزارش خود ذکر کنید.

15

انتخاب ابزار مناسب در این مرحله بر عهده شماست و محدودیتی وجود ندارد.

15

خروجی مرحله پیش پردازش را به شکل یک فایل جداگانه در گزارش خود بیاورید.

18

پارامترهای لازم را متناسب با دادگان، منابع موجود و صلاحیت خود مقداردهی کنید.

18

پارامترهای آموزش مدل را به شکل کامل در گزارش خود ذکر کنید.

21

آموزش مدل را تا زمانی که در میانه آموزش قرارگیرد ادامه دهید.

21

توصیه میشتد تا از google colab برای حل این تمرین استفاده

در این تمرین قصد داریم با روند آموزش یک مدل ماشین ترجمه و الامات آن آشنا شویم . انتظار ما این است که مدل نهایی که ارائه می دهدی در میانه مسیر آموزش باشد. یکی از راه های بررسی این موضوع کنترل دستی خروجی مدل بر روی داده های تست است برای مثال معمولاً خروجی یک مدل تازه شروع به آموزش کرده است، تکرار تنهای چند کلمه خاص است و بدینهی است که این مدل به عنوان مدل نهایی پذیرفته نیست.

21

با توجه به کم بودن حجم مجموعه داده اولیه و محدودیتهای منابع انتظار تولید یک ماشین با کیفیت را نداریم. نگران نتایج ضعیف احتمالی نباشید:

21

انتظار نداریم که مجموع زمان آموزش در هر ابزار بیش از ۶ ساعت به طول انجامد و الزاماً صرف زمان بیشتر برای آموزش مدل امتیاز محسوب نمی شود.

لطفاً علاوه بر فایل گزارش، فایل اسکریپت دستورات اجرا شده و یا اگر در Google Colab اجرا کرده اید فایل notebook آن به همراه خروجی سیستم های آموزش داده شده برای فایل های تست را نیز ارسال کنید.

21

7) پس از آموزش بایستی روند تغییرات Bleu دو مدل آموزش دیده را بر روی مجموعه dev با افزایش تعداد epochها نشان دهید. برای این کار میتوانید از دستور ذخیره مدل های میانی در ابزار مورد نظر خود استفاده کنید. (حداقل 5 نقطه را در طول آموزش مدل گزارش دهید)

22

8) با کمک دادگان موازی test، مقدار Bleu را برای این دو مدل بر روی دادگان آزمایشی گزارش دهید.

24

9) 3 تا از معیارهایی که فکر میکنید برای ارزیابی یک ابزار تولید ماشین ترجمه مناسب هستند را نام ببرید و براساس این معیارها توضیح دهید به نظرتان از میان دو ابزار انتخابی کدام انتخاب بهتری بوده است؟

25

# 1 ترجمه ماشینی

ابتدا دو تا از ابزارهای معرفی شده در قسمت مقدمه را انتخاب کنید و برای انجام تمرین موارد زیر را در نظر بگیرید:

در این تمرین از دو ابزار

OpenNMT <https://opennmt.net/>

Fairseq <https://github.com/facebookresearch/fairseq>

استفاده می‌کنیم.

۱) همانطور که من دانیم در تسک NMT نیازمند پیش پردازش هستیم، یکی از روش های پیش پردازش اعمال bpe بر پیکره آموزشی است به نظر شما چه پیش پردازش دیگری برای ترجمه ماشینی مبتنی بر شبکه عصبی مفید است؟ به دلخواه خود دو روش پیش پردازش دیگر را، با ذکر دلیل، انتخاب کرده و نام ببرید.

یکی از مهم ترین پیش پردازش ها در زبان انگلیسی که بحث هم روی آن زیاد شده است، lowercase کردن متن است. با این کار باعث می شود تعداد ووکب بسیار کمتر شود و مدل راحت تر بتواند آموزش بینند. این موضوع برای این تمرين که مدل یک ترنسفورمر ساده است، دیتاست کوچک است و منابع سخت افزاری کافی برای آموزش طولانی مدت هم نداریم بسیار اهمیت می یابد.

ضمنا در فارسی از کتابخانه <https://github.com/ICTRC/Parsivar> برای پیش پردازش داده های فارسی استفاده می کنیم که شامل اصلاح کاراکترهای خاص برای جلوگیری از افزایش بیش از حد ووکب، بررسی punctuation ها و فاصله گذاری مناسب و پtern های متعددی که در کتابخانه موجود است اصلاح می شوند تا متن یکدست شود و مدل دچار سرگشتگی نشود. همچنین نیم فاصله ها و فاصله ها نیز اگر اشتباه باشند اصلاح می شوند و ضمنا متون فینگلیش هم به نوشته فارسی آنها بازگردانده می شوند تا همه چیز نرمال شود و مدل راحت تر بتواند آموزش بینند.

## ۲) به نظر شما در زبان فارسی کدام یک از این ۳ نوع پیش پردازش از اهمیت بیشتری برخوردار است؟ چرا؟ در زبان انگلیسی چطور؟

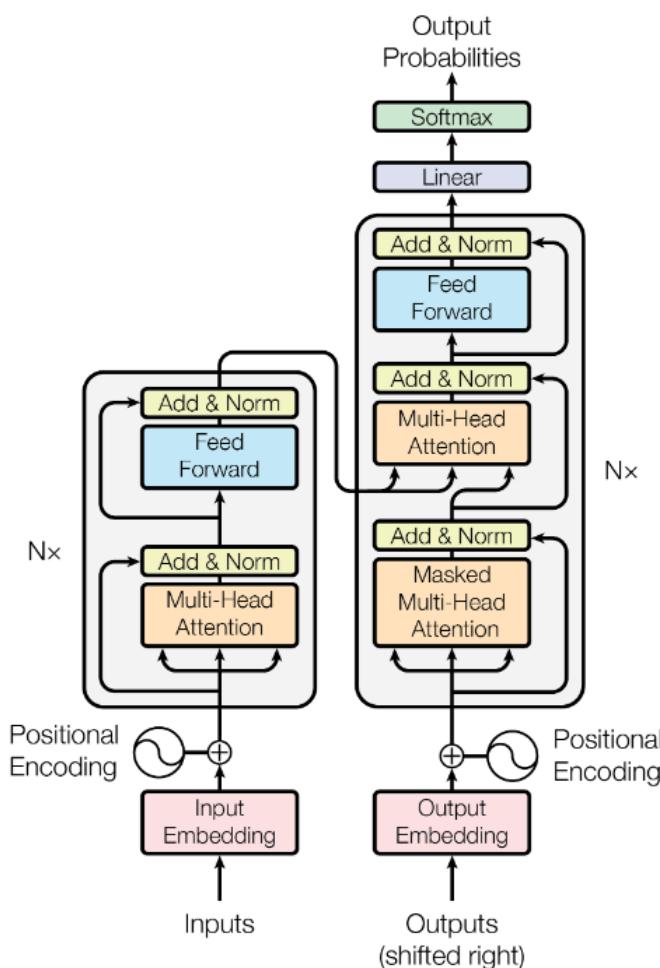
در زبان فارسی، چون morphology rich است، قطعاً مرحله BPE از اهمیت بسیار زیادی برخوردار است چون می‌تواند تا حدی این inflection های زبانی را شناسایی کند. مثلاً "تر" و "ترین" را بفهمد یا در افعال بتواند جداسازی هایی مثل "می" یا "خورد"+"م"/"نده" انجام دهد که قطعاً می‌تواند به مدل اصلی در فهم زبان کمک کند. در زبان انگلیسی هم اتفاقاً همین موضوع هست اما کمتر از فارسی البته باز توانایی تشخیص "er" مشابه "تر" فارسی اهمیت بالایی دارد. در مجموع نشان داده شده است که این روش در هر دو زبان می‌تواند نتایج را خیلی بهتر کند.

طبعتاً باقی موارد که گفته شد خاص زبان بود مثلاً lower case کردن معادل در فارسی ندارد و یا یکسان سازی نیم فاصله ها معادل در انگلیسی ندارد ولی اینها هم اهمیت دارند البته که همان BPE از اهمیت بیشتری برخوردار است.

(3) همانطور که من دانید معماری های مختلف برای آموزش مدل های **transformer** وجود دارد. در این تمرین قصد داریم تا مدلی بر مبنای معماری **sequence sequence** آموزش دهیم.

اتفاقاً مقاله اصلی machine translation نیز روی ترجمه transformer بوده است.

<https://arxiv.org/pdf/1706.03762.pdf>



شکل 1 - معماری مدل ترنسفورمر

بنابراین در ابزارها هم نوع معماری را ترنسفورمر قرار می‌دهیم.

۴) در این مرحله متناسب با ابزارهایی که انتخاب کرده اید، ۱۰ تا از پارامترهای آموزش مدل را شرح دهید (در هر ابزار به شکل جداگانه). ابزار مورد نظر خود را نام برد و ابتدا توضیح دهید که این پارامترها چه کاری انجام می دهند و سپس توضیح دهید چه پارامترهایی را لازم است تا نسبت به حالت پیش فرض تغییر دهید.

- .1. ابزار OpenNMT .1.1 Learning\_rate: نرخ یادگیری مدل برای آپدیت شدن پارامترها. مقدار پیش فرض برای این مقدار برابر با ۱ است.
- .1.2 Heads: تعداد head‌ها برای transformer در لایه attention. مقدار پیش فرض head برابر با ۸ است.
- .1.3 Batch\_size: حداکثر سایز batch برای آموزش داده‌ها. مقدار پیش فرض برابر با ۶۴ است.
- .1.4 Word\_size: تعداد کل پراسس‌های توزیع شده که در این پروژه چون تنها از یک gpu استفاده شده است مقدار آن برابر با ۱ در نظر گرفته شده است. مقدار پیش فرض نیز همین است.
- .1.5 Train\_steps: تعداد گام‌ها برای آموزش مدل. مقدار پیش فرض برابر با ۱۰۰۰۰۰ است. در واقع در صورتی که train\_steps = ۱۰۰۰۰ و batch\_size = ۶۴ باشد برای تمام شدن یک epoch لازم است تا ۱۵۶۲ گام انجام شود.
- .1.6 Save\_checkpoints\_step: پس از طی شدن گام‌های ذکر شده در این قسمت مدل به دست آمده ذخیره می‌شود. مقدار پیش فرض برای آن ۵۰۰۰ است. (در واقع checkpoint به ازای هر  $x$  گام ذخیره می‌شود)
- .1.7 Optim: روش بهینه سازی استفاده شده برای آموزش مدل. مقدار پیش فرض sgd است.
- .1.8 Src\_word\_vec\_size: سایز embedding برای داده‌های مبدا. مقدار پیش فرض برابر با ۵۰۰ در نظر گرفته شده است.
- .1.9 encoder|decoder)\_type: نوع لایه encoder یا decoder استفاده شده در مدل. به صورت پیش فرض rnn در نظر گرفته شده است.
- .1.10 enc|dec)\_layers: تعداد لایه‌ها در encoder یا decoder. مقدار پیش فرض برابر با ۲ است.

در پیاده سازی انجام شده تنها پارامتر های `valid_steps` و `train_steps`, `save_checkpoint_steps` برای بالا بردن سرعت آموزش مقدار آن کم شده است اگرچه دقت مدل را کاهش می دهد. اما در صورتی که دقت مهم تر از سرعت باشد می توان تعداد لایه های `embedding` را افزایش داد و یا اینکه تعداد `vec_size` برای `encoder` را بالا برد. علاوه بر این در صورتی که محدودیت سخت افزاری نداشته باشیم نیز می توانیم از تعداد بیشتری هسته برای پردازش موازی استفاده کنیم.

در ادامه نیز در فایلی دیگر نوع `decoder` و `encoder` را به `transformer` را به تغییر داده و از بهینه ساز `adam` با نرخ آموزش ۰۰۰۱ استفاده شد. اما از آن جا که نتایج به نسبت حالت قبل ضعیف تر بودند در ادامه نتایج آن نیامده است و تنها در پوشش گزارش وجود دارند.

## 2. ابزار Fairseq

پارامترهای این ابزار در لینک زیر توضیح داده شده است که به توصیف 10 مورد از آنها می‌پردازیم.

[https://fairseq.readthedocs.io/en/latest/command\\_line\\_tools.html#fairseq-train](https://fairseq.readthedocs.io/en/latest/command_line_tools.html#fairseq-train)

.2.1 no-progress-bar باعث می‌شود نوار پیشرفت نباشد و به جای آن

.2.2 Tensorboard-logdir با این پارامتر می‌توان از ابزار لگ تنسوربورد که برای tensorflow است بهره برد و اطلاعات و لاغهای طول آموزش را در آن ابزار به صورت نمودارهایی مشاهده کرد. این دستور به طور خاص، آدرس اتصال به ابزار تنسوربورد را می‌گیرد تا این کار را انجام دهد.

.2.3 Seed این پارامتر مقدار سید که توابع رندوم با آن کار می‌کند را مشخص می‌کند. در تحقیقات ماشین لرنینگ معمولاً یک اجرا نمی‌تواند نتیجه قطعی دهد و با استفاده از سیدهای مختلف و گزارش میانگین و انحراف معیار نتایج، می‌توان از نتیجه آزمایش خود مطمئن تر شد.

.2.4 Cpu این پارامتر باعث می‌شود از cpu به جای استفاده از ایترفیس cuda که برای gpu های nvidia است استفاده شود و برای زمانی که gpu در اختیار نداریم مناسب است.

.2.5 Tpu این پارامتر باعث می‌شود از tpu به جای cuda استفاده شود که tpu یا Tensor processing unit واحد پردازشی خاص هوش مصنوعی و ماشین لرنینگ است که توسط گوگل توسعه داده شده است و روی کولب هم قابل استفاده است.

.2.6 Fp16 باعث می‌شود از floating point عای 16 بیتی به جای 32 بیت استفاده شود که می‌تواند برای سرعت اجرا و مموری بهتر باشد با هزینه دقت محاسبات.

.2.7 Optimizer با این پارامتر می‌توان از بهینه سازی که مورد نظر است استفاده کرد که معمولاً Adam انتخاب می‌شود، اما گزینه های دیگری مثل adadelta, adafactor, adagrad, adam, adamax, composite, cpu\_adam, lamb, nag, sgd دارد.

- 2.8 Lr-scheduler این پارامتر می‌تواند روش برنامه ریزی learning rate را مشخص کند. معمولاً 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 scheduler است کم کم لرنینگ ریت در طول آموزش کم شود. برای این کار cosine, fixed, inverse\_sqrt, manual, pass\_through, polynomial\_decay, reduce\_lr\_on\_plateau, step, tri\_stage, triangular نکند.
- 2.9 Scoring نوع امتیازی که محاسبه می‌شود را مشخص می‌کند. این مقدار می‌تواند bert\_score, sacrebleu, bleu, chrf, meteor, wer همان blue که در این سوال هم خواسته شده است و نیاز به تغییر ندارد.
- 2.10 Task عملی که می‌خواهیم انجام دهیم و مدل را آموزش دهیم مشخص می‌کند که مثلاً می‌تواند language model یا خیلی موارد دیگر باشد و حالت دیفالت آن همان translation است که نیاز به تغییر ندارد.
- 2.11 Arch یا a پارامتری است که معماری مدل را مشخص می‌کند و می‌تواند مثلاً مدل‌های مختلف و سایرها مختلفشان را بگیرد که در این مسئله آن را به مقدار transformer می‌گذاریم.
- 2.12 Max-epoch حداکثر تعداد اپیاک آموزش است که بعد از آن متوقف شود. در این مسئله چون محدودیت داریم از این پارامتر استفاده می‌کنیم و عددی محدود می‌دهیم تا آموزش خیلی طولانی نشود.
- 2.13 Lr مقدار نرخ یادگیری را مشخص می‌کند که در ابتدای مدل است و مقدار اولیه 0.25 دارد و آن را نیاز نیست تغییر خاصی بدھیم.
- 2.14 Save-dir مسیر ذخیره سازی چک پوینت ها را مشخص می‌کند و دیفالت آن فولدر checkpoints است که مناسب است.

تعداد حداکثر چک پوینت های ایپاک ها که سیو شود را مشخص می کند و مقدار دیفالت -1 دارد که یعنی محدودیت ندارد.

صبر مدل در برابر بهبود نیافتن مدل روی داده dev را مشخص می کند و مقدار دیفالت -1 است یعنی متوقف نمی شود. ما این مقدار را به 3 تغییر می دهیم.

در ادامه کامل تر پارامترهایی که استفاده می کنیم توضیح داده می شود اما در این سوال هم ذکر می کنیم که پارامتر max-epoch را از مقدار پیش فرض -1 که یعنی محدودیت ندارد به یک مقدار محدود مثل 40 تغییر می دهیم تا آموزش بیش از حد طولانی نشود و کولب بسته نشود. ضمنا از patience هم استفاده می کنیم و به جای -1 یک عدد محدود مثل 5 می گذاریم تا هر وقت پیشرفته پس از 5 مرحله حاصل نمی شد متوف شویم و بیهوده آموزش را ادامه ندهیم.

5) پارامتری که فکر من کنید در کیفیت خروجی یک مدل ترجمه ماشینی تاثیرگذار هستند ولی امکان تغییر یا تنظیم آنها با توجه به منابع موجود (سخت افزاری و محدودیت های ابزار) وجود ندارد را نام برد و اهمیت هر یک را مختصررا توضیح دهید. (در هر ابزار به شکل جداگانه)

### ● ابزار OpenNMT

Word\_size ○: این پارامتر برابر با تعداد پراسس های توزیع شده است و در صورت داشتن سخت افزاری که gpu های بیشتری دارد می توان مقدار آن را افزایش داد و سرعت آموزش را بالا برد.

enc\_layer و dec\_layer ○: بالا بدن تعداد لایه های encoder و decoder می تواند باعث شود مدل پیچیدگی های بیشتری داشته باشد و مدل ها پیچیده تری را نیز یاد بگیرد اگرچه لازم است تا پارامتر های آموزش را جایی ذخیره کند که ممکن است با محدودیت RAM مواجه شویم.

Batch\_size ○: در برخی موارد بالا بدن سایز batch باعث می شود مدل بهتری داشته باشیم اما ممکن است به دلیل محدودیت حافظه قادر به بالا بدن آن نباشیم.

hidden\_state و dec\_rnn\_size ○: این پارامتر تعداد decoder در enc\_rnn\_size می تواند باعث شود مدل موردencoder را مشخص می کند که زیاد کردن آن می تواند باعث شود مدل مورد بیشتری یاد بگیرد اما مجدداً ممکن است برای ذخیره آنها با مشکل حافظه مواجه شود.

Heads ○: افزایش تعداد head های transformer نیز می تواند باعث شود مدل از جنبه های مختلفی یاد بگیرد اما ممکن است بالا بدن تعداد head ها تا هر حدی امکان پذیر نباشد و مشکل حافظه داشته باشیم.

- یک بخش بزرگی از پارامترها مربوط به distributed\_training است که یعنی بیش از یک عدد gpu داشته باشیم. که ما محدودیت داریم و همان یکی هم بعضی وقت‌ها نداریم.
- تعداد مجموع gpu های موجود را مشخص می‌کند Distributed-world-size
- تعداد Nprocs-per-node gpu در هر نود را مشخص می‌کند. اعمال محاسبات روی gpu های یک نود سریعتر است به همین دلیل اینکه روی هر نود چند gpu موجود است مهم است.
- سایز هر بچ را مشخص می‌کند که مناسب با آن حافظه GPU استفاده خواهد شد و به علت محدودیت حافظه نمی‌توانیم آن را خیلی زیاد کنیم. در بعضی موارد، افزایش سایز بچ می‌تواند مدل را بهتر به سمت مینیمم لاس هدایت کند و نتایج بهتری در انتها داشته باشد.
- تعداد ورکرها یا همان ترد و پراسس های همزمان را مشخص می‌کند اما چون روی کولب تنها یک هسته داریم، ساختن ورکر بیشتر فقط باعث ایجاد overhead می‌شود و از نظر سرعت بهتر نخواهد بود.
- ضمنا arch که قبلا توضیح داده شد هم می‌تواند مدل‌های بزرگتر با hidden states size و یا تعداد لایه بیشتر بگیرد که چون باعث بیشتر شدن پارامترها و کند شدن آموزش و نیاز به مموری بیشتر دارد برای منابع محدود ما مناسب نیست.

۶) حال من خواهیم ۲ مدل مختلف را با استفاده از ۲ ابزار انتخابی آموزش دهیم که جزییات این بخش به شرح زیر است:

» ابتداء bpe و دو، روشن پیش پردازش دیگر را بر روی دادگان آموزش اعمال کنید

- چگونگی و جزئیات پیاده سازی این بخش را در گزارش خود ذکر کنید.
  - انتخاب ابزار مناسب در این مرحله بر عینده شماست و محدودیتی وجود ندارد.
  - خروجی مرحله پیش پردازش را به شکل یک فایل جداگانه در گزارش خود بیاورید.

پیش پردازش‌هایی که اعمال می‌کنیم در بخش‌های قبلی توضیح داده شدند. در این بخش پیاده سازی و جزئیات آن را ذکر می‌کنیم.

شکل 2 - پیش پردازش‌های اولیه

در این بخش برای زبان فارسی با استفاده از کتابخانه parsivar پیش پردازش‌هایی مثل تغییر فینگلیش به فارسی، اصلاح نیم فاصله، اصلاح کاراکترهای عجیب، و اصلاح اسپیس گذاری انجام شده و برای انگلیسی متن به lower case اصلاح شده است.

سپس برای bpe چندین روش وجود دارد. مثلا با خود ابزار openNMT می توان این کار را انجام داد.

```
1 ! mkdir BPE

1 !python OpenNMT-py/tools/learn_bpe.py -i preprocessed_data/train.en -o BPE/src.code -s 10000

1 !python OpenNMT-py/tools/learn_bpe.py -i preprocessed_data/train.fa -o BPE/tgt.code -s 10000

1 !python OpenNMT-py/tools/apply_bpe.py -c BPE/src.code -i preprocessed_data/train.en -o BPE/train-bpe.en

1 !python OpenNMT-py/tools/apply_bpe.py -c BPE/src.code -i preprocessed_data/valid.en -o BPE/valid-bpe.en

1 !python OpenNMT-py/tools/apply_bpe.py -c BPE/src.code -i preprocessed_data/test.en -o BPE/test-bpe.en.txt

1 !python OpenNMT-py/tools/apply_bpe.py -c BPE/tgt.code -i preprocessed_data/train.fa -o BPE/train-bpe.fa

1 !python OpenNMT-py/tools/apply_bpe.py -c BPE/tgt.code -i preprocessed_data/valid.fa -o BPE/valid-bpe.fa

1 !python OpenNMT-py/tools/apply_bpe.py -c BPE/tgt.code -i preprocessed_data/test.fa -o BPE/valid-bpe.fa
```

شکل 3 - پیش پردازش BPE با OpenNMT

با استفاده از این ابزار ابتدا روی بخش ترین، توکنایزر BPE را آموزش می‌دهیم و سپس روی تمام بخشها اعمال می‌کنیم.

۱۰- بیان نکرده که شرایط اینکه بتواند رسم کار خود را آغاز کند می‌بایست توسط پادشاه تایید شود ام اگر پادشاه بتواند این را در می‌بیند و هشتاد و پنجمین نخسوزبیر در تاریخ تایید شود لند خواهد بود.

۱۱- بیان نکرده که شرایط اینکه بتواند رسم کار خود را آغاز کند می‌بیند و هشتاد و پنجمین نخسوزبیر در تاریخ تایید شود لند خواهد بود.

۱۲- بیان نکرده که شرایط اینکه بتواند رسم کار خود را آغاز کند می‌بیند و هشتاد و پنجمین نخسوزبیر در تاریخ تایید شود لند خواهد بود.

۱۳- بیان نکرده که شرایط اینکه بتواند رسم کار خود را آغاز کند می‌بیند و هشتاد و پنجمین نخسوزبیر در تاریخ تایید شود لند خواهد بود.

۱۴- بیان نکرده که شرایط اینکه بتواند رسم کار خود را آغاز کند می‌بیند و هشتاد و پنجمین نخسوزبیر در تاریخ تایید شود لند خواهد بود.

۱۵- بیان نکرده که شرایط اینکه بتواند رسم کار خود را آغاز کند می‌بیند و هشتاد و پنجمین نخسوزبیر در تاریخ تایید شود لند خواهد بود.

۱۶- بیان نکرده که شرایط اینکه بتواند رسم کار خود را آغاز کند می‌بیند و هشتاد و پنجمین نخسوزبیر در تاریخ تایید شود لند خواهد بود.

۱۷- بیان نکرده که شرایط اینکه بتواند رسم کار خود را آغاز کند می‌بیند و هشتاد و پنجمین نخسوزبیر در تاریخ تایید شود لند خواهد بود.

۱۸- بیان نکرده که شرایط اینکه بتواند رسم کار خود را آغاز کند می‌بیند و هشتاد و پنجمین نخسوزبیر در تاریخ تایید شود لند خواهد بود.

شکل 4 - نتیجه اجرای BPE

همانطور که دیده می شود کلمات به بخش‌های کوچکتر در مواردی شکسته شده‌اند مثلاً قدرت+مندی از هم جدا شده‌اند و انتهای بخش‌های شکسته شده که به بخش دیگر می‌چسبند برای مشخص شدن و امکان بازگشتن به متن اولیه (اگر همه اسپیس بودند فرق شکسته شدن و اسپیس واقعی مشخص نمی‌شد) کاراکتر @ گذاشته شده است.

برای اطمینان یک روش دیگر را نیز معرفی می‌کنیم. در این روش از کتابخانه هاگینگ فیس استفاده می‌کنیم.

```

1  from tokenizers import Tokenizer
2  from tokenizers.models import BPE
3  from tokenizers.pre_tokenizers import Whitespace
4  from tokenizers.trainers import BpeTrainer
5
6  tokenizer = Tokenizer(BPE())
7  tokenizer.pre_tokenizer = Whitespace()
8  trainer = BpeTrainer(special_tokens=["[UNK]"], continuing_subword_prefix="@")
9  for lang in ["en", "fa"]:
10     tokenizer.train(files=[f"preprocessed_data/train.{lang}",
11                           f"preprocessed_data/valid.{lang}",
12                           f"preprocessed_data/test.{lang}"],
13                           trainer=trainer)
14     text = "hello mohsen fayyaz" if lang == "en" else "سلام محسن فیاض"
15     output = tokenizer.encode(text)
16     print(tokenizer.decode(output.ids))

```

hello mohs @en f @ayy @az

سلام محسن فیا @@ض

شکل 5 - نتیجه اجرای BPE با هاگینگ فیس

همانطور که دیده می شود با این کتابخانه هم می توان توکنایزر تعریف کرد و با استفاده از آن را آموزش داد و در نهایت می توان برای زیرکلمه ها پریفیکس مشخص گذاشت که برای شناخت به حالت قبل @ گذاشتیم و می بینیم که این توکنایزر هم با کلماتی که کاملش را ندارد به صورت شکستن کلمه عمل می کند.

در انتها خروجی این دو مرحله در دو فایل مجزا در کنار گزارش آورده شده است.

➤ پارامترهای لازم، مناسب با دادگان، منابع موجود و صلاحدید خود مقداردهی کنید.

پارامترهای آموزش مدل را به شکل کامل در گزارش خود ذکر کنید. ○

openNMT .1

برای آموزش مدل از دستور زیر استفاده شده است:

```
python OpenNMT-py/train.py -data translate/data -save_model  
translate/model -train_steps 50000 -save_checkpoint_steps 5000  
-valid_steps 1000 -world_size 1 -gpu_rank 0
```

در این دستور تقریباً تمامی پارامترها به صورت پیش فرض انتخاب شده است تنها برای بالا بردن سرعت آموزش تعداد گام‌ها برای آموزش کاهش یافته است.

مدل آموزش داده شده به شکل زیر است:

```
(encoder): RNNEncoder(  
    (embeddings): Embeddings(  
        (make_embedding): Sequential(  
            (emb_luts): Elementwise(  
                (0): Embedding(9808, 500, padding_idx=1)  
            )  
        )  
    )  
    (rnn): LSTM(500, 500, num_layers=2, dropout=0.3)  
)  
(decoder): InputFeedRNND decoder(  
    (embeddings): Embeddings(  
        (make_embedding): Sequential(  
            (emb_luts): Elementwise(  
                (0): Embedding(10073, 500, padding_idx=1)  
            )  
        )  
    )  
    (dropout): Dropout(p=0.3, inplace=False)  
    (rnn): StackedLSTM(  
        (dropout): Dropout(p=0.3, inplace=False)  
        (layers): ModuleList(  
            (0): LSTMCell(1000, 500)  
            (1): LSTMCell(500, 500)  
        )  
    )  
    (attn): GlobalAttention(  
        (linear_in): Linear(in_features=500, out_features=500, bias=False)  
        (linear_out): Linear(in_features=1000, out_features=500, bias=False)  
    )  
)  
(generator): Sequential(  
    (0): Linear(in_features=500, out_features=10073, bias=True)  
    (1): Cast()  
    (2): LogSoftmax(dim=-1)  
)  
)
```

openNMT چه کنم NMT -6

همچنین در صورت مقدار دهی `decoder_type` و `encoder_type` می توان از دستور زیر استفاده کرد(فایل دوم که در قسمت ۴ به آن اشاره شد). این مقدار دهی به این دلیل انجام شد تا تقریباً پارامتر ها با ابزار دیگر یکسان باشد اما نتایج آن خوب نبود.

```
python OpenNMT-py/train.py -data translate/data -save_model
translate/model -param_init_glorot -label_smoothing 0.1
-enc_layers 2 --dec_layers 2 -rnn_size 512 -word_vec_size 512
-batch_size 4096 -batch_type tokens -optim adam -adam_beta2 0.998
-learning_rate 0.001 -normalization tokens -encoder_type
transformer -decoder_type transformer -train_steps 20000
-save_checkpoint_steps 2000 -valid_steps 1000 -world_size 1
-gpu_rank 0
```

## FairSeq .2

ابتدا از `preprocess` استفاده می کنیم تا فایلهای `bin` که مورد نیاز آموزش است ساخته شود. برای ورودی به این تابع، زبان مبدا و مقصد و فایلهای ترین و ولیدیشن و تست را می دهیم و نوع `bpe` را مشخص می کنیم و فرمت لاغ را مشخص می کنیم.

```
1 fairseq_preprocess = """
2 rm -r data-bin/
3 TEXT=/content/preprocessed_data
4 fairseq-preprocess --source-lang en --target-lang fa \
5   --trainpref $TEXT/train --validpref $TEXT/valid --testpref $TEXT/test \
6   --destdir data-bin/data.tokenized.en-fa \
7   --workers 20 \
8   --bpe bert \
9   --log-format json \
10 """
11 run_bash(fairseq_preprocess)

11.preprocessing | Namespace(align_suffix=None, alignfile=None, all_gather_list_size=16384, bf16=False, bpe='bert',
11.preprocessing | [en] Dictionary: 33040 types
11.preprocessing | [en] /content/preprocessed_data/train.en: 30000 sents, 544389 tokens, 0.0% replaced by <unk>
11.preprocessing | [en] Dictionary: 33040 types
11.preprocessing | [en] /content/preprocessed_data/valid.en: 400 sents, 6945 tokens, 3.01% replaced by <unk>
11.preprocessing | [en] Dictionary: 33040 types
11.preprocessing | [en] /content/preprocessed_data/test.en: 427 sents, 11267 tokens, 3.71% replaced by <unk>
11.preprocessing | [fa] Dictionary: 44136 types
11.preprocessing | [fa] /content/preprocessed_data/train.fa: 30000 sents, 520509 tokens, 0.0% replaced by <unk>
11.preprocessing | [fa] Dictionary: 44136 types
11.preprocessing | [fa] /content/preprocessed_data/valid.fa: 400 sents, 6500 tokens, 5.42% replaced by <unk>
11.preprocessing | [fa] Dictionary: 44136 types
11.preprocessing | [fa] /content/preprocessed_data/test.fa: 427 sents, 11376 tokens, 6.14% replaced by <unk>
11.preprocessing | Wrote preprocessed data to data-bin/data.tokenized.en-fa
```

شکل 7 - تابع پیش پردازش `fairseq`

بعد از آن به آموزش مدل می پردازیم.

```
fairseq-train \
  data-bin/data.tokenized.en-fa \
```

```

--arch transformer --share-decoder-input-output-embed \
--optimizer adam --adam-betas '(0.9, 0.98)' --clip-norm 0.0 \
--lr 5e-4 --lr-scheduler inverse_sqrt --warmup-updates 4000 \
--dropout 0.3 --weight-decay 0.0001 \
--criterion label_smoothed_cross_entropy --label-smoothing 0.1 \
--max-tokens 4096 \
--eval-bleu \
--eval-bleu-args '{"beam": 5, "max_len_a": 1.2, "max_len_b": 10}' \
--eval-bleu-detok moses \
--eval-bleu-remove-bpe \
--eval-bleu-print-samples \
--best-checkpoint-metric bleu --maximize-best-checkpoint-metric \
--max-epoch 50 \
--patience 10 \
--save-dir checkpoints \
# --fp16 \
--batch-size 64

```

در این ابزار معماری را ترسنفورمر می‌گذاریم. اپتیمایزر را بهینه ساز معروف adam می‌گذاریم و مقادیر بتای آن را مشخص می‌کنیم. لرنینگ ریت را  $5e-4$  می‌گذاریم و ضمناً scheduler تعريف می‌کنیم تا در طول آموزش کاهش یابد.

همچنین از warmup استفاده می‌کنیم که شروع آموزش را با لرنینگ ریت کم شروع می‌کند تا اول آموزش وزنها به جاهای اشتباه پرت نشوند و کم کم به محلی که درست است منتقل شوند و سپس به لرنینگ ریت اولیه میرسیم.

از dropout استفاده کردیم تا حدود 0.3 نورون‌ها به صورت رندوم خاموش شوند تا از overfitting مدل جلوگیری شود و generalization بهتر داشته باشیم.

برای ارزیابی از معیار bleu استفاده می‌کنیم و بهترین چکپوینت از این نظر را ذخیره می‌کنیم در کنار همه چکپوینت‌ها. ضمناً بج سایز را 64 قرار می‌دهیم و آدرس ذخیره سازی چکپوینت را مشخص می‌کنیم و البته patience هم مشخص می‌کنیم که صبر مدل برای early stopping است تا اگر بعد از 10 مرحله بهبودی حاصل نشد متوقف شویم.

و ضمناً بیشترین تعداد ایپاک را هم محدود به 50 می‌کنیم تا زمان بیش از حد گرفته نشود که کولب

بسته شود و همه زحمت ها به باد رود.

ضمنا می توانیم به جای fp32 از fp16 استفاده کنیم اما چون تفاوت سرعت زیادی در تغییر آن دیده نشد از آن صرف نظر کردیم.

## > آموزش مدل را تا زمانی که در میانه آموزش قرار گیرد ادامه دهد.

توصیه میشد تا از google colab برای حل این تمرین استفاده

در این تمرین قصد داریم با روند آموزش یک مدل ماشین ترجمه و الزامات آن آشنا شویم . انتظار ما این است که مدل نهایی که ارائه می دهد در میانه مسیر آموزش باشد. یکی از راه های بررسی این موضوع کنترل دستی خروجی مدل بر روی داده های تست است برای مثال معمولاً خروجی یک مدل ترجمه که تازه شروع به آموزش کرده است، تکرار تنها چند کلمه خاص است و بدینه است که این مدل به عنوان مدل نهایی پذیرفته نیست.

با توجه به کم بودن حجم مجموعه داده اولیه و محدودیت های منابع انتظار، تولید یک ماشین با کیفیت را نداریم. نگران نتایج ضعیف احتمال نباشید :

انتظار نداریم که مجموع زمان آموزش در هر ابزار بیش از ۶ ساعت به طول انجامد و الزاماً صرف زمان بیشتر برای آموزش مدل امتیاز محسوب نمی شود.

لطفاً علاوه بر فایل گزارش، فایل اسکریپت دستورات اجرا شده و یا اگر در Google Colab اجرا گردد اید فایل notebook آن به همراه خروجی سیستم های آموزش داده شده برای فایل های تست را نیز ارسال کنید.

آموزش هر دو ابزار انجام شد و نوت بوک ها و خروجی ها در فایلهای کنار گزارش آمده اند.

نمونه ای از نتایج هم در زیر آمده که نشان می دهد مدل در میانه آموزش بوده و مدل فقط تکرار نمی کند.

ملموس قبل از اینکه بتواند به کار رود این کشور را فرابگیرد ، باید سکوت کند .

اگر مسیری های شاه از او رنج بگیرد ، در " تاریخ اتحاد " در " تاریخ " خواهد بود .

هود در همان محلی که از زندان به زندان آمده بود در حمله به زندان می بخشید .

امروز دشیب گفت : خیل خطرناک ترین کار برای آینده و خواهرت وجود دارد .

میان حزب دموکراتیک و قانونگذاری که بسی در جامعه عمیقاً رشد می کند ، در سال گذشته و به سوی نظامی عبور می کند .

بیش از 90 نفر کشته و صدها نفر زخمی شدند .

7) پس از آموزش بایستی، وند تغییرات Bleu دو مدل آموزش دیده را بر روی مجموعه dev با افزایش تعداد epoch‌ها نشان دهید. برای این کار، میتوانید از دستور ذخیره مدل‌های مبیانی در ابزار مورد نظر خود استفاده کنید. (حداقل 5 نقطه را در طول آموزش مدل گزارش دهید)

openNMT .1

تغییرات bleu به ازای هر 5000 گام به صورت زیر است:

```
[['BLEU = 2.04, 20.3/3.5/0.9/0.3 (BP=1.000, ratio=1.163, hyp_len=7351, ref_len=6322)'],
 ['BLEU = 3.14, 29.8/6.8/2.0/0.5 (BP=0.850, ratio=0.860, hyp_len=5437, ref_len=6322)'],
 ['BLEU = 3.84, 32.2/7.6/2.3/0.8 (BP=0.842, ratio=0.853, hyp_len=5394, ref_len=6322)'],
 ['BLEU = 3.86, 32.3/7.9/2.4/0.8 (BP=0.808, ratio=0.824, hyp_len=5210, ref_len=6322)'],
 ['BLEU = 4.44, 31.1/7.8/2.6/0.9 (BP=0.907, ratio=0.911, hyp_len=5760, ref_len=6322)'],
 ['BLEU = 4.32, 31.6/7.9/2.5/0.9 (BP=0.885, ratio=0.891, hyp_len=5633, ref_len=6322)'],
 ['BLEU = 4.04, 31.0/7.2/2.3/0.9 (BP=0.862, ratio=0.870, hyp_len=5502, ref_len=6322)'],
 ['BLEU = 3.96, 31.9/7.7/2.4/0.8 (BP=0.846, ratio=0.857, hyp_len=5416, ref_len=6322)'],
 ['BLEU = 4.07, 32.8/7.9/2.5/0.9 (BP=0.842, ratio=0.853, hyp_len=5394, ref_len=6322')],
 ['BLEU = 4.20, 30.5/7.5/2.5/0.8 (BP=0.908, ratio=0.912, hyp_len=5764, ref_len=6322)']]
```

شکل 7 - تغییرات bleu در openNMT

همانطور که دیده می‌شود حداً کثر مقدار 4.44 است.

شکل 8 - تغییرات bleu در fairseq

همانطور که دیده می‌شود مقدار blue در این ابزار حداکثر به ۲.۰۶ می‌رسد.

## ۸) با کمک دادگان موازی test، مقدار Bleu را برای این دو مدل بر روی دادگان آزمایشی گزارش دهید.

openNMT .1

```
!perl OpenNMT-py/tools/multi-bleu.perl data/test.fa < pred/test/pred_test.txt  
BLEU = 3.94, 33.7/8.0/2.3/0.6 (BP=0.891, ratio=0.897, hyp_len=10208, ref_len=11385)
```

شکل ۸- مقادیر bleu بر روی داده های تست به کمک openNMT

3.94

Fairseq .2

```
! fairseq-generate data-bin/data.tokenized.en-fa --path  
checkpoints/checkpoint_best.pt --batch-size 128 --beam 5 --remove-bpe  
--eval-bleu --results-path generate_results  
Generate test with beam=5: BLEU4 = 1.22, 23.3/2.6/0.4/0.1 (BP=1.000,  
ratio=1.047, syslen=11460, reflen=10949)
```

1.22

## ۹) ۳ تا از معیارهایی که فکر میگنید برای ارزیابی یک ابزار تولید ماشین ترجمه مناسب هستند را نام ببرید و براساس این معیارها توضیح دهید به نظرتان از میان دو ابزار انتخابی کدام انتخاب بهتری بوده است؟

- امکان تنظیم پارامترها جهت آموزش مدل مناسب با داده های در دسترس
- امکان موازی سازی پردازش ها جهت بالا بردن سرعت آموزش
- وجود آموزش و داکیومنتیشن های کافی برای کار با ابزار
- پشتیبانی اسپانسرهای مطرح از ابزار
- جامعه توسعه دهندهای بزرگ
- سهولت استفاده از ابزار
- بهینه بودن نتایج نهایی ابزار
- پشتیبانی از سخت افزارهای به روز مانند TPU
- پشتیبانی از framework های متنوع

از میان دو ابزار، در مجموع openNMT گزینه بهتری نسبت به Fairseq بود. اولا نتایج خیلی بهتری را با پارامترهای معمول آموزش توانست به دست آورد که نشان از بهینه تر بودن پایپلاینهای اجرایی آن دارد. ضمنا از نظر داکیومنتیشن غنی تر بود و بهتر پارامترهای آن توضیح داده شده بود. (مثلا یکی از پارامترهایی که در داکیومنتیشن fairseq بود اصلا در دستور آن کار نمی کرد!) از نظر پشتیبانی از openNMT نیز، ابزار fairseq پشتیبانی می کند در حالی که ابزار Framework هم از Pytorch و هم از Tensorflow پشتیبانی می کند که بهتر است.

برای تکمیل بحث، معیارهای ارزیابی خود ماشین ترجمه را نیز توضیح می دهیم.

### BLEU

این معیار با داشتن تعدادی متن اصلی و ترجمه آن به عنوان رفرنس سعی می کند تا بر اساس تعداد n-gram های تطابق یافته میان ترجمه اصلی و ترجمه انسان یک عدد به مدل نسبت دهد. مشکل این روش این است که این تطابق ها باید دقیقا مشابه متن مرجع باشند و در صورت استفاده از واژگان یا عبارات متفاوت معیار BLEU کاهش می یابد.

### NIST

این معیار تقریباً شبیه به BLEU است با این تفاوت که در n-gram ها وزن های یکسانی داده می شود اما در این روش به n-gram هایی وزن بیشتری داده می شود که احتمال اتفاق افتادن آن ها کمتر باشد.

## TER

این متریک مبتنی بر کاراکتر عمل می کند به این صورت که تعداد ادیت های لازم برای تبدیل متن ترجمه شده توسط ماشین به متن ترجمه شده توسط انسان را معیاری برای ارزیابی قرار می دهد.