به نام خدا

دانشگاه تهران

دانشکدگان فنی

دانشکده مهندسی برق و کامپیوتر

# درس پردازش زبان طبیعی

پاسخ تمرین ۵

نام و نام خانودگی: زهرا ریحانیان

شماره دانشجویی: ۸۱۰۱۰۱۱۷۷

**خرداد ماه ۱۴۰۳**

# فهرست

## پاسخ سوال اول

کد مربوط به این بخش در مسیر codes/Q1.ipynb موجود است.

بعد از نصب و import کتابخانه های لازم، موارد زیر انجام شد.

### دادگان

ابتدا داده ها را از لینک داده شده دانلود و unzip کردم.

۱. تعداد کل خطوط:

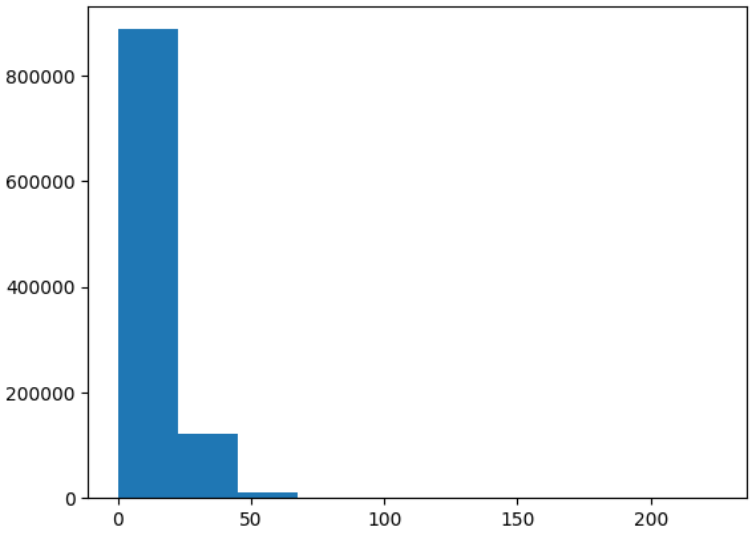
انگلیسی: ۱۰۲۱۵۹۷

فارسی: ۱۰۲۱۵۹۷

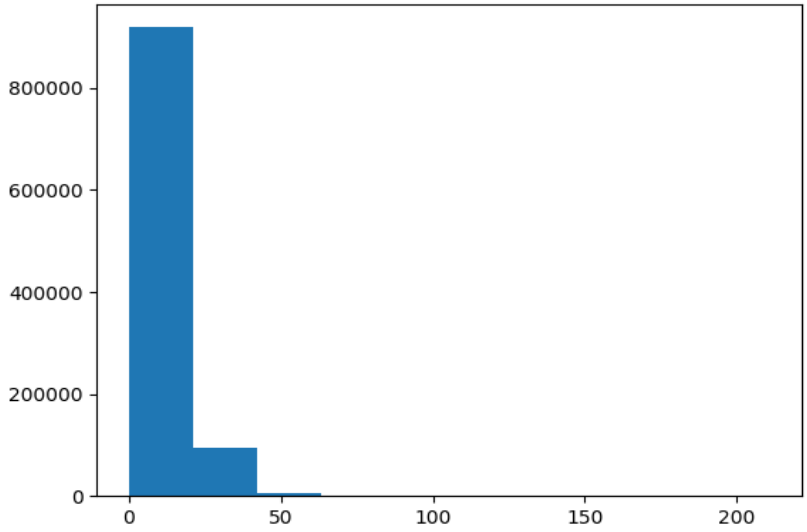
محتوای سه خط اول:

| | |
|---|---|
| انگلیسی: | The story which follows was first written out in Paris during the Peace Conference<br>from notes jotted daily on the march, strengthened by some reports sent to my chiefs in Cairo .<br>Afterwards, in the autumn of 1919, this first draft and some of the notes were lost. |
| فارسی: | داستانی که از نظر شما می‌گذرد، ابتدا ضمن کنفرانس صلح پاریس از روی یادداشت‌هائی که به طور روزانه در حال خدمت در صف برداشته شده بودند<br>و از روی گزارشاتی که برای رؤسای من در قاهره ارسال گردیده بودند نوشته شد.<br>بعدا در پائیز سال ۱۹۱۹، این نوشته اولیه و بعضی از یادداشت‌ها، مفقود شدند. |

۲. برای توکن سازی داده ها در هر سطر، ابتدا که یونیکد نیم فاصله است را با فاصله جا گذاری می کنیم سپس آن را با کمک متد split بر اساس فاصله جداسازی می کنیم. هیستوگرام خواسته شده برای تعداد توکن های هر سطر به صورت زیر بدست آمد:




شکل ۱ هیستوگرام تعداد توکن های فارسی در هر سطر




شکل ۲ هیستوگرام تعداد توکن های انگلیسی در هر سطر

۳. برای کاهش حجم دیتاست، در بین توکن ها فارسی در هر سطر، سطر هایی که بیش تر از ۵۰ و یا کمتر از ۱۰ توکن داشتند، حذف شد. همچنین سطر متناظر آن در دیتای انگلیسی نیز حذف شد. من برای این قسمت یک دیکشنری با دو کلید en و fa ساختم و در یک حلقه for، سطر هایی از داده فارسی را که بیش تر از ۵۰ و یا کمتر از ۱۰ توکن داشتند، skip کردم وگرنه آن را به دیکشنری به همراه سطر متناظر آن در انگلیسی، اضافه کردم. تعداد سطر های جدید ۵۸۹۸۷۰ تا شد.

۴. برای این قسمت، من ابتدا دیکشنری بدست آمده از مرحله قبل را تبدیل به یک دیتافریم کردم. آن را با random_state=۴۲ که کار random seed را در متد sample دیتافریم می کند، shuffle کردم و تقسیم بندی های خواسته شده را انجام دادم.

۵. داده های بدست آمده در مرحله قبل هر کدام در ۶ فایل مجزا ذخیره شدند.

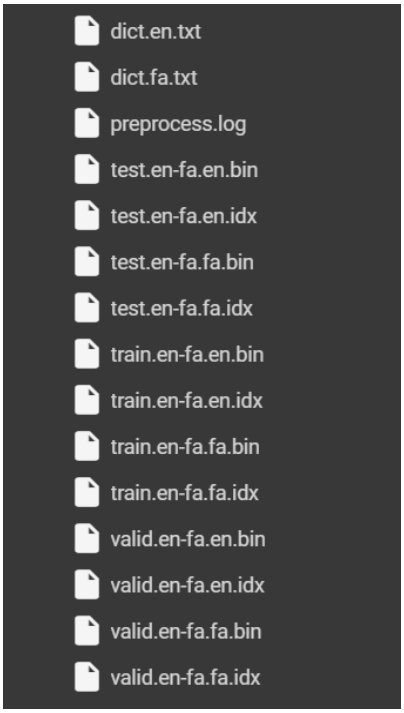
## پاسخ بخش اول: آموزش توکنایزر BPEو پیش پردازش دادگان

در این بخش با استفاده از کتابخانه sentencepiece بر روی دادگان آموزش فارسی و انگلیسی به طور جداگانه، توکنایزر با روش bpe آموزش داده شد. سپس مدل های آموزش دیده فارسی و انگلیسی برای توکنایز داده ها، لود شد و هر یک از ۶ فایل ساخته شده در قسمت قبل، با استفاده از آن ها توکنایز شدند. برای این کار یک تابع تعریف کردم که مدل، مسیر فایل داده و مسیر فایل خروجی را میگیرد. سپس هر دو فایل را باز می کند. از فایل داده، خط به خط می خواند و آن را با مدل encode یا در واقع توکنایز می کند و آن را در فایل خروجی می نویسد.

در مرحله بعد نوبت به پیش پردازش داده ها می رسد که این کار با fairseq-preprocess انجام شد. من این قسمت از کد را برای هر یک از مدل ها به طور جداگانه انجام دادم و کد آن در ابتدای Section2 و Section3 نوت بوک پیوست شده، است. پیش پردازش همان گونه که در تمرین خواسته شده بود، انجام شد.

فرمان fairseq-preprocess کاری که انجام می دهد این است که وکب را برای هر زبان می سازد و داده ها را به فرم باینری تبدیل می کند که بتوان با آن، مدل را با fairseq آموزش داد.

بعد از اجرای آن، این فایل ها تولید شد:




**شکل ۳ فایل های تولید شده بعد از فرمان fairseq-preprocess**

## پاسخ بخش دوم: آموزش مدل LSTM encoder-decoder

در این قسمت ابتدا دو پوشه checkpoints و logs برای ذخیره بهترین مدل و log هنگام آموزش، ساخته شد. بعد از آن از دستور fairseq-train برای آموزش مدل translation استفاده شد به صورت زیر:

```
!fairseq-train \
    "./data_bin/" \
    --arch lstm --encoder-layers 6 --decoder-layers 6 \
    --task translation \
    --optimizer adam --adam-betas '(0.9, 0.98)' --clip-norm 0.1 \
    --lr 2e-3 --lr-scheduler inverse_sqrt --warmup-updates 4000 \
    --dropout 0.2 --weight-decay 0.0 \
    --criterion label_smoothed_cross_entropy --label-smoothing 0.2 \
    --max-tokens 4096 \
    --batch-size 128 \
    --fp16 --memory-efficient-fp16 \
    --max-epoch 5 \
    --keep-best-checkpoints 1 \
    --eval-bleu \
    --eval-bleu-detok moses \
    --eval-bleu-print-samples \
    --best-checkpoint-metric bleu --maximize-best-checkpoint-metric \
    --fp16 --memory-efficient-fp16 \
    --save-dir ./data_bin/checkpoints/ \
    --log-file training.log \
    --tensorboard-logdir ./data_bin/logs \
```

توضیحات هایپرپارامتر های مهم:

--arch --encoder-layers 6 --decoder-layers 6

معماری مدل را مشخص می کند. در اینجا همان طور از خواسته شده بود، از مدل encoder decoder با lstm که هر کدام از encoder و decoder دارای ۶ لایه هستند، استفاده شد.

--potimizer adam --adam-betas '(0.9, 0.98)'

بهینه ساز را مشخص می کند که در اینجا از بهینه ساز adam با پارامتر های beta به مقدار 0.9 و 0.98 استفاده شد.

--clip-norm 0.1

تعیین حداکثر آستانه برای norm گرادیان هاست که اگر از این آستانه که در اینجا 0.1 گذاشتم، تجاوز کرد، گرادیان ها به تناسب کاهش می یابند تا به این آستانه برسند.

--lr 2e-3

نرخ یادگیری را مشخص می کند که آن را 2e-3 قرار دادم. با مقادیر بزرگتر یا کوچکتر از این مقدار، مدل جواب خوبی را نمی داد.

--lr-scheduler inverse_sqrt

زمانبندی نرخ یادگیری را مشخص می کند که آن را inverse_sqrt قرار دادم. برنامه های نرخ یادگیری به دنبال تنظیم نرخ یادگیری در طول آموزش با کاهش نرخ یادگیری بر اساس یک برنامه از پیش تعریف شده هستند.

--warmup-updates 4000

این پارامتر تعداد به‌روزرسانی‌ها (مراحل آموزشی) را مشخص می‌کند که طی آن نرخ یادگیری به صورت خطی از مقدار کوچک (اغلب نزدیک به صفر) به نرخ یادگیری هدف مشخص شده توسط "lr--" افزایش می‌یابد. با مقدار ۴۰۰۰ و نرخ یادگیری مشخص شده، به بهترین نتیجه رسیدم.

--dropout 0.2

تعیین میزان dropout که به جلوگیری از overfitting کمک می کند.

--weight-decay 0.0

این پارامتر برای تعیین کاهش وزن (L2 regularization) اعمال شده بر روی وزن های مدل در طول تمرین استفاده می شود.

--criterion label_smoothed_cross_entropy --label-smoothing 0.2

تابع خطا را مشخص می کند که همان طور که خواسته شده بود از تابع خطای label_smoothed_cross_entropy با مقدار label-smoothing برابر با 0.2 استفاده شد.

--max-tokens 4096

--batch-size 128

هایپرپارمتر max-token حداکثر تعداد توکن ها را در هر batch تنظیم می کند. هایپرپارامتر batch-size حداکثر تعداد جملات در هر دسته را کنترل می کند که برای پردازش دسته ای سازگار مفید است. در سایر پیاده سازی ها، ممکن است مینی دسته ها را از نظر تعداد جملات مشخص کنند. به عنوان مثال، اندازه دسته ای ۶۴ به این معنی است که هر دسته کوچک دارای ۶۴ جمله است. با این حال، از آنجایی که جملات طول‌های متفاوتی دارند، به این معنی است که هر دسته کوچک می‌تواند دارای تعداد بسیار متفاوتی از نشانه‌ها باشد. در fairseq-py ما به صراحت حداکثر تعداد توکن‌ها را در هر دسته مشخص می‌کنیم و سپس دسته‌ها را با هر تعداد جمله پر می‌کنیم[1]. در اینجا max-token را برابر 4069 قرار دادم. این به این معنی است که تعداد کل توکن ها در یک دسته از ۴۰۹۶ تجاوز نمی کند. این برای کنترل حافظه مفید است. از آنجایی که مصرف حافظه GPU با تعداد توکن ها بیشتر از تعداد جملات مرتبط است. همچنین batch-size را ۱۲۸ قرار دادم. این هایپرپارامتر تضمین می کند که یک دسته بدون توجه به

[1] https://github.com/facebookresearch/fairseq/issues/37

تعداد کل نشانه ها، بیش از ۱۲۸ جمله نداشته باشد. Batch-size تضمین می کند که تعداد جملات در هر دسته قابل پیش بینی است، که می تواند به کنترل تکرارهای آموزشی و حفظ یک برنامه آموزش ثابت کمک کند.

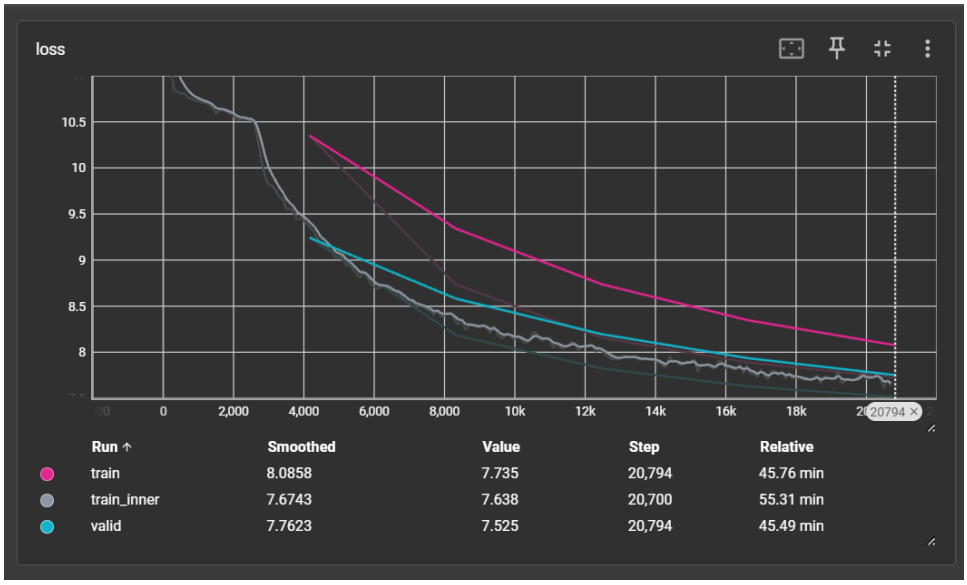
--max-epoch 5

تعداد ایپاک های آموزش، یعنی چند بار مدل کل داده ها را می بیند، همان طور که خواسته شده بود برابر ۵ قرار دادم.

--keep-best-checkpoints 1

تعداد بهترین نقاط بازرسی را برای حفظ در طول آموزش مشخص می‌کند. این امکان می‌دهد که بهترین مدل را ذخیره کنیم. من معیار bleu را برای ذخیره بهترین مدل، تنظیم کردم. یعنی هر مدلی که در طول آموزش از نظر bleu بهتر باشد، ذخیره خواهد شد.

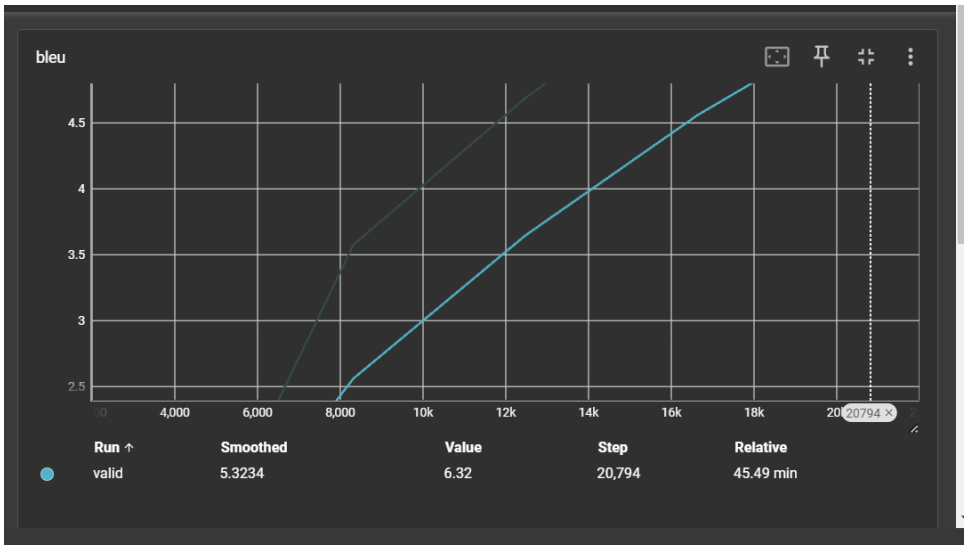
بعد از آموزش مدل، با استفاده از ابزار tensorboard لاگی که مدل در طول آموزش ذخیره کرده بود را به صورت نمودار، نمایش دادم. نمودار زیر مربوط به Loss مدل در طول آموزش است:




شکل ۴ نمودار loss در طول آموزش برای مدل با lstm

در پوشه codes/lstm_loss که در کنار همین گزارش پیوست شده است، فایل های log مقادیر loss برای آموزش و ارزیابی با فرمت csv موجود است.

نمودار زیر نمودار مقادیر bleu مدل را در طول آموزش نشان می دهد:




شکل ۵ نمودار bleu در طول آموزش برای مدل با lstm

در ادامه کد مربوط به بخش چهارم تمرین یعنی ارزیابی مربوط به این مدل را زدم.

بعد از آموزش مدل، فرمان fairseq-generate استفاده شد:

```
!fairseq-generate \
    "./data_bin" \
    --batch-size 64 \
    --path "./data_bin/checkpoints/checkpoint_best.pt" \
    --beam 5 > "./data_bin/results/new_eval.txt"
```

بعد از اجرای این دستور، فایل خروجی آن را با استفاده از کد پایتون خوانده شد که معیار Bleu را برای داده های تست نمایش داده شود و نتیجه این شد:

```
Generate test with beam=5: BLEU4 = 6.33, 35.8/10.5/4.2/1.8 (BP=0.872,
ratio=0.880, syslen=186010, reflen=211406)
```

یعنی معیار Bleu حدود 6.33 برای داده های تست بدست آمد که بهترین نتیجه ای بود که توانستم بدست بیاورم.

در مرحله بعد برای بدست آوردن معیار comet، لازم بود تا مدل های توکنایزر برای هر زبان لود شوند. به همین خاطر ابتدا این مدل ها لود شدند. با استفاده از پایتون، فایل خروجی fairseq-ganerate را باز کردم و در هر خط اگر با S- شروع شد یعنی source است و به لیست sentences_source اضافه می شود، اگر با T- شروع شد یعنی target است و به لیست target_ sentences اضافه می شود و در نهایت اگر با H- شروع شد یعنی machine-translated است و به لیست translation_ sentences اضافه می شود. در نهایت روی تک تک المان های هر لیست به صورت جداگانه متد decode از توکنایزر مربوطه اعمال شد. یعنی برای source توکنایزر انگلیسی و برای دوتای دیگر توکنایزر فارسی اعمال شد.

بعد از آن unbabel-comet را نصب کردم و مدل "Unbabel/wmt22-comet-da" را دانلود کردم و سپس آن را لود کردم. دیتای مورد نیاز آن را به فرمتی که می خواست تبدیل کردم. یعنی یک حلقه for روی لیست هایی که در مرحله قبل به دست آوردم زدم و آن ها در یک لیست که شامل دیکشنری هایی از داده های source، target و translation است، ذخیره کردم. model.predict را روی این داده ها با batch_size=8 اعمال کردم و از score های بدست آمده که همان معیار comet هستند، میانگین گرفتم و مقدار 0.55 را بدست آوردم.

## پاسخ بخش سوم: آموزش مدل Transformer encoder-decoder

در این بخش به خاطر اروری که موقع اجرای fairseq-train می گرفتم، مجبور شدم یک سری کتابخانه را به همراه fairseq دوباره نصب کنم که این ارور برطرف شود. در ادامه همانند بخش قبل، ابتدا fairseq-preprocess و پس از آن fairseq-train اجرا شد. از همان هایپرپارامتر های بخش قبل استفاده کردم به جز lr و warmup که آن ها را این گونه تنظیم کردم:

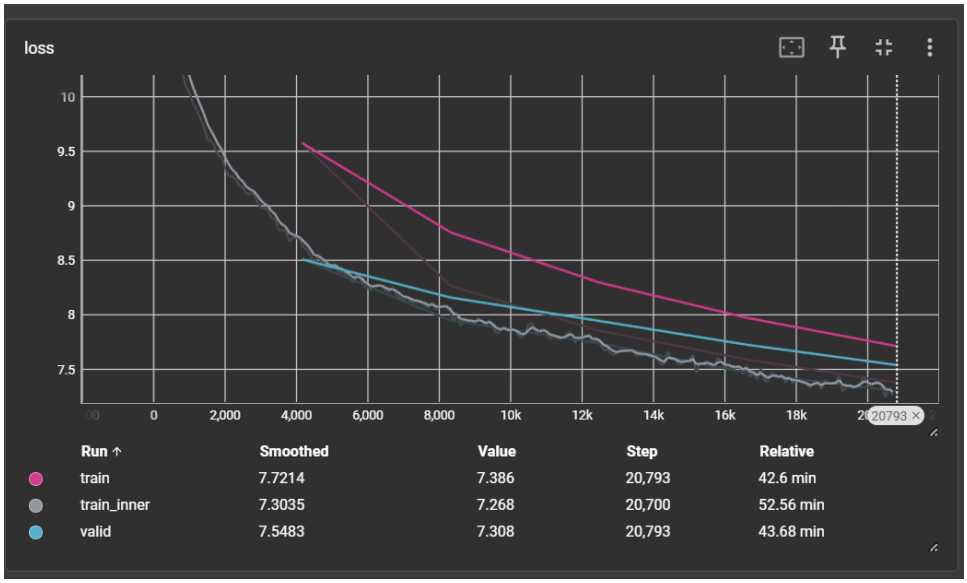
--lr 3e-4

--warmup-updates 6000

--warmup-init-lr 1e-07

با این مقادیر بهترین نتیجه را در بین تلاش هایی که داشتم، بدست آوردم.

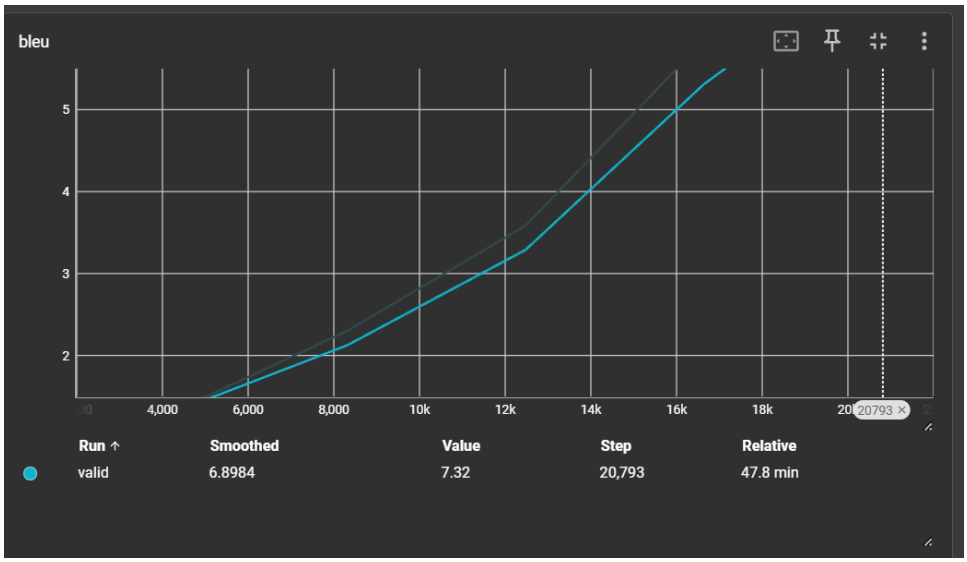
مانند بخش قبل، بعد از آموزش مدل، با استفاده از ابزار tensorboard لاگی که مدل در طول آموزش ذخیره کرده بود را به صورت نمودار، نمایش دادم. نمودار زیر مربوط به Loss مدل در طول آموزش است:




شکل ۶ نمودار loss در طول آموزش برای مدل با transfomer

در پوشه codes/transformer_loss که در کنار همین گزارش پیوست شده است، فایل های log مقادیر loss برای آموزش و ارزیابی با فرمت csv موجود است.

نمودار زیر نمودار مقادیر bleu مدل را در طول آموزش نشان می دهد:




شکل ۷ نمودار bleu در طول آموزش برای مدل با transformer

در ادامه کد مربوط به بخش چهارم تمرین یعنی ارزیابی مربوط به این مدل را زدم.

همانند بخش قبل بعد از آموزش مدل، فرمان fairseq-generate استفاده شد و پس از آن فایل خروجی تولید شده را با استفاده از پایتون خواندم تا مقدار bleu را برای داده های تست، نمایش دهم که این حاصل شد:

```
Generate test with beam=5: BLEU4 = 7.54, 37.9/12.3/5.2/2.3 (BP=0.876, ratio=0.883, syslen=186643, reflen=211406)
```

یعنی معیار Bleu حدود 7.54 برای داده های تست بدست آمد که بهترین نتیجه ای بود که توانستم بدست بیاورم.

در مرحله بعد، همان کار هایی که برای آماده سازی داده های مورد نیاز comet (پردازش فایل خروجی fairseq-generate و decode آن و در نهایت ذخیره در لیستی از دیکشنری ها) را انجام دادم و همان مدل "Unbabel/wmt22-comet-da" را دانلود کردم. در نهایت به طور میانگین مقدار 0.60 را برای معیار comet روی داده های تست، بدست آوردم که این هم در بین تلاش هایی که داشتم بهترین بود.

## پاسخ بخش چهارم

قسمت های مربوط به ارزیابی هر مدل در بخش مخصوص به خود گفته شد.

در این بخش به معرفی Comet و مقایسه نتایج comet و bleu می پردازیم.

معیار ارزیابی COMET (متریک بهینه شده بین زبانی برای ارزیابی ترجمه) یک معیار مبتنی بر یادگیری ماشینی است که برای ارزیابی کیفیت سیستم‌های ترجمه ماشینی طراحی شده است. هدف آن ارائه یک ارزیابی دقیق تر و قابل اعتمادتر در مقایسه با معیارهای سنتی مانند BLEU است که بر همپوشانی n-gram بین ترجمه تولید شده توسط ماشین و یک یا چند ترجمه مرجع متکی است.

COMET از مدل های عصبی از پیش آموزش دیده استفاده می کند، که اغلب بر اساس معماری هایی مانند BERT، RoBERTa، یا XLM-R هستند که در وظایف تخمین کیفیت ترجمه به خوبی تنظیم شده اند. این مدل ها به گونه ای طراحی شده اند که شباهت معنایی و روان بودن ترجمه ها را به تصویر بکشند. هم جمله منبع، هم ترجمه ماشینی و هم ترجمه مرجع (در صورت موجود بودن) به صورت نمایش های برداری متراکم در یک فضای جاسازی مشترک کدگذاری می شوند. این به مدل اجازه می دهد تا محتوای معنایی جملات را فراتر از تطبیق نشانه surface-level مقایسه کند. مدل‌های COMET بر روی مجموعه داده‌های بزرگ نمرات کیفیت ترجمه مشروح‌شده توسط انسان آموزش داده می‌شوند. فرآیند آموزش شامل یادگیری پیش‌بینی نمره کیفیت بر اساس جاسازی‌های منبع، مرجع و ترجمه فرضیه است. در طول ارزیابی، COMET جمله منبع، ترجمه ماشینی و در صورت تمایل یک ترجمه مرجع را می گیرد و امتیاز کیفی ایجاد می کند. این امتیاز به گونه ای طراحی شده است که نشان دهد ترجمه ماشینی تا چه حد معنای جمله منبع را به خوبی منتقل می کند و چقدر روان و طبیعی است. به طور کلی، COMET نشان دهنده پیشرفت قابل توجهی در ارزیابی کیفیت ترجمه است که از قدرت یادگیری عمیق و مدل های زبانی از پیش آموزش دیده برای ارائه ارزیابی های دقیق تر و معنادارتر استفاده می کند.

مقایسه نتایج comet و blue

| | bleu | comet |
|---|---|---|
| **مدل با lstm** | 6.33 | 0.55 |
| **مدل با transformer** | 7.54 | 0.60 |

مقدار comet یک مقداری بین صفر و یک است و هر چه به یک نزدیک تر باشد یعنی مدل ما کار ترجمه را بهتر انجام داده است و نشان دهنده کیفیت بالای ترجمه مطابق با مدل COMET است که با قضاوت انسان ارتباط نزدیک تری دارد. معیار bleu هم هر چقدر بالاتر باشد یعنی ترجمه تولید شده بیشتر شبیه ترجمه اصلی آن است. می بینیم که در اینجا این دو معیار با هم رابطه ای مستقیم دارند. یعنی با افزایش یکی، دیگری هم افزایش پیدا کرده است. همچنین مدل transformer نتایج بهتری را در مقایسه با lstm گرفته که احتمالا به خاطر ساختار پیچیده تر آن می باشد.

معیار Bleu یک ارزیابی ساده و مبتنی بر n گرم از کیفیت ترجمه ارائه می دهد. برای مقایسه سریع خوب است، اما ممکن است به طور کامل تفاوت های ظریف ترجمه را درک نکند.

معیار comet با در نظر گرفتن اطلاعات معنایی و زمینه ای، ارزیابی دقیق تری را ارائه می دهد که منجر به قضاوت های همسو با انسان می شود.