دانشگاه تهران

پردیس دانشکده‌های فنی

دانشکده برق و کامپیوتر

تکلیف کامپیوتری سری چهارم

درس پردازش زبان طبیعی

دکتر فیلی – دکتر یعقوب‌زاده

شیدا اسحاقی

۸۱۰۱۹۹۰۸۴

پرنیان فاضل

۸۱۰۱۹۸۵۱۶

بهار ۱۴۰۱

۱. ابزارهای انتخاب شده OpenNMT و Fairseq هستند که برای پیش پردازش مدل به کمک هر کدام مراحل زیر را داریم.

- اولین نکته مورد بررسی که در این corpora رعایت شده است قرار گیری یک جمله در هر سطر است که با یکی از علائم نگارشی "."، "!" و یا "؟" پایان یافته است.
- تمام کلمات را با یک space جدا می‌کنیم و به عبارتی چند space را با یک space جابجا می‌کنیم.
- با حذف نیم فاصله نیز مرحله دیگری از پیش پردازش را انجام می‌دهیم.
- به کمک کتابخانه hazm یا parsivar پیش پردازش زبان فارسی را انجام می‌دهیم. Normalization یکی از پیش پردازش‌های مهم به کمک این کتابخانه‌ها است. تغییر اعداد به زبان انگلیسی نیز بخشی از عملیات این کتابخانه است.
- در مورد کلماتی که علائم نگارشی به آن‌ها چسبیده است نیز به این صورت عمل می‌کنیم که ابتدا این علائم را با فاصله از کلمات جدا می‌کنیم و مراحل دیگر را بر روی کلمات انجام می‌دهیم.
- برای دیتاست انگلیسی ابتدا تمام کلمات را به کلمات با حروف کوچک تبدیل می‌کنیم.
- علت استفاده نکردن از lemmatization و stemming اهمیت داشتن شکل اصلی کلمات است. با حذف پیشوند ها یا تغییر در شکل کلمات ممکن است موجب تغییر ترجمه آن‌ها شویم.
- در آخر tokenization پیش پردازش انجام شده برای هر دو دیتاست می‌باشد.
- استفاده از فایل preprocess.py در این بخش نیز یکی از مراحل پیش پردازش ما است.

در مدل با استفاده از OpenNMT به کمک ابزار طراحی شده در این مترجم ماشینی می‌توانیم از bpe tokenization به صورت خودکار بهره ببریم. در این مرحله ما به کمک فایل‌های پیش پردازش شده فارسی و انگلیسی فایل bpe tokenized آن‌ها را استخراج می‌کنیم. فرآیند به کمک فایل‌های learn_bpe و apply_bpe صورت می‌گیرد.

در شکل زیر تابعی که برای پیش پردازش فایل های داده شده پیاده سازی شده است را میبینیم.

```
from parsivar import Normalizer
from parsivar import SpellCheck
parsivar_normalizer = Normalizer(pinglish_conversion_needed=True)
parsivar_tokenizer = Tokenizer()

def no_space(char, prev_char):
        return char in set(',.!?') and prev_char != ' '

def farsi_preprocess():
  farsi_files_to_be_preprocessed = ["train.fa", "valid.fa", "test.fa"]
  for files in farsi_files_to_be_preprocessed:
    lines = []
    with open(files) as file:
        for line in file:
          line = ''.join([' ' + char if i > 0 and no_space(char, line[i - 1]) else char for i, char in enumerate(line)])
          line = re.sub('[)(]{-]', ' ', line)
          lines.append(' '.join(parsivar_tokenizer.tokenize_words(parsivar_normalizer.normalize(line.rstrip())))) #word_tokenize in parsivar
        with open(f"preprocessed_{files}", 'w') as f:
          f.write('\n'.join(lines))

def english_preprocess():
  english_files_to_be_preprocessed = ["train.en", "valid.en", "test.en"]
  for files in english_files_to_be_preprocessed:
    lines = []
    with open(files) as file:
        for line in file:
          line = ''.join([' ' + char if i > 0 and no_space(char, line[i - 1]) else char for i, char in enumerate(line)])
          line = re.sub('[)(]{-]', ' ', line)
          lines.append(' '.join(nltk.word_tokenize(' '.join([word.lower() for word in line.split(' ')]))))
        with open(f"preprocessed_{files}", 'w') as f:
          f.write('\n'.join(lines))

farsi_preprocess()
english_preprocess()
```

به کمک کتابخانه parsivar مراحل tokenize و normalize کردن متن فارسی را پیاده سازی می‌کنیم. سپس تمام پیش پردازش‌های گفته شده در بالا مانند هندل کردن نیم فاصله، نرمال سازی اعداد انگلیسی و حذف پرانتز را انجام میدهیم. برای tokenization متن انگلیسی نیز از nltk استفاده می کنیم. سپس این فایل های پیش پردازش شده را با نام های جدید در پروژه ذخیره میکنیم و به عنوان ورودی به مراحل بعدی میدهیم.

۲. در پردازش زبان فارسی به دلیل morphological rich بودن و تغییر زیاد کلمات با بن واژه‌های یکسان نیاز داریم تا به خوبی پیشوند و پسوندهای کلمات را تشخیص دهیم و جداسازی بن واژه ها به خوبی شکل بگیرد که به کمک bpe و tokenizer ای که با زبان فارسی سازگار باشد به خوبی می توان این کار را انجام داد. همچنین به دلیل اینکه ۹۰ درصد از مشکلات نوشتاری در زبان فارسی بر اثر جابجایی یا مشکلات بر پایه فاصله است، هندل کردن نیم فاصله، و تعدد فاصله نیز می‌تواند بخش مهمی از پیش پردازش باشد.

در زبان انگلیسی یکسان کردن capitalization حروف پیش پردازشی مهم محسوب می‌شود.

۴.

ابزار Fairseq :

۱. Optimizer: الگوریتم‌ها یا روش‌هایی هستند که برای تغییر ویژگی‌های شبکه عصبی مانند وزن‌ها و نرخ یادگیری برای کاهش تلفات استفاده می‌شوند و برای انجام این تسک با کمینه سازی تابع استفاده می شوند. {مقادیر مجاز: adadelta, adafactor, adagrad, adam, adamax, composite, cpu_adam, lamb, nag, sgd که برای آموزش مدل با این ابزار از nag استفاده می‌شود که با حالت پیش فرض تفاوت دارد که از بهینه ساز استفاده نمی‌کند. }
۲. Lr: این پارامتر نشان دهنده‌ی مقدار نرخ یادگیری است که مقدار پیش‌فرض آن برابر 0.25 است و به دلیل خطای overflow در حین یادگیری مدل آن را برابر ۰/۰۱ قرار می‌دهیم.
۳. Max-tokens: برابر است با تعداد بیشترین توکن های در یک batch است که ما در اینجا برابر ۴۰۰۰ قرار داده ایم و در حالت پیش فرض مقداری ندارد.
۴. Arch: تقریبا مهم ترین پارامتر این مدل برای بخش آموزش است که مشخص کننده معماری این model است. مقادیر متنوعی می‌تواند بگیرد که دو مقداری که ما استفاده کرده ایم fconv_iwslt_de_en و mbart_base است.
۵. Clip-norm: این پارامتر برای مقداردهی میزان آستانه gradient است که مقدار پیش فرض ۰/۲۵ است اما ما برابر ۰/۱ قرار می دهیم.
۶. Max-epoch: برابر با بیشترین تعداد epoch مه در صورت مقدار دهی نشدن خاتمه نمی‌یابد.
۷. Tonekizer: نوع توکنایز شدن متن برای آموزش را انتخاب می کنیم که در اینجا moses قرار می دهیم.
۸. Save dir: محل ذخیره شدن checkpoint ها در سیستم را مشخص می‌کند.
۹. Criterion : برای انتخاب criterion مدل استفاده می‌شود که به صورت پیش فرض cross entropy است.
۱۰. Bpe : در این ابزار به کمک این پارامتر می توانیم جزییات توکنایز شدن با روش bpe را مشخص کنیم که مقادیر مجاز عبارتند از: byte_bpe, bytes, characters, fastbpe, gpt2, bert, hf_byte_bpe, sentencepiece, subword_nmt که این مقدار را ما در preprocessing مشخص کردیم.

پارامترهای arch, batch-size, max-epoch, max-token, leraning-rate,optimizerروی خروجی مدل تاثیر بسیاری دارند و آن ها را از حالت پیش فرض خارج کرده ایم. در آخر با اضافه کردن fp16– فرایند آموزش مدل سریع تر انجام می شود.

ابزار openNMT:

۱. layers: این پارامتر تعداد لایه‌های شبکه را مشخص میکند که به صورت پیش فرض روی ۱- ست شده است.
۲. heads: تعداد head های لایه attention را مشخص میکند. با افزایش این پارامتر، شبکه میتواند ارتباط بیشتری بین کلمات پیدا کند.
۳. rnn_type: نوع gate در rnn را مشخص میکند.
۴. batch_size: حدامثر اندازه batch در مرحله آموزش را مشخص میکند که مقدار پیش‌فرض آن برابر ۶۴ است.
۵. dropout: احتمال dropout را مشخص می‌کند که در LSTM stack ها اثر میگذارد که مقدار پیش فرض آن برابر ۰/۳ است.
۶. learning_rate: این پارامتر نشان دهنده‌ی مقدار نرخ یادگیری است که مقدار پیش‌فرض آن برابر ۱ است.
۷. tgt_word_vec_size: این پارامتر اندازه word embedding مقصد را مشخص میکند که مقدار پیش فرض آن برابر ۵۰۰ است.
۸. encoder_type: این پارامتر نوع شبکه های رمزنگار را مشخص میکند که میتوان طبق زبان که ترجمه میشود و زبان ترجمه شده، انتخاب شود. انتخاب های موجود برای این پارامتر rnn و brnn و ggnn و mean و transformer و cnn و transformer_lm هستند. عملکرد این پارامتر مانند decoder_type است.

۹. optim: روش بهینه‌سازی را مشخص میکند که میتواند یکی از روش‌های sgd, adagrad, adadelta, adam, sparseadam, adafactor, fusedadam باشد که به طور پیش‌فرض sgd است.
۱۰. early_stopping: تعداد validation steps هایی است که بهبودی در نتیجه مدل ایجاد نشده است. به صورت پیش فرض برابر صفر است.

پارامترهایی مثل تعداد لایه‌ها، batch_size، learning_rate و encoder_type و decoder_type را بهتر است تغییر دهیم تا بتوانیم مدل را قوی‌تر کرده و نتیجه بهتری بگیریم.

۵. در ابزار :Fairseq

Learning rate, batch size, learning rate scheduler و arch می توانند تاثیر زیادی در خروجی مدل داشته باشند.

Lr-schedular: تعیین می کند تا نرخ یادگیری در هر مرحله چطور تغییر کند و مقادیر مختلفی مانند fixed,inverse_sqrt cosine, triangular ,tri_stage step reduce_lr_on_plateau, polynomial_decay, ,manual,pass_through, را می تواند بگیرد تا نتیجه بهتری بدهد اما به صورت پیش فرض فیکس است و پس از تغییر احتمال رخ دادن overflow در هر مرحله بیشتر می شود.
برای پارامتر های دیگر در بالا توضیح داده شد و با تغییر هرکدام محدودیت های GPUو COLAB این امکان را از ما میگرفت.

در ابزار Opennmt:

- ✓ پارامتر world_size: این پارامتر تعداد distributed process ها را نشان میدهد که به صورت پیش فرض ۱ است. اگر میتوانستیم این مقدار را افزایش دهیم میتوانستم با سرعت بیشتری پردازش‌ها را انجام دهیم.
- ✓ پارامتر gpu_ranks: این پارامتر تعداد رنک‌های هر process را نشان میدهد که من روی google colab میتوانستم تنها صفر بگذارم اما با افزایش آن(یعنی مثلا این پارامتر را به 0,1,2,3 مقدار میدادیم) میتوانستیم پردازش‌های سریعتر و بهتری داشته باشیم.
- ✓ Heads: این پارامتر را در سوال ۴ توضیح دادیم. با وجود اینکه افزایش این پارامتر میتواند باعث قدرتمندتر شدن مدل شود اما به دلیل کمبود حافظه و gpu، برای این پارامتر محدودیت وجود دارد.
- ✓ پارامتر batch_size: میدانیم که این پارامتر روی خروجی مدل خیلی تاثیر گذار است اما وقتی مقدار آن از یک حدی بیشتر میشد google colab ارور کم بودن حافظه gpu را میداد و مجبور بودیم مقدار آن را محدود کنیم.

۶. Fairseq: برای پیاده سازی bpe در این مدل ما نیازی به توکنایز کردن جداگانه نداریم و کافی است با اضافه کردن این پارامتر در preprocess این را انجام دهیم.

```
!fairseq-preprocess --source-lang en --target-lang fa --bpe byte_bpe --tokenizer moses --optimizer nag
```

قطعه کد بالا بخشی از مرحله پیش پردازش این مدل به کمک Fairseq را نشان می دهد که سه فایل پیش پردازش شده به کمک تابع های تعریف شده در پروژه برای پیش پردازش کلی برای هر دو ابزار را به عنوان ورودی می‌گیرد و روی آنها پردازش انجام میدهد.
لازم به ذکر است که استفاده از توکنایزر sobword-snmt را نیز به عنوان یک روش دیگر خارج از حالت byte_bpe امتحان کردیم که نتیجه این توکنایز اصلا خوب نبود و به نظر میرسید برروی زبان فارسی خوب عمل نمیکند در نتیجه از پارامتر خود Fairseq preprocess استفاده کردیم.

فایل خروجی این پیش پردازش در فولدر پروژه با عنوان data-bin/custom.tokenized.en-faذخیره شده است. همچنین tokenizer moses و optimizer nag پس از امتحان کردن حالت های دیگر بهترین خروجی را نتیجه میدادند.
عکس زیر بخشی از فایل train فارسی پس از پیش پردازش اولیه به کمک تابع نوشته شده است که تمام نیم فاصله ها، کلمات چسبیده به علائم نگارشی، اعداد فارسی، جایگزینی کاراکتر های خاص و نرمال سازی روی آن هندل شده است.

```
. ]مگی درآمد که : شاید فقط رفته به واهاین
. اینها رو از یه جسد برداشتم و ظاهر پوست و استخوانش : میگوید رو عوض کردم
. اسب روی علف‌های خیس لیز می‌خورد
. اما یک نوع رشوهی دیکری هم هست که گاهی اسمش را موفقیت مالی می‌گذارند
قطار هاگوارتز نیز زودتر از آن که انتظارش می‌رفت شروع به کم کردن سرعتش کرد و در ایستگاه نه و سه چهارم متوقف شد
نیان در قالب مقالات و در زمینه علوم پایه از " نمایه‌نامه استنادی علوم " استخراج‌شده و وضعیت ایران با جهان مقایسه شده‌است
در آن دم که آدم در خواب‌بود از پهلوی چپ او دنده‌ای برداشت
? جان گفت : پیرهنمو در بیارم
اگرچه پلک زیرین چشم‌های گوریو قدری ورم کرده و برگشته بود
. صورتش همه ترس‌های مرا و چیز دیگری را به من برگرداند
ن موقع ، ادموند مشغول صحبت با جناب گرانت شد تا قضایای کار و بارشان را حل و فصل کنند ، و هر دو گرم صحبتشان شدند
. با این حال او گفت دامنه آن به اندازه عملیات مارجاه نخواهدبود
. اکثر روستاها برق یا آب جاری ندارند
، دولت و تجار فشار می‌آوردند تا موجب ترقی و پیشرفت مملکت بشوند ; فعالیت‌هایی که برای آن‌زمان بسیار شهامت می‌خواستند
زیرا بچه که فن خویش را خوب فراگرفته‌بود و به فوت و فن کاسه گری آشنایی وافر داشت ، خویشتن را به بی‌حالی زد
. گریزی نداشتند مگر آن که با نصف غذای سگ اسکیمو که به آن‌ها می‌دادند ، بمیرند
. سایر سربازان در ساختمان‌های همسایه که پیش‌تر تخلیه شده‌بودند ، موضع گرفتند
```

عکس زیر نیز بخشی از فایل dict-fa است که پس از انجام پیش پردازش ابزار Fairseq توسط آن ساخته می شود.

مانند همین فایل برای زبان en نیز انجام شده که فایل آن در پروژه با عنوان preprocessed-fairseq قرار گرفته است.

```
. 21427
و 16643
، 15896
به 14476
در 13779
که 13199
از 10797
را 10123
این 7008
با 5376
است 3622
او 3479
یک 3356
برای 3299
: 2958
آن 2919
" 2689
من 2445
گفت 2308
کرد 2111
خود 2057
```

OpenNMt:

در ابتدا لازم است ذکر شود که ما از برنچ legacy ریپازیتوری گیتهاب openNMT استفاده کردیم تا از امکانات بیشتر آن مثل preprocess.py استفاده کنیم.

```
[10] !git clone -b legacy https://github.com/OpenNMT/OpenNMT-py

Cloning into 'OpenNMT-py'...
remote: Enumerating objects: 17675, done.
remote: Total 17675 (delta 0), reused 0 (delta 0), pack-reused 17675
Receiving objects: 100% (17675/17675), 273.61 MiB | 29.02 MiB/s, done.
Resolving deltas: 100% (12753/12753), done.
```

با استفاده از قطعه کد زیر مراحل پیش پردازش را که در سوال ۱ و ۲ بیشتر توضیح داده شد، اعمال می‌کنیم:

```
from parsivar import Normalizer
from parsivar import SpellCheck
parsivar_normalizer = Normalizer(pinglish_conversion_needed=True)
parsivar_tokenizer = Tokenizer()

def no_space(char, prev_char):
        return char in set(',.!?') and prev_char != ' '

def farsi_preprocess():
  farsi_files_to_be_preprocessed = ["train.fa", "valid.fa", "test.fa"]
  for files in farsi_files_to_be_preprocessed:
    lines = []
    with open(files) as file:
        for line in file:
          line = ''.join([' ' + char if i > 0 and no_space(char, line[i - 1]) else char for i, char in enumerate(line)])
          line = re.sub('[)(}{-]', ' ', line)
          lines.append(' '.join(parsivar_tokenizer.tokenize_words(parsivar_normalizer.normalize(line.rstrip())))) #word_tokenize in parsivar
        with open(f"preprocessed_{files}", 'w') as f:
          f.write('\n'.join(lines))

def english_preprocess():
  english_files_to_be_preprocessed = ["train.en", "valid.en", "test.en"]
  for files in english_files_to_be_preprocessed:
    lines = []
    with open(files) as file:
        for line in file:
          line = ''.join([' ' + char if i > 0 and no_space(char, line[i - 1]) else char for i, char in enumerate(line)])
          line = re.sub('[)(}{-]', ' ', line)
          lines.append(' '.join(nltk.word_tokenize(' '.join([word.lower() for word in line.split(' ')]))))
        with open(f"preprocessed_{files}", 'w') as f:
          f.write('\n'.join(lines))

farsi_preprocess()
english_preprocess()
```

در این قسمت vocabulary را ساختیم و برای اعمال BPE، از ابزار openNMT استفاده کردیم و همچنین از preprocess.py که در این ابزار وجود دارد استفاده کردیم. استفاده از BPE باعث میشود تا مدل بتواند کلمات ناآشنا را بهتر شناسایی کند. جزئیات بیشتر این بخش در زیر آمده است:

```
[13] !python OpenNMT-py/tools/learn_bpe.py -i preprocessed_train.en -o BPE_train.en -s 10000

[14] !python OpenNMT-py/tools/learn_bpe.py -i preprocessed_train.fa -o BPE_train.fa -s 10000

[15] !python OpenNMT-py/tools/apply_bpe.py -c BPE_train.fa -i preprocessed_train.fa -o train_bpe.fa

[16] !python OpenNMT-py/tools/apply_bpe.py -c BPE_train.en -i preprocessed_train.en -o train_bpe.en

[17] !python OpenNMT-py/tools/apply_bpe.py -c BPE_train.fa -i preprocessed_valid.fa -o valid_bpe.fa

[18] !python OpenNMT-py/tools/apply_bpe.py -c BPE_train.en -i preprocessed_valid.en -o valid_bpe.en

[19] !python OpenNMT-py/tools/apply_bpe.py -c BPE_train.en -i preprocessed_test.en -o test_bpe.en

[20] !python OpenNMT-py/preprocess.py -train_src train_bpe.en -train_tgt train_bpe.fa -valid_src valid_bpe.en -valid_tgt valid_bpe.fa -save_data BPE_Model
```

میتوانید فایل‌های مربوط به خروجی پیش پردازش را در فولدر preprocess_openNMT ببینید که شامل خروجی های پیش‌پردازش‌شده کورپس‌های train و validation و test است و همچنین خروجی BPE هم آورده شده است.
موارد دیگری مثل تبدیل حروف انگلیسی کلماتی مثل isaf که در کورپس‌ها آمده به حروف فارسی و یا در نظر گرفتن Named Entityها و Part of Speech کلمات بود.

نتایج و تحلیل آزمایشات انجام شده بر روی Fairseq:

برای بررسی نتیجه این مدل که روی ابزار Fairseq آموزش داده شده است ما به دو روش عمل کرده‌ایم. اولین آن‌ها بررسی معماری transformer-based است که پس از آزمایش چندین مدل pretrain شده بر پایه transformer و مشاهده نتایج بسیار ضعیف آن ها که حاکی از آموزش ندیدن مدل برای زبان فارسی بوده است، مدلی بر پایه mbart_base را پیاده سازی کردیم. Mbart از این جهت برای ما اهمیت دارد که multiligual است و زبان فارسی را هرچند ناکارآمد اما به هرحال آموزش می بیند.

نتیجه روی معماری Mbart_base:

برای پیاده سازی آموزش با این معماری کد زیر را استفاده می کنیم.

```
!CUDA_VISIBLE_DEVICES=0 fairseq-train data-bin/custom.tokenized.en-fa \
   --optimizer nag --lr 0.3 --clip-norm 0.25 --dropout 0.2 --max-tokens 6000 --fp16 --max-epoch 25\
   --arch mbart_base --save-dir checkpoints/models
```

که تمامی پارامترها در سوال پیش توضیح داده شد. لایه های این مدل mbart را بررسی میکنیم. این معماری شامل transformer encoder و decoder transformer است که هرکدام از این encoder و decoder از لایه های مختلف دیگری تشکیل شده اند که هر دوی آن ها لایه attention layer را نیز دارند. با بررسی log این آموزش نکات زیر به دست می آیند.

- زمان آموزش زیاد است و با fp16– می توان مقداری این زمان را بهینه کرد.
- با بررسی perplexity و loss از شروع تا پایان epoch ۲۵ که در اینجا در نظر گرفته ایم این دو مقدار به شدت تغییر میکنند و این مسئله میتواند یادگیری خوب مدل را نشان دهد. پس از epoch ۲۵ نیز هم زمان زیادی صرف می شود و هم به overfit شدن مدل نزدیک می شویم.

در دو شکل زیر مقایسه loss و perplexity دو مرحله اول و آخر را داریم.

برای epoch اول:

```
2022-06-09 10:50:48 | INFO | train | epoch 001 | loss 11.382 | ppl 2668.82
```

برای epoch نهایی:

```
2022-06-09 11:38:50 | INFO | train | epoch 025 | loss 1.021 | ppl 2.03
```

پس از آموزش مدل باید به کمک Fairseq-genetare برای این مدل جملات را بر اساس آموزشی که داده ایم تولید کنیم و در نهایت با بررسی بهترین checkpoint از مدل های تولید شده در مراحل مختلف که آخرین epoch است، مقادیر bleu را گزارش کنیم.

```
!fairseq-generate /content/data-bin/custom.tokenized.en-fa \
    --path /content/checkpoints/models/checkpoint_best.pt \
    --batch-size 512 --beam 3 \
    --skip-invalid-size-inputs-valid-test
```

برای این بخش batch size را بیشتر کرده و برابر ۵۱۲ قرار میدهیم و beam = 4 که این مقدار در واقع پارامتر اصلی برایbeam search ای است که این ماشین ترجمه برای پیشنهاد جمله ترجمه شده نهایی انجام میدهد. شکل زیر خروجی bleu برای داده های تست است.

```
2022-06-09 11:39:35 | INFO | fairseq_cli.generate | Translated 427 sentences (16358 tokens) in 8.9s (48.01 sentences/s, 1839.19 tokens/s)
Generate test with beam=3: BLEU4 = 0.62, 15.3/1.5/0.2/0.0 (BP=1.000, ratio=1.428, syslen=15931, reflen=11154)
```

پس از اینکه مدل ساخته شد یکی از ارزیابی هایی که میتوانیم انجام دهیم استفاده از متد interactive با این ابزار است که به شکل زیر پیاده سازی می شود.

```
!fairseq-interactive \
    --path /content/checkpoints/models/checkpoint_best.pt /content/data-bin/custom.tokenized.en-fa \
    --beam 4 --source-lang en --target-lang fa \
    --tokenizer moses \
```

پس از اجرای کد بالا میتوانیم جملات انگلیسی را به این مدل بدهیم و جملات ترجمه شده فارسی را بررسی کنیم. برای چند نمونه به شکل زیر داریم.

```
S-0     he said
W-0     0.180   seconds
H-0     -0.7647108435630798     گفت
D-0     -0.7647108435630798     گفت
P-0     -1.0824 -0.4471
this is a book
S-1     this is a book
W-1     0.040   seconds
H-1     -1.4000506401062012     این کتاب است .
D-1     -1.4000506401062012     این کتاب است.
P-1     -0.8005 -3.0582 -2.3328 -0.8005 -0.0083
i have car
S-2     i have car
W-2     0.033   seconds
H-2     -1.2344274520874023     من
D-2     -1.2344274520874023     من
P-2     -0.8005 -1.6683
```

بخشی از قسمت interactive machine translation

در اینجا با بررسی این سه جمله مشاهده میکنیم که در جملاتی که احتمالا روی داده های ترین ما وجود نداشته اند مدل کاملا رندوم عمل کرده و حتی تعداد کلماتی که باید نتیجه داده می شده را در نظر نگرفته.

در زیر نیز یکی ازجملات generate شده توسط این مدل را بررسی میکنیم.

```
. خود ، اشاره‌ای به آینده این طرح کرده‌است <<unk>> به نظر می‌رسد مدیرعامل سابق ایل ، استیو جابز در گفتگو با
-3.618614912033081      . " به گزارش داد که در حال حاضر به آسیای میانه ، در حال حاضر به نام ، به نام ، به نام ، به نام
-3.618614912033081      . " به گزارش داد که در حال حاضر به آسیای میانه ، در حال حاضر به نام ، به نام ، به نام ، به نام
```

شکل بالا اهمیت <unk> را در تولید جملات ترجمه نشان میدهد و داده های پیش پردازش شده را به کمک مدل آموزش دیده ترجمه میکند.

نتیجه روی معماری Convolutional:

پس از بررسی چندین مدل از معماری های قابل اجرا برای train این مدل از یک convolutional architecture استفاده میکنیم.
ماژول لیست های این مدل به شرح زیر هستند.

```
(convolutions): ModuleList(
  (0): ConvTBC(256, 512, kernel_size=(3,), padding=(1,))
  (1): ConvTBC(256, 512, kernel_size=(3,), padding=(1,))
  (2): ConvTBC(256, 512, kernel_size=(3,), padding=(1,))
  (3): ConvTBC(256, 512, kernel_size=(3,), padding=(1,))
```

کد زیر تمام پارامترهای این مدل را مقداردهی میکند.

```
!CUDA_VISIBLE_DEVICES=0 fairseq-train /content/data-bin/custom.tokenized.en-fa \
    --optimizer nag --lr 0.01 --clip-norm 0.1 --dropout 0.2 --max-tokens 4000 --max-epoch 30 \
    --arch fconv_iwslt_de_en --save-dir checkpoints/fconv
```

مدلی که در اینجا از آن استفاده میکنیم بر پایه ترجمه آلمانی و انگلیسی است و decoder,encoder آن معماری convolutional دارند. در انتها یک لایه attention نیز در این مدل پیاده سازی شده است.
مقادیر loss و perplexity نیز در اولین epoch به شرح زیر است.

```
| train | epoch 001 | loss 14.714 | ppl 26876.7 | wps 31004.2 | ups 10.68
```

مقادیر loss و perplexity در ۴۰ ام epoch به شرح زیر است.

```
| train | epoch 040 | loss 9.096 | ppl 547.13 | wps 30678.2 | ups 10.6
```

این مقادیر همگی با افزایش تعداد epoch و batch سایز قابل بهبود هستند که به دلیل محدودیت پردازشی قابل انجام نبودند. همچنین تغییر و بهبود مقادیر پس از epoch ۳۰ و ۴۰ بسیار کند صورت میگرفت که نشان از نزدیک شدن به overfit شدن مدل دارد.

پس از آموزش مدل باید به کمک Fairseq-genetare برای این مدل جملات را بر اساس آموزشی که داده ایم تولید کنیم و در نهایت با بررسی بهترین checkpoint از مدل های تولید شده در مراحل مختلف که آخرین epoch است، مقادیر bleu را گزارش کنیم.

```
!fairseq-generate /content/data-bin/custom.tokenized.en-fa \
    --path checkpoints/fconv/checkpoint_best.pt \
    --batch-size 512 --beam 5
```

```
2022-06-09 10:51:14 | INFO | fairseq_cli.generate | NOTE: hypothesis and token scores are output in base 2
2022-06-09 10:51:14 | INFO | fairseq_cli.generate | Translated 427 sentences (11145 tokens) in 2.8s (153.56 sentences/s, 4008.01 tokens/s)
Generate test with beam=5: BLEU4 = 3.10, 28.5/5.9/1.5/0.4 (BP=0.960, ratio=0.961, syslen=10718, reflen=11154)
```

این bleu = ۳٫۱۰ بیشترین مقدار امتیازی است که توانستیم با تغییر دادن پارامترها به دست بیاوریم.

برای بررسی پنج نقطه میانی روند آموزش نیز از مدل‌های ذخیره شده در پوشه checkpoint و به کمک دستور generate برای این مدل ها امتیاز bleu را به دست می آوریم.

کد زیر را برای checkpoint ها با مقادیر متقاوت تکرار میکنیم.

```
!fairseq-generate /content/data-bin/custom.tokenized.en-fa \
    --path checkpoints/fconv/checkpoint8.pt \
    --batch-size 512 --beam 5
```

خروجی به شکل زیر است.

```
2022-06-10 19:35:12 | INFO | fairseq_cli.generate | NOTE:
2022-06-10 19:35:12 | INFO | fairseq_cli.generate | Trans
Generate test with beam=5: BLEU4 = 0.47, 16.2/1.3/0.1/0.0
```

سپس برای بررسی مدل میتوانیم جملات انگلیسی را به این مدل بدهیم و جملات ترجمه شده فارسی را بررسی کنیم. برای چند نمونه به شکل زیر داریم. و مشاهده میکنیم که این مدل برای کلمات به تنهایی بسیار خوب عمل می کند اما برای جملات بلند ناکارآمد است.

```
how are you
S-5     how are you
W-5     0.018   seconds
H-5     -2.5587823390960693     چطور است
D-5     -2.5587823390960693     چطور است.
P-5     -3.6657 -3.1067 -3.4204 -0.0423
why
S-6     why
W-6     0.014   seconds
H-6     -1.6404590606689453     چرا
D-6     -1.6404590606689453     چرا
P-6     -3.1479 -0.1330
how
S-7     how
W-7     0.013   seconds
H-7     -3.0108823776245117     چطور
D-7     -3.0108823776245117     چطور
P-7     -5.4393 -0.5825
please
S-8     please
W-8     0.015   seconds
H-8     -2.4519195556640625     لطفا
D-8     -2.4519195556640625     لطفا
P-8     -4.7233 -0.1805
```

نتایج و تحلیل آزمایشات انجام شده بر روی OpenNMT:

نتیجه روی معماری Transformer:

برای این بخش ۲ مدل را به دقت بررسی میکنیم (این ۲ مدل از همه مواردی که آزمایش کردیم که بیشتر از ۵ مدل مختلف بود بهتر بود):

مدل اول با config زیر است:

```
save_checkpoint_steps: 1000
valid_steps: 1000
train_steps: 16000
world_size: 1
gpu_ranks: 0
batch_size: 4096
batch_type: tokens
normalization: tokens
encoder_type: transformer
decoder_type: transformer
position_encoding: true
param_init_glorot: true
enc_layers: 6
dec_layers: 6
rnn_size: 512
word_vec_size: 512
label_smoothing: 0.1
dropout: 0.1
attention_dropout: 0.1
```

در نتیجه آزمایشاتی که انجام دادیم، با افزایش batch_size و افزایش layerها میتوانیم دقت را بالاتر ببریم. لازم به ذکر است که مقدار پیش فرض لایه‌ها در encoder و decoder در این ابزار برابر ۲ لایه است که در اینجا ما آن را به ۲ تغییر دادیم. Label_smoothing مقدار اپسیلون را مشخص میکند که احتمال لیبل‌های نادرست را به اندازه این اپسیلون smooth میکند؛ برای اطلاعات بیشتر درباره این پارامتر میتوانید به این لینک[۱] مراجعه کنید. Dropout برای جلوگیری از overfitting استفاده شده است. لازم به ذکر است که رد صورت استفاده از transformer برای encoder_type و decoder_type باید ۲ پارامتر word_vec_size و rnn_size را مقدار دهی کنیم. پارامترهای زیاد دیگری مثل warmup_step، accum_count، decay_method و... را هم آزمایش کردیم که به بهبود نتیجه کمکی نکرد.

نتیجه این حالت:

Validation Data:

[۱] https://arxiv.org/pdf/1512.00567.pdf

```
Ckeckpoint 0: 0.0
Ckeckpoint 1: 0.0
Ckeckpoint 2: 0.0
Ckeckpoint 3: 0.0
Ckeckpoint 4: 0.76
Ckeckpoint 5: 0.88
Ckeckpoint 6: 1.45
Ckeckpoint 7: 1.7
Ckeckpoint 8: 1.96
Ckeckpoint 9: 2.0
Ckeckpoint 10: 2.1
Ckeckpoint 11: 2.3
Ckeckpoint 12: 2.72
Ckeckpoint 13: 2.6
Ckeckpoint 14: 2.86
Ckeckpoint 15: 2.74
Ckeckpoint 16: 2.59
```

Figure ۱ نتایج مقدار bleu روی ۱۶ نقطه در طول آموزش روی داده Validation

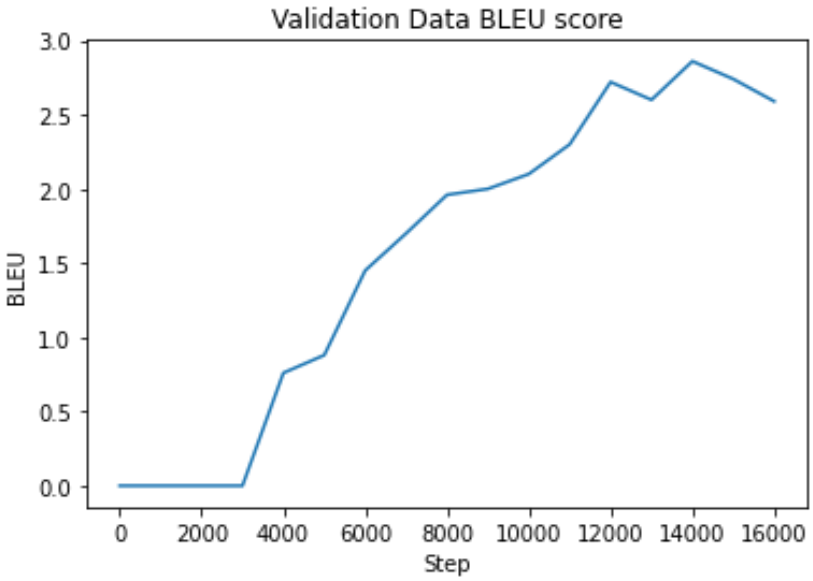

Figure ۲ نتایج مقدارbleu برای ۱۵ نقطه در طول آموزش روی داده Validation

همانطور که دیده می‌شود بهترین نتیجه bleu روی دادگان validation برابر ۲٫۸۶ است که در قدم ۱۴۰۰ رخ داده است.

:Test Data

```
Ckeckpoint 0: 0.0
Ckeckpoint 1: 0.0
Ckeckpoint 2: 0.0
Ckeckpoint 3: 0.2
Ckeckpoint 4: 0.69
Ckeckpoint 5: 0.76
Ckeckpoint 6: 0.89
Ckeckpoint 7: 1.31
Ckeckpoint 8: 1.59
Ckeckpoint 9: 1.42
Ckeckpoint 10: 1.7
Ckeckpoint 11: 1.89
Ckeckpoint 12: 2.26
Ckeckpoint 13: 2.15
Ckeckpoint 14: 2.36
Ckeckpoint 15: 2.08
Ckeckpoint 16: 2.11
```

Figure ۳ نتایج مقدار bleu روی ۱۶ نقطه در طول آموزش روی داده Test

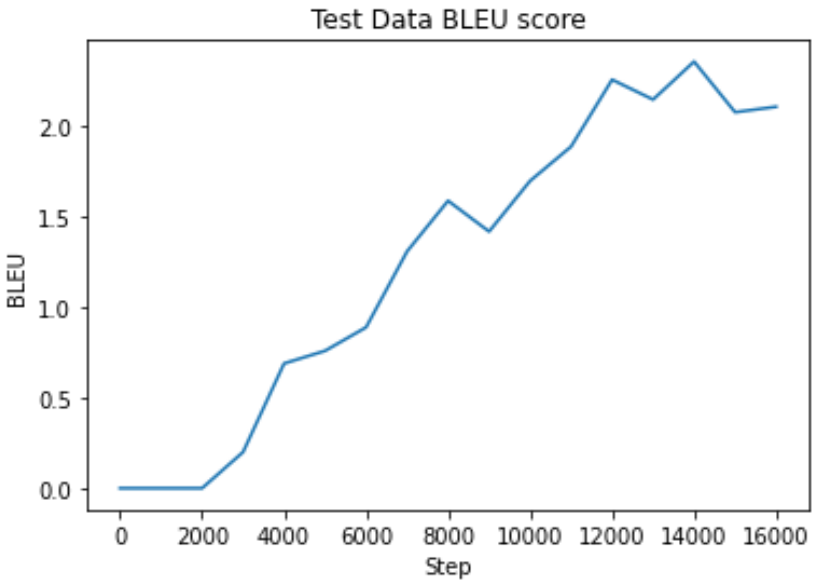

Figure ۴ نتایج مقدارbleu برای ۱۵ نقطه در طول آموزش روی داده Test

همانطور که دیده می‌شود بهترین مقدار bleu روی نقطه ۴ با مقدار bleu برابر ۲٫۳۶ اتفاق افتاده است.

حال ترجمه یکی از جملات داده تست (برای نمونه خط ۴۰۴ داده تست) را در نظر گرفته و برای ۳ نقطه مختلف ۳ و ۱۰ و ۱۴ (یعنی قدم‌های ۳۰۰۰ و ۱۰۰۰۰ و ۱۴۰۰۰) نتیجه را میبینیم (نتایج ترجمه همه جمله‌ها در فایل notebook قابل مشاهده است):
نتیجه نقطه‌ی ۳ که مقدار bleu برابر ۰٫۲ است:

```
[2022-06-09 17:23:23,658 INFO]
SENT 404: ['this', 'is', 'likely', 'to', 'have', 'been', 'a', 'single', 'shot', 'intervention', '.']
PRED 404: این است .
```

نتیجه نقطه‌ی ۱۰ که مقدار bleu برابر ۱٫۷ است:

```
[2022-06-09 17:25:53,977 INFO]
SENT 404: ['this', 'is', 'likely', 'to', 'have', 'been', 'a', 'single', 'shot', 'intervention', '.']
PRED 404: این احتمال ممکن است منجر به ایجاد یک رکود عمومی در اختیار می‌گذارد .
```

نتیجه نقطه‌ی ۱۴ که مقدار bleu برابر ۲٫۳۶ است:

```
[2022-06-09 17:28:31,491 INFO]
SENT 404: ['this', 'is', 'likely', 'to', 'have', 'been', 'a', 'single', 'shot', 'intervention', '.']
PRED 404: این احتمال وجود دارد که احتمالا آن‌ها یک شف@@ اعت داشته باشد .
```

طبق چیزی که انتظار داشتیم، ترجمه با نقطه‌ای که بیشترین مقدارbleu را داشته بهتر از ۲ نقطه دیگر است. ترجمه نقطه‌ی ۱۴ دقیق است. اما مثلا برای نقطه چون آموزش به تازگی شروع شده مدل قادر به ترجمه نیست و صرفا "این است" ترجمه کرده که اشتباه است و احتمالا برای ترجمه this is بوده که در داده آموزش زیاد تکرار شده است.

مدل دوم با config زیر است:

```
save_checkpoint_steps: 1000
valid_steps: 1000
train_steps: 35000
world_size: 1
gpu_ranks: 0
batch_size: 4096
batch_type: tokens
normalization: tokens
encoder_type: transformer
decoder_type: transformer
position_encoding: true
param_init_glorot: true
enc_layers: 6
dec_layers: 6
param_init: 0
rnn_size: 512
word_vec_size: 512
label_smoothing: 0.1
dropout: 0.1
attention_dropout: 0.1
```

نتیجه این حالت:
:Validation Data

```
Checkpoint 0: 0.0
Checkpoint 1: 0.0
Checkpoint 2: 0.0
Checkpoint 3: 0.0
Checkpoint 4: 0.0
Checkpoint 5: 0.0
Checkpoint 6: 0.71
Checkpoint 7: 0.94
Checkpoint 8: 1.7
Checkpoint 9: 0.0
Checkpoint 10: 0.85
Checkpoint 11: 0.96
Checkpoint 12: 0.79
Checkpoint 13: 0.99
Checkpoint 14: 1.09
Checkpoint 15: 1.35
Checkpoint 16: 1.54
Checkpoint 17: 1.24
Checkpoint 18: 1.8
Checkpoint 19: 1.16
Checkpoint 20: 1.37
Checkpoint 21: 1.53
Checkpoint 22: 1.34
```

Figure 5 نتایج مقدار bleu روی ۲۲ نقطه در طول آموزش روی داده Validation

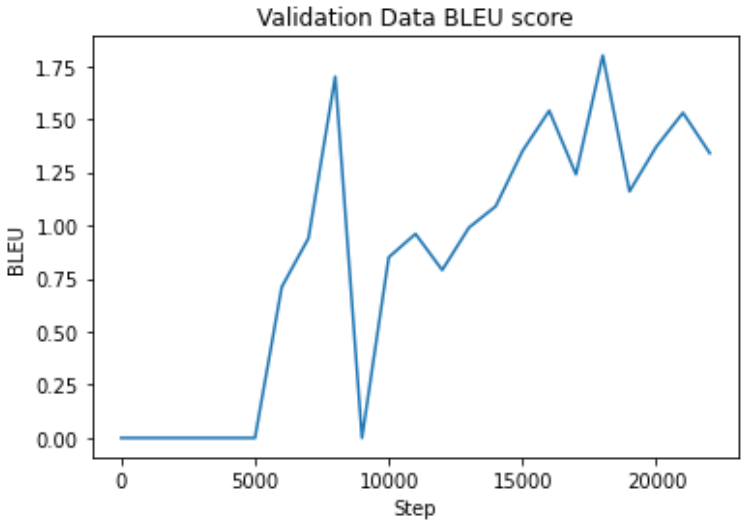

Figure ۶ نتایج مقدارbleu برای ۱۵ نقطه در طول آموزش روی داده Validation

همانطور که دیده می‌شود بهترین نتیجه bleu روی دادگان validation برابر 1.54است که در قدم ۱۴۰۰ رخ داده است.

Test Data:

```
Checkpoint 0: 0.0
Checkpoint 1: 0.0
Checkpoint 2: 0.0
Checkpoint 3: 0.0
Checkpoint 4: 0.18
Checkpoint 5: 0.34
Checkpoint 6: 0.33
Checkpoint 7: 0.0
Checkpoint 8: 0.81
Checkpoint 9: 0.56
Checkpoint 10: 0.28
Checkpoint 11: 0.82
Checkpoint 12: 0.7
Checkpoint 13: 0.46
Checkpoint 14: 0.69
Checkpoint 15: 0.73
Checkpoint 16: 1.04
Checkpoint 17: 0.89
Checkpoint 18: 0.98
Checkpoint 19: 0.94
Checkpoint 20: 1.03
Checkpoint 21: 0.9
Checkpoint 22: 0.86
```

Figure ۷ نتایج مقدار bleu روی ۲۲ نقطه در طول آموزش روی داده Test

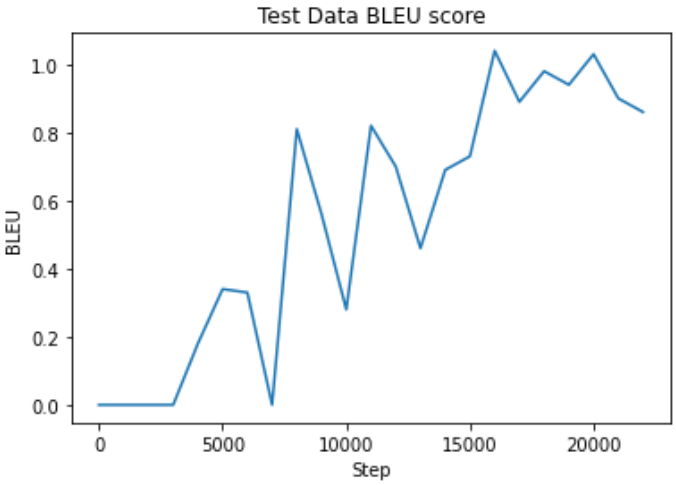

Figure ۸ نتایج مقدار bleu روی ۲۲ نقطه در طول آموزش روی داده Test

همانطور که دیده می‌شود بهترین مقدار bleu روی نقطه ۱۶ با مقدار bleu برابر ۱٫۰۴ اتفاق افتاده است که از مدل قبلی بدتر است. از تفاوت این ۲ مدل که با معماری transformer بودند، میتوان به پارامتر param_init اشاره کرد که در این مدل له صفر مقداردهی شده بود که طبق documentation ابزار openNMT یعنی هیچ initializationای صورت نگیرد؛ اما در مدل قبلی برای این پارامتر مقدار پیش فرض آن یعنی ۰٫۱ در نظر گرفته شده و به نتیجه بهتر مدل کمک کرده است.

حال ترجمه یکی از جملات داده تست (برای نمونه خط ۴۰۴ داده تست) را در نظر گرفته و برای ۲ نقطه مختلف۱۰ و ۱۶ (یعنی قدم‌های ۱۰۰۰۰ و ۱۶۰۰۰) نتیجه را میبینیم:
نتیجه نقطه‌ی ۱۰ که مقدار bleu برابر ۰٫۲۸ است:

```
[2022-06-09 20:24:59,929 INFO]
SENT 412: ['afghan', 'president', 'hamid', 'karzai', 'offered', 'his', 'sympathies', 'at', 'a', 'national', 'security', 'meeting', 'sunday', '.']
PRED 412: رئیس‌جمهور صالح در روز 14 اوت در یک مراسم افتتا@@ جیه ملی در خارج شد .
```

نتیجه نقطه‌ی ۱۶ که مقدار bleu برابر ۱٫۰۴ است:

```
[2022-06-09 20:27:37,247 INFO]
SENT 412: ['afghan', 'president', 'hamid', 'karzai', 'offered', 'his', 'sympathies', 'at', 'a', 'national', 'security', 'meeting', 'sunday', '.']
PRED 412: حامد کرزی رئیس‌جمهور افغانستان راست در طی مراسم افتتا@@ جیه شورای عالی صلح در روز شنبه اعلام کرد .
```

طبق چیزی که انتظار داشتیم، ترجمه با نقطه ۱۶ که بیشترین مقدارbleu را داشته از نقطه دیگر که مقدار bleu پایین‌تری دارد، بهتر شده است. ترجمه نقطه‌ی ۱۶ خوب است اما برای مثال به این اشاره کرد که Sunday به درستی یکشنبه ترجمه نشده و شنبه ترجمه شده است. البته تشخیص اینکه مدل یک روزی از روزهای هفته را برای ترجمه این کلمه در نظر گرفته هم خوب است اما به دلیل کمبود داده‌های آموزش و نداشتن منابع سخت‌افزاری خوب، این نتیجه قابل توجیه و قابل قبول است.

ما آزمایشات زیادی برای بهتر شدن نتیجه انجام دادیم که یکی دیگر از این نتایج را به اختصار برای آخرین checkpoint می‌آوریم:
معماری:

```
save_checkpoint_steps: 1000
valid_steps: 1000
train_steps: 8000

world_size: 1
gpu_ranks: 0

param_init_glorot: true
encoder_type: transformer
decoder_type: transformer
position_encoding: true
rnn_size: 512
word_vec_size: 512
label_smoothing: 0.1
```

نتیجه:

```
BLEU = 0.35, 6.4/1.0/0.1/0.0 (BP=1.000, ratio=2.593, hyp_len=29519, ref_len=11385)
```

نتیجه روی معماری RNN:

طبق حالت‌های خیلی زیادی که روی این معماری آزمایش کردیم، بهترین نتیجه وقتی رخ میداد که پارامترها را به طور پیش فرض قرار میدادیم و تنها مقدار train_steps را کم کردیم چون با آزمایش‌های انجام شده بعد از مدتی دقت ثابت مانده و بهبود نمیافت. مقدار پیش فرض برای این پارامتر برابر ۱۰۰۰۰۰ است که ما ۳۰۰۰۰ قرار دادیم.

نتیجه روی بهترین حالت بدست آمده:

Validation Data:

```
Checkpoint 0: 0.0
Checkpoint 1: 1.08
Checkpoint 2: 2.99
Checkpoint 3: 4.7
Checkpoint 4: 5.32
Checkpoint 5: 4.29
Checkpoint 6: 4.15
Checkpoint 7: 4.13
Checkpoint 8: 4.71
Checkpoint 9: 4.49
Checkpoint 10: 4.01
```

Figure ۹ نتایج مقدار bleu روی ۱۰ نقطه در طول آموزش روی داده validation

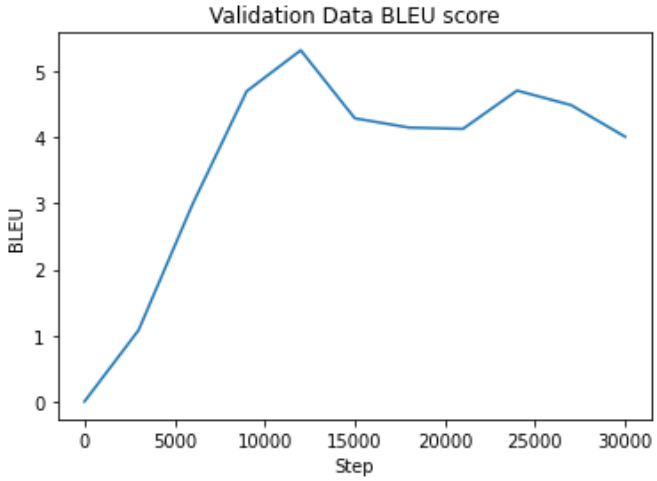

Figure ۱۰ نتایج مقدار bleu برای ۱۰ نقطه در طول آموزش روی داده validation

همانطور که دیده می‌شود بهترین نتیجه bleu روی دادگان validation برابر ۵٫۳۲ است که در قدم ۱۲۰۰ رخ داده است و بعد از مقدار bleu افت کرده است که میتواند نشان overfit شدن باشد.

:Test Data

```
Checkpoint 0: 0.0
Checkpoint 1: 0.65
Checkpoint 2: 3.26
Checkpoint 3: 4.71
Checkpoint 4: 4.56
Checkpoint 5: 4.42
Checkpoint 6: 4.31
Checkpoint 7: 3.77
Checkpoint 8: 4.51
Checkpoint 9: 4.52
Checkpoint 10: 3.91
```

Figure ۱۱ نتایج مقدار bleu روی ۱۰ نقطه در طول آموزش روی داده Test

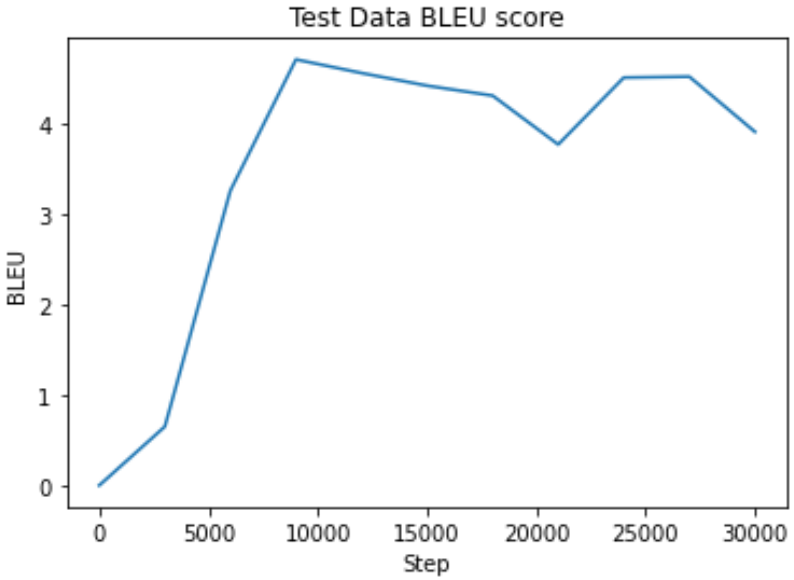

Figure ۱۲ نتایج مقدارbleu برای ۱۰ نقطه در طول آموزش روی داده Test

همانطور که دیده می‌شود بهترین مقدار bleu روی نقطه ۴ با مقدار bleu برابر ۴/۵۶ اتفاق افتاده است.

حال ترجمه یکی از جملات داده تست (برای نمونه خط ۳۲۴ داده تست) را در نظر گرفته و برای ۳ نقطه مختلف ۱ و ۵ و ۱۰ (یعنی قدم‌های ۳۰۰۰ و ۱۵۰۰۰ و ۳۰۰۰۰) نتیجه را میبینیم:
نتیجه نقطه‌ی ۱ که مقدار bleu برابر ۰/۶۵ است:

```
[2022-06-09 16:13:11,775 INFO]
SENT 324: ['and', 'as', 'the', 'winds', 'of', 'change', 'sweep', 'the', 'arab', 'world', ',', 'they', 'seem', 'to', 'stop', 'at', 'iran', "'s", 'borders', '.']
PRED 324: . و با این حال ، این امر را به عنوان این امر به عنوان این امر به کار می‌آورد
```

نتیجه نقطه‌ی ۵ که مقدار bleu برار ۴/۴۲ است:

```
[2022-06-09 16:39:31,544 INFO]
SENT 324: ['and', 'as', 'the', 'winds', 'of', 'change', 'sweep', 'the', 'arab', 'world', ',', 'they', 'seem', 'to', 'stop', 'at', 'iran', "'s", 'borders', '.']
PRED 324: . و با توجه به شدت تغییر این دنیای عرب را دارند ، در حالی که در حال حاضر کردن در مرزهای ایران هستند
```

نتیجه نقطه‌ی ۱۰ که مقدار bleu برابر ۳/۹۱ است:

```
[2022-06-09 16:40:25,893 INFO]
SENT 324: ['and', 'as', 'the', 'winds', 'of', 'change', 'sweep', 'the', 'arab', 'world', ',', 'they', 'seem', 'to', 'stop', 'at', 'iran', "'s", 'borders', '.']
PRED 324: . و در همان حال ایروی تج@@ @@ فی@@ فی عرب به دنبال مرزهای ایران هستند
```

مطابق انتظار ترجمه خروجی قدم ۵ که امیتاز bleu بیشتری دارد بهتر از ۲ نقطه دیگر است و میتوان بهبود در نتیجه ترجمه را با افزایش قدم‌ها مشاهده کرد.

نتیجه نهایی مقدار blue روی آخرین checkpoint را برای چند حالت دیگر روی این معماری به طور خلاصه نشان می‌دهیم:
پارامترها:

```
!python OpenNMT-py/train.py -data BPE_Model -save_model Train_With_BPE_Model -world_size 1 -gpu_rank 0 -train_steps 30000 -valid_steps 1000 -save_checkpoint_steps 3000 -dropout 0.2

[2022-06-06 21:57:09,671 INFO] Loading dataset from BPE_Model.train.0.pt
[2022-06-06 21:57:10,640 INFO] number of examples: 28523
[2022-06-06 21:57:10,818 INFO] Step 28100/30000; acc:  71.11; ppl:  3.18; xent: 1.16; lr: 1.00000; 13316/14210 tok/s;   1909 sec
[2022-06-06 21:57:13,897 INFO] Step 28150/30000; acc:  72.74; ppl:  2.93; xent: 1.07; lr: 1.00000; 17074/18274 tok/s;   1912 sec
```

نتیجه:

```
BLEU = 3.05, 34.2/6.8/1.9/0.4 (BP=0.836, ratio=0.848, hyp_len=9656, ref_len=11385)
```

پارامترها:

```
!python OpenNMT-py/train.py -data BPE_Model -save_model Train_With_BPE_Model -world_size 1 -gpu_rank 0 -train_steps 30000 -valid_steps 1000 -save_checkpoint_steps 3000

[2022-06-07 13:47:59,420 INFO] Loading dataset from BPE_Model.train.0.pt
[2022-06-07 13:48:00,381 INFO] number of examples: 28501
[2022-06-07 13:48:00,556 INFO] Step 28100/30000; acc:  65.68; ppl:  4.23; xent: 1.44; lr: 1.00000; 12788/13836 tok/s;   1895 sec
[2022-06-07 13:48:03,781 INFO] Step 28150/30000; acc:  63.88; ppl:  4.74; xent: 1.56; lr: 1.00000; 18032/18488 tok/s;   1898 sec
```

نتیجه:

```
BLEU = 3.75, 30.3/6.7/2.0/0.5 (BP=0.992, ratio=0.992, hyp_len=11298, ref_len=11385)
```

۹. ارزیابی یک ابزار تولید ترجمه ماشین باید در دو بعد وفاداری (Fidelity) و روان بودن (Fluency) باید باشد. توافق خاصی بین مترجم‌ها وجود ندارد چون این مسئله subjective است (یعنی بستگی دارد که موضوع ترجمه چیست و چه ابزاری را مورد استفاده قرار دادیم).

ساده ترین روش ارزیابی به صورت human rater است. این کار خیلی هزینه‌بر و البته دقیق است. هم Fidelity و هم Fluency مورد ارزیابی دقیق قرار میگیرد.

برای ارزیابی Fluency متن ترجمه شده را به انسان میدهند و از او سوالاتی از قبیل اینکه clear و natural و intelligible و readable هست می‌پرسند و نمره میدهند. لازم به ذکر است که برای این نمره دهی یا عددی مثلا بین ۱ تا ۵ میدهند و یا از cloze test استفاده می‌کنند.

برای ارزیابی Fluency سه روش را اشاره میکنیم: ۱- هر دو متن مبدا و مقصد را به انسان میدهند و او نمره‌ای مثلا بین ۱ تا ۵ از نظر روان بودن میدهد. ۲- فقط متن ترجمه شده را به انسان میدهند و سوالات درک مطلب (comprehension) از او میپرسند و اینکه او میتواند

به این سوالات پاسخ بدهد یا خیر معیاری برای ارزیابی fluency است. ۳- هر دو متن مبدا و مقصد به انسان داده میشود و او باید متن ترجمه شده از طریق ماشین را post-edit کند. زمانی که طول میشکد تا این عملیات انجام شود معیاری برای تشخیص روان‌بودن متن است.

همه روش های ذکر شده که از طرف انسان امجام میود بسیار هزینه‌بر و زمان‌بر است. روش‌های اوتوماتیک وجود دارد که در ادامه به توضیح بعضی از آن ها میپردازیم. روش های heuristic مثل: Bleu، NIST، TER، Precision and Recall و METEOR. معیار همه این روش‌ها این است که ارزیابی MT خوب آن است که به ارزیابی انسان نزدیک‌تر باشد(Translation Closeness). در واقع زبان مبدا را به چند نفر انسان میدهند تا ترجمه کنند(رفرنس‌های ۱ تا n که gold data هم میگویند) حال این متن مبدا را به ماشین هم میدهند تا ترجمه کند. از شباهت ترجمه ماشین با gold data ها میتوان معیارها را حساب کرد. از معروف‌ترین روش‌های اوتومایتک BLEU است که در این پروژه هم ارزیابی بر اساس همین معیار خواسته شده است. این روش در واقع یک weighted average از N-gramهایی است که با ترجمه انسان مشترک است(overlap دارد). فرمول این روش به صورت زیر است که توضیحات دقیق‌تر این فرمول در کلاس گفته شد:

$$BP = \begin{cases} 1 & \text{if } c > r \\ e^{(1-r/c)} & \text{if } c \leq r \end{cases}$$

$$\text{Bleu} = \text{BP} \times \exp\left(\frac{1}{N}\sum_{n=1}^{N} \log p_n\right)$$