درس : NLP

استاد: آقای دکتر هشام فیلی

دانشجوی مهمان : رقیه یزدانی

پروژه CA شماره 6

پروژه آموزش مترجم ماشینی مبتنی بر شبکه عصبی

در این پروژه، از OpenNMT-py که ابزاری open-source برای ترجمه ماشینی مبتنی بر شبکه عصبی می‌باشد، استفاده شده و سعی می‌شود یک مترجم ماشینی انگلیسی به فارسی و همچنین یک مترجم ماشینی برای نویسه‌گردانی فارسی به انگلیسی آموزش داده شود.

بخش اول – مترجم ماشینی انگلیسی به فارسی

برای این بخش، فایلهای train.en و train.fa برای مرحله آموزش و فایلهای dev.en و dev.fa بعنوان development set و فایل test.en به همراه چهار ترجمه مرجع test.fa0 و test.fa1 و test.fa2 و test.fa3 برای ارزیابی ترجمه ماشینی آموزش داده شده در اختیار قرار گرفته است. این فایلها sentence-level aligned هستند و مشخصات کلی آنها در جدول زیر ذکر شده است :

فایلها	تعداد سطر (جمله)
train.en و train.fa	26،146
dev.en و dev.fa	277
test.en و test.fa0 و test.fa1 و test.fa2 و test.fa3	251

ابتدا با استفاده از دستورات زیر، نصب های لازم برای امکان استفاده از OpenNMT-py انجام شد :

```
conda install -c anaconda mkl
conda install -c pytorch pytorch torchvision
pip install OpenNMT-py
```

سپس پروژه متن‌باز OpenNMT-py از GitHub دانلود شده و فایلهای مراحل train,dev,test به مسیر data/ En2Fa_Translation از آن منتقل شد.

سوال 1 – آموزش سیستم ترجمه مبتنی برRNN بدون استفاده از BPE

برای انجام مرحله پیش‌پردازش، دستور زیر اجرا شد. دستورات برای خواناتر شدن در این گزارش در چند سطر نشان داده شده است.

```
onmt_preprocess
        -train_src data/En2Fa_Translation/Train/train.en
        -train_tgt data/En2Fa_Translation/Train/train.fa
        -valid_src data/En2Fa_Translation/Dev/dev.en
        -valid_tgt data/En2Fa_Translation/Dev/dev.fa
        -save_data data/En2Fa_Translation/en2fa-preprocess
```

پس از اجرای دستور بالا، فایلهای زیر در مسیر تعیین شده در پارامتر save_data- دستور بالا ایجاد شد:

```
data/En2Fa_Translation/ en2fa-preprocess.train.0.pt
data/En2Fa_Translation/ en2fa-preprocess.valid.0.pt
data/En2Fa_Translation/ en2fa-preprocess.vocab.pt
```

سپس با استفاده از دستور زیر، مرحله آموزش مدل صورت گرفت :

```
python train.py
        -world_size 1
        -gpu_ranks 0
        -data data/En2Fa_Translation/en2fa-preprocess
        -train_steps 50000
        -save_checkpoint_steps 1000
        -save_model data/En2Fa_Translation/en2fa-model
```

دو پارامتر اول برای تنظیم استفاده از GPU می‌باشد، پارامتر بعدی، نام و مسیر فایلهای خروجی مرحله پیش‌پردازش را تعیین می‌کند. تعداد epoch ها، 50000 تعیین شده و بعد از هر 1000 تکرار، مدل در مسیر مشخص شده؛ ذخیره خواهد شد.

اجرای دستور بالا، 51 دقیقه طول کشید و در نهایت 50 مدل میانی با نامهای زیر ایجاد گردید. سایز هر مدل؛ 67MB بوده.

```
en2fa-model_step_1000.pt
en2fa-model_step_2000.pt
…
en2fa-model_step_50000.pt
```

در مرحله بعد دستور زیر برای ترجمه فایل test.en بر اساس آخرین مدل آموزش دیده، اجرا شد :

```
python translate.py -model data/En2Fa_Translation/en2fa-model_step_50000.pt
        -src data/En2Fa_Translation/Test/test.en
```

-output data/En2Fa_Translation/pred_50000.txt
-replace_unk
-verbose

الف)

دستور زیر برای ارزیابی ترجمه صورت گرفته، اجرا شد : (البته قبل از اجرا، Perl نصب گردید)

perl tools/multi-bleu.perl
data/En2Fa_Translation/Test/test.fa < data/En2Fa_Translation/pred_50000.txt

خروجی آن در زیر نشان داده شده، ابتدا معیار BLEU برای ارزیابی ترجمه نهایی بر اساس چهار جمله مرجع محاسبه شد و سپس بر اساس هر جمله مرجع، به تنهایی هم مقدار BLEU بدست آمد :

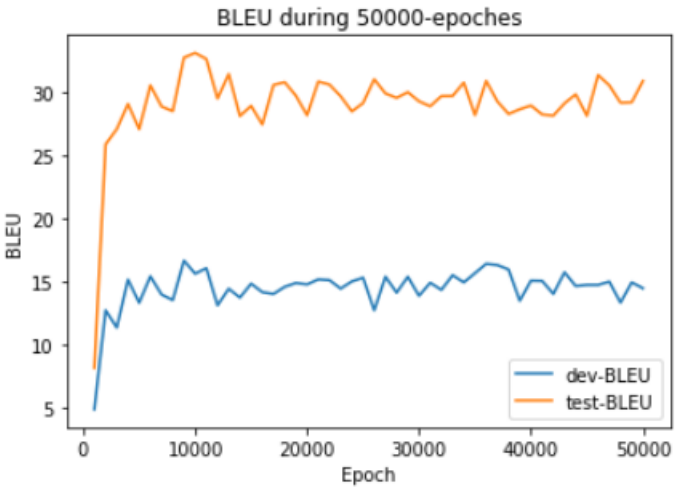
```
(py3-TF2.0) F:\Projects\OpenNMT-py-master>perl  tools/multi-bleu.perl data/En2Fa_Translation/Test/test.fa < data/En2Fa_T
ranslation/pred_50000.txt
BLEU = 30.83, 72.7/40.5/24.8/14.3 (BP=0.964, ratio=0.965, hyp_len=2476, ref_len=2567)

(py3-TF2.0) F:\Projects\OpenNMT-py-master>perl  tools/multi-bleu.perl data/En2Fa_Translation/Test/test.fa0 < data/En2Fa_
Translation/pred_50000.txt
BLEU = 18.93, 59.6/27.7/14.7/7.5 (BP=0.916, ratio=0.920, hyp_len=2476, ref_len=2692)

(py3-TF2.0) F:\Projects\OpenNMT-py-master>perl  tools/multi-bleu.perl data/En2Fa_Translation/Test/test.fa1 < data/En2Fa_
Translation/pred_50000.txt
BLEU = 20.45, 60.9/29.6/16.4/8.7 (BP=0.909, ratio=0.913, hyp_len=2476, ref_len=2713)

(py3-TF2.0) F:\Projects\OpenNMT-py-master>perl  tools/multi-bleu.perl data/En2Fa_Translation/Test/test.fa2 < data/En2Fa_
Translation/pred_50000.txt
BLEU = 14.53, 53.6/21.8/10.8/5.1 (BP=0.914, ratio=0.917, hyp_len=2476, ref_len=2700)

(py3-TF2.0) F:\Projects\OpenNMT-py-master>perl  tools/multi-bleu.perl data/En2Fa_Translation/Test/test.fa3 < data/En2Fa_
Translation/pred_50000.txt
BLEU = 15.32, 52.6/22.2/10.6/5.2 (BP=0.961, ratio=0.962, hyp_len=2476, ref_len=2575)

(py3-TF2.0) F:\Projects\OpenNMT-py-master>
```

همانطور نتیجه ارزیابی براساس هر چهار ترجمه مرجع، 30.83 و کمترین مقدار آن بر اساس ترجمه مرجع سوم با مقدار 14.53 می‌باشد.

ب)

برای این بخش، مشابه بالا، برنامه translate.py به ازای هر یک از 50 مدل ذخیره شده، اجرا شده و سپس برای فایل pred_????.txt هر مرحله مشابه مرحله قبل، معیار BLEU بدست آمد که نمودار آن بصورت زیر می‌باشد. این قسمت هم بر روی مجموعه dev و هم برای مجموعه test انجام شد :

Epoch	BLEU-dev	BLEU-test
1000	4.91	8.18
2000	12.73	25.83
3000	11.36	27.04
4000	15.15	29.02
5000	13.32	27.03
6000	15.41	30.49
7000	13.97	28.82
8000	13.53	28.45
9000	16.65	32.68
10000	15.62	33.05
11000	16.05	32.57
12000	13.12	29.45
13000	14.43	31.38
14000	13.73	28.07
15000	14.83	28.87
16000	14.16	27.4
17000	14.02	30.52
18000	14.58	30.74
19000	14.88	29.67
20000	14.77	28.13
21000	15.17	30.79
22000	15.11	30.55
23000	14.44	29.63
24000	15.01	28.43
25000	15.3	29.09
26000	12.73	30.97
27000	15.38	29.85
28000	14.12	29.5
29000	15.38	29.95
30000	13.87	29.23
31000	14.9	28.84
32000	14.34	29.64
33000	15.5	29.65
34000	14.93	30.71
35000	15.66	28.14
36000	16.4	30.84
37000	16.3	29.22
38000	15.95	28.23
39000	13.48	28.6
40000	15.07	28.9
41000	15.05	28.19
42000	14.02	28.1
43000	15.73	29.07
44000	14.64	29.78
45000	14.73	28.09
46000	14.73	31.3
47000	14.99	30.49
48000	13.33	29.12
49000	14.93	29.15
50000	14.47	30.83



همانطور که از شکل پیداست، بهترین معیار برای 10000 تکرار است و پس از آن، معیار BLEU دائماً نوسان داشته و بهبود چشمگیری مشاهده نمی شود.

پ)

بدلیل حجم پایین داده‌های آموزشی، کیفیت ترجمه ماشینی صورت گرفته؛ مناسب نیست و تفاوت ظاهری و معنایی زیادی بین ترجمه ماشینی بدست آمده با ترجمه های مرجع وجود دارد. از جمله موارد زیر :

نمونه 1 :

we can take a taxi from the station to the hotel.

جملات مرجع :

- ✓ ما میتوانیم از ایستگاه تا هتل یک تاکسی بگیریم .
- ✓ ما میتوانیم از ایستگاه تا هتل یک تاکسی بگیریم .
- ✓ ما میتوانیم یک تاکسی از اینجا تا هتل بگیریم .
- ✓ ما میتوانیم از ایستگاه به هتل تاکسی بگیریم .

نتیجه ترجمه ماشینی :

- ✓ ما میتوانیم از ایستگاه central از ایستگاه تا در هتل .

یکی از مشکلات ترجمه بالا، اشتباه بودن جمله از نظر نحوی و ساختاری است که شاید بتوان، با افزودن بخشی برای اصلاح ساختاری جمله، تا حدی این مشکل را برطرف کرد.

نمونه 2 :

yes . we have two rooms at the Gr"unschnabel . I will reserve a taxi right now .

جملات مرجع :

- ✓ بله . ما دو اتاق در گرانشابل داریم . من الان یک تاکسی رزرو میکنم .
- ✓ بله . ما دو اتاق در گریچنبل داریم . من همین الان یک تاکسی رزرو میکنم .
- ✓ بله . ما دو اتاق در گراند-چینابل داریم .من همین الان یک تاکسی میگیرم .
- ✓ بله . ما دو اتاق در گرانشنبل داریم . من الان یک تاکسی رزرو خواهم کرد .

نتیجه ترجمه ماشینی :

- ✓ بله . ما باید دو اتاق در ورودی تنیس در رستوران رزرو خواهم کرد .

یکی از مشکلاتی در ترجمه های بالا وجود دارد، مربوط به اسامی خاص است، که در اینجا، نام هتل، در هر یک از چهار ترجمه مرجع، به شکل متفاوتی ظاهر شده است و در ترجمه ماشینی، اصلا نیامده است. می‌توان؛ زمان ایجاد داده های آموزشی و ترجمه های مرجع، برای اسامی خاص، از ترجمه ثابتی استفاده کرد، تا هنگام آموزش، مدل به درستی آن را یاد بگیرد.

ت)

پارامتر -replace_unk در زمان ترجمه بدین معناست که کلمات unknown یا همان OOV (Out Of Vocabulart) در ترجمه به صورت اصلی خود آورده شوند و حذف نشوند. برای مدل نهایی مرحله الف در بالا، مجددا برنامه translate اجرا شد و این‌بار پارامتر -replace_unk استفاده نشد، اما در فایل pred تولید شده، تفاوتی با حالت قبل مشاهده نشد!!

```
python translate.py -model data/En2Fa_Translation/en2fa-model_step_50000.pt
        -src data/En2Fa_Translation/Test/test.en
        -output data/En2Fa_Translation/pred_50000_wo_unk.txt
        -verbose
```

ث)

-word_vec_size : سایز بردار embedding را برای source و target تعیین می کند و می تواند برای هریک بصورت جدا از طریق پارامترهای -src_word_vec_size و -tgt_word_vec_size تنظیم شود. مقدار پیش فرض 500 است.

-encoder_type : نوع انکدر را مشخص کرده و می تواند یکی از مقادیر rnn, brnn, ggnn, mean, transformer, cnn باشد و پیش فرض rnn است.

-decoder_type : نوع دیکدر را مشخص کرده و می تواند یکی از مقادیر rnn, transformer, cnn باشد و پیش فرض rnn است.

-layers : تعداد لایه ها در انکدر و دیکدر را مشخص می‌کند. می‌تواند تعداد لایه ها برای انکدر و دیکدر متفاوت باشد که از طریق دو پارامتر -enc_layers و -dec_layers قابل تنظیم است. مقدار پیش فرض 2 است.

-rnn_size : سایز state در rnn انکدر و دیکدر است که می‌تواند برای هر یک بصورت جدا از طریق دو پارامتر -enc_rnn_size و -dec_rnn_size تنظیم شود. مقدار پیش فرض 500 است.

-batch_size : مقدار پیش فرض آن 64 است.

ج)

از پارامترهای مرحله قبل، تعدادی از آنها بصورت زیر نسبت به مرحله قبل، تغییر داده شد:

```
python train.py
        -world_size 1
        -gpu_ranks 0
        -data data/En2Fa_Translation/en2fa-preprocess
        -encoder_type brnn
        -rnn_size 1000
        -batch_size 32
        -train_steps 50000
        -save_checkpoint_steps 1000
        -save_model data/En2Fa_Translation/en2fa-model-j
```

زمان اجرا برای 22000 ایپاک 50 دقیقه طول کشید که نسبت به سری قبل طولانی تر بود و این به دلیل سایز بزرگتر مدل و در نتیجه محاسبات بیشتر برای یادگیری پارامترهای بیشتر بود و در نهایت مقدار BLEU برابر 28.60 برای چهار مرجع بدست آمد که تفاوت چندانی با قبل ندارد.

سوال 1 – آموزش سیستم ترجمه مبتنی بر RNN با استفاده از BPE

برای این بخش لازم است که ابتدا، براساس مجموعه train برای train.en و train.fa کدهای BPE را استخراج کنیم که این کار با استفاده از دستورات زیر انجام شد :

```
python tools/learn_bpe.py
        -i data/En2Fa_Translation/Train/train.en
        -o data/En2Fa_Translation/Train/train_src.code
        -s 4000
python tools/learn_bpe.py
```

-i data/En2Fa_Translation/Train/train.fa
-o data/En2Fa_Translation/Train/train_tgt.code
-s 5000

در دستورات بالا، ابتدا برای پارامتر s- که تعداد symbol ها را تعیین می‌کند، مقدار 10000 درنظر گرفته شد، که به دلیل حجم کوچک داده‌های آموزشی خطای زیر تولید می‌شد و هر بار از مقدار آن کم شد تا درنهایت با مقادیر بالا، کدهای bpe تولید شدند.

error : no pair has frequency >= 2. Stopping

سپس این کدهای bpe که از فایل train.en استخراج شده بودند، روی فایلهای train.en و dev.en و test.en اعمال شد :

python tools/apply_bpe.py -c data/En2Fa_Translation/Train/train_src.code -i data/En2Fa_Translation/Train/train.en -o data/En2Fa_Translation/Train/train_src_bpe.txt

python tools/apply_bpe.py -c data/En2Fa_Translation/Train/train_src.code -i data/En2Fa_Translation/Dev/dev.en -o data/En2Fa_Translation/Dev/dev_src_bpe.txt

python tools/apply_bpe.py -c data/En2Fa_Translation/Train/train_src.code -i data/En2Fa_Translation/Test/test.en -o data/En2Fa_Translation/Test/test_src_bpe.txt

همچنین کدهای bpe که از فایل train.fa استخراج شده بودند، روی فایلهای train.fa و dev.fa اعمال شد:

python tools/apply_bpe.py -c data/En2Fa_Translation/Train/train_tgt.code -i data/En2Fa_Translation/Train/train.fa -o data/En2Fa_Translation/Train/train_tgt_bpe.txt

python tools/apply_bpe.py -c data/En2Fa_Translation/Train/train_tgt.code -i data/En2Fa_Translation/Dev/dev.fa -o data/En2Fa_Translation/Dev/dev_tgt_bpe.txt

حال که مجموعه های train و dev و test با استفاده از bpe مجدداً بازتولید شدند، می توان مشابه سوال 1، مراحل پیش پردازش، آموزش، ترجمه و ارزیابی را انجام داد. ابتدا مرحله پیش پردازش :

python preprocess.py
-train_src data/En2Fa_Translation/Train/train_src_bpe.txt
-train_tgt data/En2Fa_Translation/Train/train_tgt_bpe.txt
-valid_src data/En2Fa_Translation/Dev/dev_src_bpe.txt
-valid_tgt data/En2Fa_Translation/Dev/dev_tgt_bpe.txt
-save_data data/En2Fa_Translation/en2fa-bpe

و در مرحله بعد، آموزش انجام شد:

python train.py
-world_size 1 -gpu_ranks 0
-data data/En2Fa_Translation/en2fa-bpe

```
-train_steps 50000
-save_checkpoint_steps 1000
-save_model data/En2Fa_Translation/en2fa-bpe-model
```

الف)

روش BPE(Byte Pair Encoding) الگوریتمی است که برای جداسازی subwordها و عملیات segmentation روی کلمات استفاده می‌شود. یکی از راههای مقابله با کلمات OOV استفاده از این روش است. در این روش، قبل از پیش پردازش، هر کلمه به subword ها شکسته می‌شود و در زمان آموزش، شبکه توانایی ترجمه این subwordها را بدست می‌آورد. لذا در زمان ترجمه، حتی اگر یک کلمه در داده های زمان آموزش نباشد، چون به subword هایش شکسته می شود، مدل می تواند sunword ها را ترجمه کند و بدین شکل به ترجمه نزدیکی از آن کلمه برسد.

ب)

برای اندازه گیری معیار BLEU، ابتدا ترجمه بر اساس آخرین مدل بدست آمده، انجام می‌شود :

```
python translate.py
    -model data/En2Fa_Translation/en2fa-bpe-model_step_50000.pt
    -src data/En2Fa_Translation/Test/test.en
    -output data/En2Fa_Translation/pred_bpe_50000.txt
    -replace_unk
    -verbose
```

سپس با استفاده از دستور زیر، detokenize انجام می‌شود :

```
sed -i "s/@@ //g"  pred_bpe_50000.txt
```

و در نهایت معیار BLEU محاسبه می‌شود :

```
perl  tools/multi-bleu.perl data/En2Fa_Translation/Test/test.fa <
data/En2Fa_Translation/pred_bpe_50000.txt
```

مقدار بدست آمده بر اساس چهار ترجمه مرجع، 29.31 می باشد.

ج)

نمونه 1 :

I could , but I don't do it .

جملات مرجع :

- ✓ من میتوانم , اما من ان را انجام نمیدهم .
- ✓ من میتوانم ، اما آن را انجام نمیدهم .

ترجمه بدون BPE :

- ✓ من ، اما من میتوانم ، اما don't .

ترجمه با BPE :

- ✓ من میتوانم ، اما من میتوانم این کار را انجام دهم .

در ترجمه های بالا، کلمه don’t که در حالت بدون BPE، ترجمه نشده بود، در حالت با استفاده از BPE، با استفاده از subword ها ترجمه شده، ترجمه نصفه نیمه آن، باعث شده معنای جمله معکوس شود!

نمونه 2 :

we have already agreed on the hotel .

جملات مرجع :

- ✓ ما قبلا درباره هتل به توافق رسیده-ایم .
- ✓ ما الان درمورد هتل به توافق رسیده-ایم .

ترجمه بدون BPE :

- ✓ ما قبلا توافق کردیم در هتل .

ترجمه با BPE :

- ✓ ما قبلا درمورد هتل به توافق رسیدیم .

در این مثال، در حالت با BPE، ترجمه درست تری ارائه شده است.

بخش دوم – نویسه‌گردانی فارسی به انگلیسی

برای این بخش، فایلهای train.en و train.fa برای مرحله آموزش و فایلهای dev.en و dev.fa بعنوان development set و فایلtest.en و test.fa برای ارزیابی نویسه‌گردانی در اختیار قرار گرفته است. این فایلها sentence-level aligned هستند و مشخصات کلی آنها در جدول زیر ذکر شده است :

فایلها	تعداد سطر (جمله)
train.en و train.fa	11،933
dev.en و dev.fa	1000
test.en و test.fa	1000

با توجه به اینکه، در سوال خواسته شده که داده های sentence-level به character-level تبدیل شوند، لذا ابتدا کد پایتون نوشته شد که هر سطر را خوانده و کاراکترهای آن را با space جدا نموده و یک فایل جدید می سازد. به عنوان مثال برای فایل train.en برنامه زیر اجرا شد:

```
ftrain=open("data\\Transliteration\\train.en", "r",encoding="utf-8")
lines = ftrain.readlines()
with open("data\\Transliteration\\train_chars.en", "w", encoding="utf-8") as f:
  f.write(
    "".join([
      " ".join([c if c != " " else "<b>" for c in l.lower()])
      for l in lines
    ])
  )
```

مشابه این عملیات برای هر 6 فایل جدول بالا اجرا شد. به عنوان مثال در زیر یک نمونه جمله فارسی و معادل فینگلیش آن به صورت عادی و بصورت character-level آمده است :

u yek shaaer e mashhur nist u yek shakhsiyat e shenaakhte shode nist

u <b> y e k <b> s h a a e r <b> e <b> m a s h h u r <b> n i s t <b> u <b> y e k <b> s h a k h s i y a t <b> e <b> s h e n a a k h t e <b> s h o d e <b> n i s t

او یک شاعر مشهور نیست او یک شخصیت شناخته شده نیست

ش <b> ش خ ص ی ت <b> ی ک <b> ا و <b> ن ی س ت <b> م ش ه و ر <b> ش ا ع ر <b> ی ک <b> ا و <b> ن ی س ت <b> ش د ه <b> ن ا خ ت ه

پس از انجام مرحله بالا، مشابه حالت الف بخش قبل، مراحل پیش‌پردازش، آموزش، ترجمه و ارزیابی انجام شد.

دستور اجرا شده برای مرحله پیش پردازش :

```
python preprocess.py
        -train_src data/Transliteration/train_chars.fa
        -train_tgt data/Transliteration/train_chars.en
        -valid_src data/Transliteration/dev_chars.fa
        -valid_tgt data/Transliteration/dev_chars.en
        -save_data data/Transliteration/trans_fa2en
```

دستور اجرا شده برای مرحله آموزش :

```
python train.py
        -world_size 1 -gpu_ranks 0
        -data data/Transliteration/trans_fa2en
        -train_steps 50000
        -save_checkpoint_steps 1000
        -save_model data/Transliteration/trans_fa2en_model
```

دستور اجرا شده برای مرحله ترجمه :

```
python translate.py
        -model data/Transliteration/trans_fa2en_model_step_50000.pt
        -src data/Transliteration/test_chars.fa
        -output data/Transliteration/pred_50000.txt
        -replace_unk
        -verbose
```

الف)

دستور اجرا شده برای مرحله محاسبه BLEU :

```
perl  tools/multi-bleu.perl data/Transliteration/test_chars.en <
data/Transliteration/pred_50000.txt
```

مقدار بدست آمده بلو بعد از 50000 بار تکرار، همانطور که در شکل زیر مشخص شده برابر 67.13 می‌باشد.

```
(py3-TF2.0) F:\Projects\OpenNMT-py-master>perl  tools/multi-bleu.perl data/Transliteration/test_chars.en < data/Transliteration/pred
_50000.txt
BLEU = 67.13, 95.6/88.6/82.5/78.1 (BP=0.781, ratio=0.802, hyp_len=57679, ref_len=71937)

(py3-TF2.0) F:\Projects\OpenNMT-py-master>
```

ب)

معیار بلو، در کل معیار کاملی نیست حتی برای ارزیابی ترجمه مثلا انگلیسی به فارسی یا بالعکس. زیرا به ساختار نحوی جمله ترجمه و ساختار معنایی آن بی‌توجه است و صرفاً بر اساس معیار شباهت n-gram ها کار می‌کند. این مساله برای نویسه گردانی حادتر هم هست. زیرا در ترجمه، برای هر کلمه، ترجمه معادل مشخصی وجود دارد، اما در نویسه گردانی، لزوماً نویسه دقیقی وجود ندارد و می تواند چندین نویسه معادل یک کلمه در نظر گرفته شود. لذا معیار شباهت n-gram هم خیلی مناسب نیست.

ج)

تفاوتی که در نویسه گردانی نسبت به ترجمه وجود دارد، این است که ترجمه یک جمله n کلمه‌ای می‌تواند یک جمله m کلمه‌ای باشد که n ممکن است بزرگتر، مساوی یا کوچکتر از m باشد. یعنی حین ترجمه، ممکن است چند کلمه حذف یا اضافه شوند و بسته به ساختار زبان مقصد، جابجا شوند. اما در نویسه گردانی، ساختار مقصد عیناً همان ساختار مبدا است و معمولاً نویسه ها، یک به یک به زبان مقصد نوشته می‌شوند. به این دلایل و همینطور نکته ذکر شده در بند قبلی، میتوان بجای BLEU از WER استفاده کرد. معیار WER (Word Error Rate)، معیاری است که عموماً برای ارزیابی سیستمهای ترجمه ماشینی یا تشخیص گفتار استفاده می‌شود و فرمول آن بصورت زیر است :

$$WER = \frac{S + I + D}{N}$$

S stands for substitutions (replacing a word).
I stands for insertions (inserting a word).
D stands for deletions (omitting a word).
N is the number of words that were actually said *Note: WER will be calculated incorrectly if you forget to normalize capitalization, punctuation, numbers, etc. across all transcripts*

د)

همان استدلالی که برای استفاده از bpe در ترجمه ماشینی مطرح می‌شود که میتواند باعث بهبود ترجمه با هندل کردن کلمات oov شود، برای نویسه گردانی هم می‌تواند صادق باشد. مثلا اگر در داده های آموزشی؛ کودکستان، بیمارستان، کوهستان و ... داریم؛ در ترجمه کلمه جدید، انارستان، میتواند با شکستن آن، به نویسه های معادل در فینگلیش برسد.