گزارش تمرین اول درس فهم زبان

موضوع تمرین:

پیاده‌سازی مدل‌های کدگذار-کدگشا و ترنسفورمر در تبدیل متن به واج

نام استاد درس: دکتر حسین زینلی

نام دانشجو: مجید ادیبیان

شماره دانشجویی: ۴۰۰۱۳۱۰۷۸

پاییز ۱۴۰۱

۱. مقدمه

در این تمرین باید دو نمونه از مدل‌های دنباله-به-دنباله در تسک تبدیل متن به واج پیاده‌سازی و ارزیابی شوند. برای این کار دو مدل کدگذار-کدگشا با مکانیزم توجه و ترنسفورمر در نظر گرفته شده‌اند. جهت آموزش مدل داده‌های آموزشی و آزمون فراهم شده است که داده‌های آموزشی شامل ۶۵۰۰۰ کلمه و دنباله واجی متناظر آن‌ها است و داده‌های آزمون شامل ۱۰۱۶۴ کلمه و دنباله واجی آن می‌باشد.

۲. روش انجام کار

۱.۲. پیاده‌سازی مدل

جهت پیاده‌سازی مدل‌های گفته شده از چارچوب تنسورفلو استفاده شده است. پیاده‌سازی دو مدل مورد نظر در فایل‌های مجرا در پوشه utils قرار گرفته‌اند.

مدل کدگذار-کدگشا: در پیاده‌سازی این مدل لایه‌های کدگذار، کدگشا و لایه توجه پیاده‌سازی شده‌اند. در قسمت کدگذار از دولایه LSTM دو طرفه و در قسمت کدگشا یک لایه LSTM یک‌طرفه استفاده شده است. ورودی‌های قسمت کدگشا پس از یک لایه embedding بر روی خروجی‌های کدگذار از مکانیزم توجه استفاده می‌کنند. در نهایت بر روی خروجی‌های کدگشا یک لایه dense استفاده شده تا خروجی را به ابعاد تعداد واج‌ها ببرد و واج‌ها مرحله به مرحله پیش‌بینی شوند. در پیاده‌سازی مورد نظر تعداد تعداد ابعاد LSTM و همچنین تعداد ابعاد embedding برابر ۲۵۶ قرار داده شده است. همچنین از cross entropy به عنوان تابع خطا و از Adam برای بهینه‌ساز استفاده شده است.

مدل ترنسفورمر: در این مدل قسمت کدگذار و کدگشا به صورت مجزا پیاده‌سازی شده‌اند و در هر یک بر اساس پارامترهای ورودی تعدادی بلوک کدگذار و کدگشا استفاده شده است. در ابتدا ورودی‌ها embed شده و بر روی آن‌ها positional embedding اضافه می‌شود. در پیاده‌سازی انجام شده تعداد بلوک‌های کدگذار و کدگشا هر دو برابر ۳ بلوک قرار داده شده است و از multi-head attention با ۸ head استفاده شده است. همچنین برای تابع خطا از cross entropy و از Adam به عنوان بهینه‌ساز استفاده شده است.

۲.۲. آماده‌سازی داده‌ها

داده‌های فراهم شده در یک فایل متنی قرار گرفته‌اند که در هر سطر آن یک کلمه فارسی و معادل واجی آن وجود دارد. پس از خواندن فایل متنی هر سطر آن استخراج شده و در هر سطر دنباله حروف و دنباله واجی نظیر آن به دست می‌آید. سپس از این دنباله‌ها در ساخت داده‌های ورودی کدگذار، ورودی کدگشا و خروجی کدگشا استفاده شده است. به این صورت که ابتدای ورودی کدگشا دارای توکن <bos> و انتهای خروجی کدگشا دارای توکن <eos> می‌باشد. برای هم اندازه کردن ورودی‌ها نیز از توکن <pad> استفاده شده است.

۳.۲. آموزش مدل

ابتدا یک درصد از داده‌های آموزشی را برای ارزیابی مدل در حین آموزش جدا کرده‌ایم. برای آموزش مدل کدگذار-کدگشا، مدل را بر روی داده‌های آموزشی ۱۰ دور کامل آموزش داده‌ایم که زمان هر دور تقریبا برابر ۵۰ ثانیه می‌باشد. برای مدل ترنسفورمر نیز مدل را ۴ دور کامل آموزش می‌دهیم که باز هم هر دور آموزش آن حدود یک دقیقه طول می‌کشد.

نمونه‌ای از روند آموزش مدل کدگذار کدگشا و ترنسفورمر به صورت زیر است:

encoder-decoder:

```
2011/2011 [==============] - 54s 22ms/step - loss: 0.1868 - val_loss: 0.0675
2011/2011 [==============] - 41s 21ms/step - loss: 0.0570 - val_loss: 0.0510
2011/2011 [==============] - 41s 20ms/step - loss: 0.0447 - val_loss: 0.0443
2011/2011 [==============] - 41s 20ms/step - loss: 0.0380 - val_loss: 0.0406
```

transformer:

```
2011/2011 [==============] - 84s 37ms/step - loss: 0.1473 - val_loss: 0.0737
2011/2011 [==============] - 70s 35ms/step - loss: 0.0706 - val_loss: 0.0597
2011/2011 [==============] - 69s 34ms/step - loss: 0.0609 - val_loss: 0.0545
2011/2011 [==============] - 69s 34ms/step - loss: 0.0549 - val_loss: 0.0511
```

۳.۲. ایده‌های پیاده‌سازی

جهت بهبود دقت مدل در پیاده‌سازی موارد زیر انجام شده است:

- ورودی قسمت کدگذار که دنباله مربوط به حروف فارسی هستند به دو حالت مختلف بررسی شده‌اند. در حالت اول دنباله هر حرف و در حالت دیگر از دنباله دو حرفی‌ها استفاده شده است. این تغییر در هر دو مدل کدگذار-کدگشا و ترنسفورمر بررسی خواهد شد.
- در هنگام تولید دنباله واجی در فاز تست، قواعدی بررسی می‌شود که واج تولید شده مطابق با دنباله واجی معتبر در فارسی باشد. به این صورت که بررسی می‌شود اگر واج قبلی هم‌خوان بوده واج فعلی می‌تواند

واکه باشد و اگر واج قبلی واکه بوده واج فعلی تنها باید هم‌خوان باشد. همچنین نباید بیشتر از دو واج تکراری کنار هم قرار گیرند.

۴.۲. نحوه اجرا

در هنگام اجرای برنامه می‌توان پارامترهای اجرای کد را به عنوان ورودی به آن داد. یکی از این پارامترها --verbose است که مشخص می‌کند خروجی‌ها چاپ شوند یا خیر. همچنین می‌توان امکان آموزش و آزمون مدل به دو صورت unigram و bigram اجرا کرد که برای این کار پارامتر --ngram را برابر یکی از آن‌ها قرار می‌دهیم. برای تعیین نوع مدل استفاده شده باید پارامتر --model را برابر یکی از مقادیر encoder_decoder یا transformer قرار می‌دهیم.

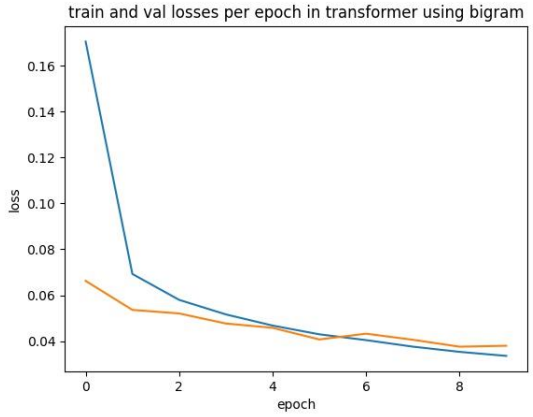
نمونه تمام دستورات برای اجرای کدها در فایل notebook موجود در پوشه code وجود دارد. نمونه‌ای از دستور آموزش و آزمون مدل به صورت زیر است:

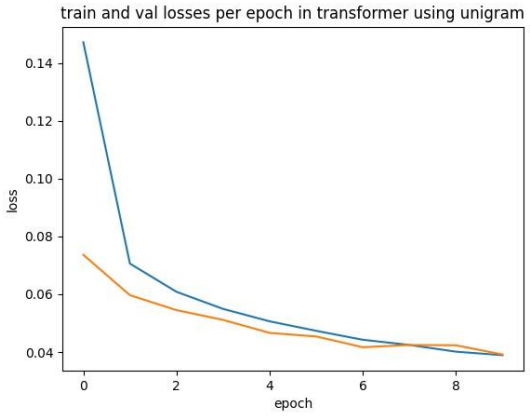
```
python train.py --model 'encoder_decoder' --save_model_path 'models/'
                --n_gram 'bigram' --data_path 'data/train_data.txt' --batch_size 32
                --val_size 0.01

python test.py --model 'encoder_decoder' --data_path 'data/test_data.txt'
               --n_gram 'bigram' --verbose True
```

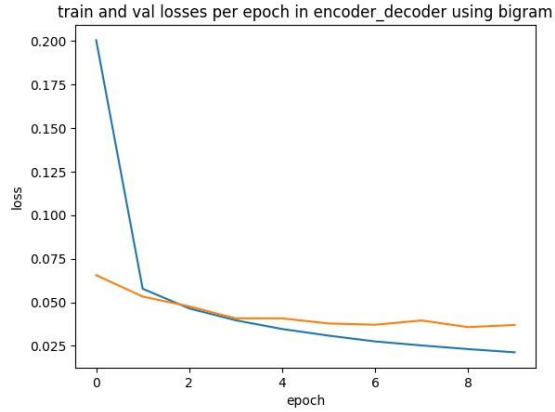
۳. نتایج:

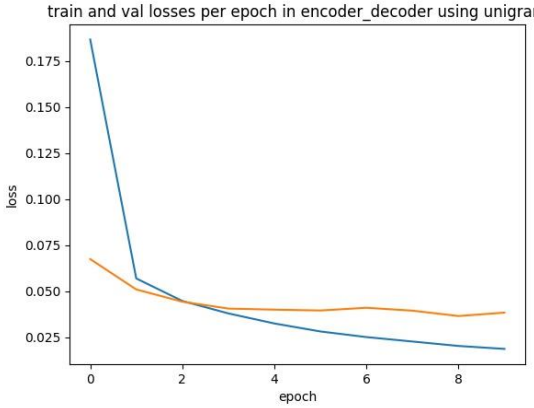
با استفاده از مدل‌های آموزش دیده سعی می‌کنیم دنباله واجی متناظر با داده‌های آزمون را به دست آوریم. ابتدا با آزمون و خطای بسیار بهترین پارامترها را برای مدل‌ها یافته‌ایم. در انتها هر یک از دو مدل را در دو حالت استفاده از unigram و bigram آموزش می‌دهیم و نتیجه آن‌ها را بر روی داده آزمون بررسی می‌کنیم.

تغییرات loss در هر یک از چهار حالت گفته شده به صورت زیر است:









همچنین هر یک از چهار حالت گفته شده را بر روی داده‌های آزمون اجرا می‌کنیم و مقدار خطای واجی و خطای کلمه‌ای را در هر حالت به دست می‌آوریم که نتایج به صورت زیر است:

	کدگذار-کدگشا		ترنسفورمر	
	unigram	bigram	unigram	bigram
نرخ خطای کلمه	0.531	0.582	0.375	0.356
نرخ خطای واج	0.166	0.177	0.084	<u>0.076</u>

دیده می‌شود که مقدار خطا در مدل کدگذار-کدگشا با استفاده از unigram و در مدل ترنسفورمر با استفاده از bigram بهتر شده است. نمونه‌ای از خروجی‌های مدل در این دو حالت در جدول زیر آورده شده است:

کلمه	دنباله واجی	نتیجه کدگذار-کدگشا	نتیجه ترنسفورمر
پدافند	padAfand	podAfand	pedAfand
تاریخچه	tArixCe	tArixCe	tArixCe
گرمایش	garmAyeS	garmAyeS	garmAyeS
انزجارآمیز	‘enzejAr’Amiz	enzejAr'Amiz	enzejAr'Amiz
نامنظم	nAmonazzam	nAmonzam	nAmanzam

نتیجه‌گیری:

همان‌طور که از نتایج به دست آمده مشخص است مقدار خطای واجی به دست آمده در مدل کدگذار-کدگشا از خطا در ترنسفورمر بیشتر است.

همچنین دیده می‌شود که استفاده از bigram ها در قسمت کدگذار توانسته در مدل ترنسفورمر باعث بهبود مدل شود در حالی که در مدل کدگذار-کدگشا مقدار خطا را افزایش داده است که ممکن است دلیل این افزایش خطا عدم توانایی مدل کدگذار-کدگشا در توجه کافی خروجی‌های کدگذار و همچنین عدم آموزش کافی به دلیل داده ناکافی برای حالت bigram باشد. ولی استفاده از bigram در ترنسفورمر باعث بهبود خطا شده است که دلیل این امر آن است که استفاده از bigram ها باعث می‌شود برای هر توکن ورودی مشخص باشد که حرف فعلی و بعدی آن چیست و در نتیجه مدل بهتر می‌تواند حرکت‌گذاری مناسبی برای هر حرف پیش‌بینی کرد. می‌توان این فرایند را برای trigram ها نیز تکرار کرد که البته هرچه تعداد ngram ها را افزایش دهیم به دیتای آموزشی بیشتری نیاز است که انواع حالت‌های آن ngram را به خوبی پوشش دهد.