بسم‌الله الرحمن الرحیم

گزارش فاز نهایی پروژه درس NLP

نرگس سادات حسینی

لینک دسترسی به پروژه : NLP Project

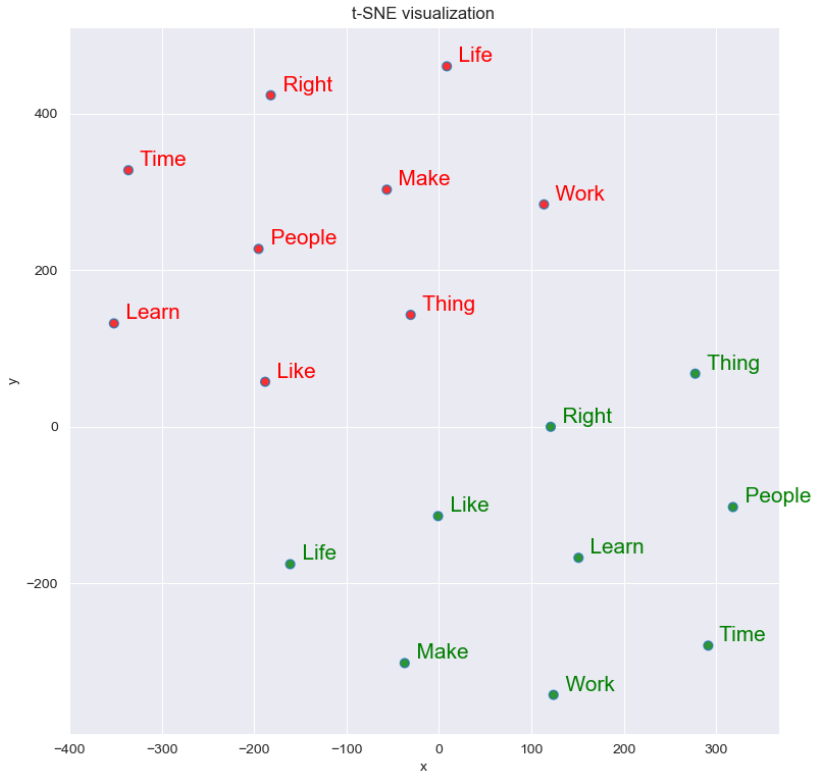
۵ تیر ۱۴۰۰

فهرست مطالب

عنوان صفحه

۱ بخش word2vec

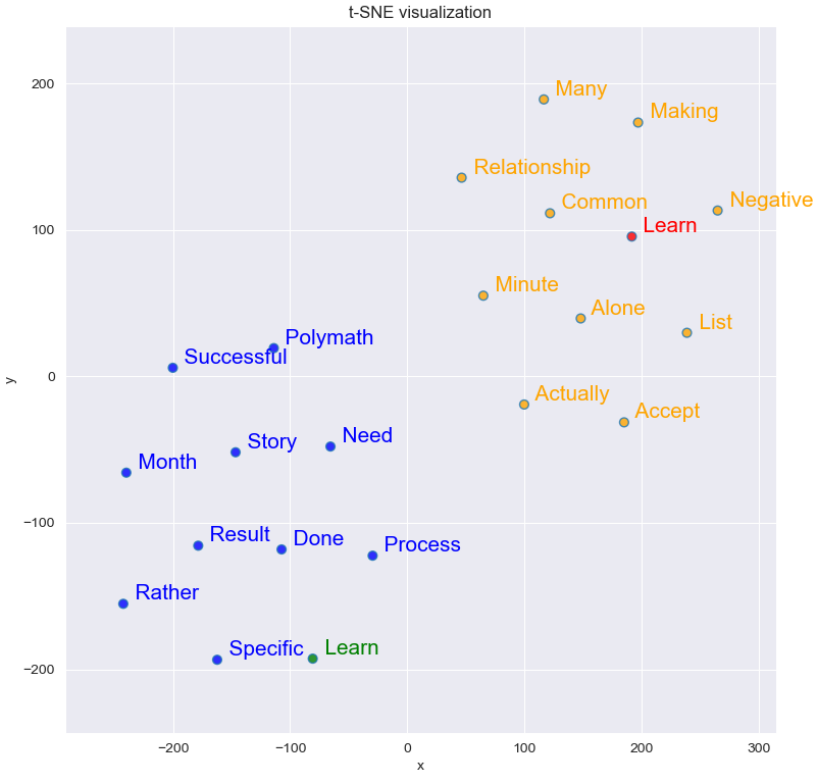
برای به دست آوردن بردارد کلمات در این قسمت از کتابخانه gensim استفاده شده است. علت استفاده از این کتابخانه و عدم استفاده از تمرین ۲A این بود که در تمرین ۲A داده ورودی به حالت خاصی بود که تغییر داده‌های موجود به دست‌آمده از فاز قبل به شکل داده‌های تمرین۲ کار وقت گیری بود و به همین علت از کتابخانه آماده برای آموزش مدل استفاده شد.
برای آموزش مدل، ابتدا مدل اولیه توسط تابع Word۲Vec ساخته می‌شود و پارامترهای آن مانند اندازه بردار کلمات مشخص می‌شوند. سپس با استفاده از داده ورودی، کلمات مدل ساخته می‌شوند و درآخر مدل روی داده آموزش می‌بیند. داده‌ای که برای آموزش مدل استفاده شده است، از combined data word broken.csv می‌باشد که داده جمع‌آوری شده و پردازش شده از فاز قبل است که به کلمات شکسته شده‌است. مدل‌های آموزش دیده در پوشه models ذخیره می‌شوند.
پس از آموزش مدل، بردار تعدادی از کلمات پرتکرار مشترک بین دو برچسب رسم شده است که به صورت زیر می‌باشد:



شکل ۱: بردار کلمات مشترک

همانطور که مشخص می‌باشد بردار یک کلمه در هر لیبل متفاوت می‌باشد(نقاط سبز برای برچسب motiva- tional و نقاط قرمز برای برچسب nonMotivational می‌باشند). از نظر من علت تفاوت این می‌باشد که هر کلمه

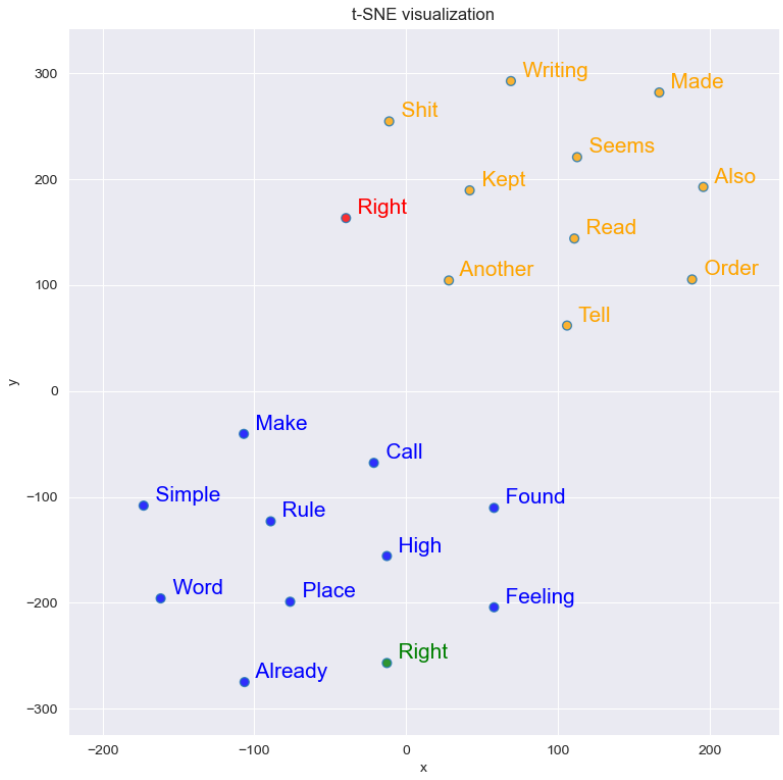
در هر برچسب در یک context خاص آمده و به همین علت بردار آن‌ها متفاوت می‌باشد. برای بررسی بیشتر دو کلمه به طور جداگانه بررسی شداند و همچنین برای فهم بهتر کلمات مشابه با آن کلمات نیز استفاده شده‌اند.
در مثال اول بردار کلمه learn و کلمات مشابه آن برای هربرچسب رسم شده‌است(برای تشابه از تابع most similar استفاده شده‌است) . نقاط آبی و سبز برای برچسب motivational و نقاط نارنجی و قرمز برای برچسب nonMo-tivational می‌باشند.



شکل ۲: بردار learn

با دقت در کلمات نزدیک هر برچسب دیده می‌شود که برای برچسب motivational کلماتی تا حدودی بار مثبت دارند و مفاهیمی همچون process یا انجام دادن را بیشتر می‌رسانند. اما در مقابل کلمات برچسب nonmotiva-tional این بار مثبت را ندارند و تا حدودی می‌توان گفت حتی بار منفی دارند. به طور مثال کلماتی همچون alone می‌توانند تاییدکننده این موضوع باشند.

مثال دوم کلمه right می‌باشد که بردار کلمات به صورت زیر می‌باشد:



شکل ۳: بردار right

در این مثال چیزی که قابل توجه است این می‌باشد که دیگر در این حالت نمی‌توان ادعا داشت که یک کلمه بار مثبت و دیگری منفی دارد(برخلاف مثال قبل). از نظر من علت این‌که کلمه right برای هر دو برچسب تقریبا یک معنی دارد(هرچند بردار آن‌ها با توجه به متفاوت بودن context و این که هر کدام از آن‌ها رو یک مدل آموزش دیده‌اند، متفاوت است) این می‌باشد که بعضی کلمات تنها یک معنی دارند و نمی‌توانند بار مثبت یا منفی بگیرند. به طور مثال کلمه learn که در مثال قبل بود، می‌تواند به معنی آموزش موضوعات جدید برای پیشرفت باشد یا می‌تواند به معنی درس گرفتن از یک اتفاق ناخوشایند پیش‌آمده باشد، اما تمامی معانی کلمه right به گونه‌ای هستند که نمی‌توان این نوع دسته‌بندی برای معانی آن در نظر گرفت.
برای بررسی بایاس در مدل از تابع most similar استفاده شده‌است. یک مثال از بایاس به صورت زیر می‌باشد:

male->sexual = female -> [('time'), ('feel'), ('assault')]
female->sexual = male -> [('assault'), ('survivor'), ('harassment')]

برای پیدا کردن بایاس کلمات متفاوتی انتخاب شدند و با توجه به اینکه همه‌ی داده‌ها از یک موضوع نبودند و جمع چندین موضوع متفاوت بودند، پیدا کردن بایاس کار سختی بود و بعضا خروجی مناسبی تولید نمی‌شد(همانند مثال

بالا برای female) اما در مثال گفته شده بایاس مشخص می‌باشد به این صورت که برای female کلمات مرتبطی را تولید نمی‌کند اما برای male اولین کلمه‌ای که پیدا شده کلمه assault می‌باشد(و دو کلمه دیگر نیز حدودا باهمین مفهوم هستند) که به نظر من این نوعی بایاس می‌باشد که یک بدبینی نسبت به male دربرابر female می‌باشد(مثلا در موضوع assault) که این بایاس در افکار جامعه و در نتیجه در متن جمع‌آوری شده نیز وجود دارد.

۲ بخش tokenization

برای حل این قسمت ابتدا داده تمیز شده خوانده می‌شود. سپس داده به ۵ قسمت تقسیم می‌شود و در هر بار از ۵ بار اجرا، یک قسمت به عنوان داده تست و بقیه به عنوان داده آموزش انتخاب می‌شوند. سپس داده به صورت فایل txt. ذخیره می‌شود. برای اندازه کلمات ۵ اندازه، ۱۰، ۳۰، ۱۰۰، ۱۰۰۰ و ۴۰۵۴ انتخاب شده است. سپس توسط کتابخانه sentencepiece عملیات آموزش بر روی داده و با اندازه کلمات داده شده انجام می‌شود. مدل‌های تولید شده در پوشهmodels قرار دارند. تعداد unk های تولید شده نسبت به طول کل توکن‌های تولید شده در tokenization counts.csv ذخیره شده است. نسبت unk های تولید شده به صورت زیر می‌باشد:

	count
0	0.484258
1	0.459780
2	0.413959
3	0.321619
4	0.271389

شکل ۴: نسبت unk های تولید شده

همانطور که در تصویر مشخص می‌باشد، هرچه تعداد کلمات بیشتر شود، نسبتunk های تولید شده کاهش میابد.

۳ بخش parsing

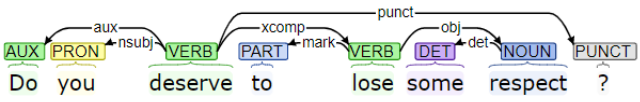
برای حل این قسمت از کد تمرین سوم استفاده شده است. در این تمرین یک parser با داده‌های انگلیسی آموزش می‌بینید. برای بررسی عملکرد parser سه جمله به فرم conll تبدیل می‌شود و parser روی آن‌ها اعمال می‌شود. فایل conll به صورت زیر می‌باشد:

```
1   Skipped _   VERB    VBD _   0   root    _   _
2   the _   DET DT  _   3   det _   _
3   gym _   NOUN    NN  _   1   obj _   _
4   ?   _   PUNCT   .   _   1   punct   _   _

1   You _   PRON    PRP _   5   nsubj   _   _
2   are _   AUX VBP _   5   aux _   _
3   n't _   PART    RB  _   5   advmod  _   _
4   just    _   ADV RB  _   5   advmod  _   _
5   defending   _   VERB    VBG _   0   root    _   _
6   an  _   DET DT  _   7   det _   _
7   individual  _   NOUN    NN  _   5   obj _   _
8   in  _   ADP IN  _   10  case    _   _
9   these   _   DET DT  _   10  det _   _
10  debates _   NOUN    NNS _   7   nmod    _   _
11  .   _   PUNCT   .   _   5   punct   _   _
```

شکل ۵: فایل پارس درست جمله‌ها

دقت parser برای این ۳ جمله ۹۵.۶۵ می‌باشد.
یک نمونه از مواردی که parser در آن اشتباه دارد جمله زیر می‌باشد. درخت پارس جمله به صورت زیر می‌باشد:



شکل ۶: درخت پارس

اما پارسر در انتخاب head کلمه some اشتباه دارد و head آن را کلمه lose در نظر می‌گیرد.

۴ بخش language model

!برای اجرا این بخش با توجه به این‌که به GPU نیاز داشتم کد را در نوت‌بوک زدم و هم فایل نوت‌بوک و هم فایل py. آن موجود است.
برای ساختن language model در این بخش از keras استفاده شده است. در مرحله اول داده‌ی هر برچسب به صورت کلمه کلمه به عنوان لیستی از token ها ذخیره می‌شوند. سپس برای هر کلمه یه context به طول length در نظر گرفته می‌شود تا مدل با استفاده از context کلمه بعدی را پیش‌بینی کند. سپس برای انجام محاسبات بعدی context ذخیره شده(یا sequence) به صورت اعداد int ذخیره می‌شوند.
سپس ورودی و خروجی مدل برای آموزش بر اساس sequence تولید می‌شود. معماری شبکه به صورت زیر می‌باشد:

```
model = Sequential()
model.add(Embedding(vocab_size, 50, input_length=seq_length))
model.add(LSTM(100, return_sequences=True))
model.add(LSTM(100))
model.add(Dense(100, activation='relu'))
model.add(Dense(vocab_size, activation='softmax'))
```

شکل ۷: معماری شبکه

برای بررسی عملکرد شبکه یک متن دلخواه با طول مشخص به شبکه می‌دهیم . خروجی شبکه برای برچسب non-Motivational به صورت زیر می‌باشد:

[the war between syria and ISIS has affected life of many muslims] pattern toward learn feel suppressing revolution 2000 debt turkey rick describing among able published nation assume military stole appeal ptsd happen quite terrible outline voice blair guardian west administration want attack syria began respected israel imposed mandated dimension since dimension soviet since since since sold continues racism egypt russia rejected

وبرای motivational :

to be productive you should start state like kinetic applied area hour ratchet

impact single midlow receive move mean thing start wish wished quantity single midlow difference mutual time abstract goal time want make life want become going succeed business people want compelled write mean manager total daily people available information large daily year every habit

با توجه به متن تولید شده به نظر می‌آید که شبکه کلمات را تقریبا متناسب با برچسب می‌دهد اما جملات تولید شده از نظر قواعد و معنی درست نیستند و به نظر من علت این موضوع این باشد که داده‌ای که مدل روی آن آموزش دیده است مناسب نبوده است، زیرا در مراحل اولیه پردازش داده بسیاری از اطلاعات (به طور مثال puncutaion و stopwrds) حذف شده‌اند و تنها کلمات به مدل داده شده‌اند. به همین علت جملاتی تولیدی کیفیت مطلوب را ندارند اما می‌توان گفت کلمات تولید شده به درستی و مرتبط با برچسب تولید شده‌اند.

۵ بخش fine tuning

!این قسمت نیز همانند قسمت ۴ در نوت‌بوک نوشته شده است.

۱.۵ language model

برای تولید language model از آنجا که استفاده از bert کار مشکلی بود از مدل distilgpt۲ استفاده شده است. در مرحله اول داده آموزش و تست به نسبت ۸۵ به ۱۵ از یکدیگر جدا می‌شوند. سپس مدل pretreined distilgpt2 بارگذاری می‌شود و به صورت جدا روی هر کلاس داده fine tune می‌شود.
خروجی مدل برای کلاس motivational به صورت زیر می‌باشد:

```
   pipline('being productive is to')[0]['generated_text']

Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.

'being productive is to use tools that may help," in this situation, it is likely that you have developed tools that
may help the user use tools that may help improve productivity using tools that may help productivity in your own li
fe.\n\n\n'
```

شکل ۸: خروجی مدل برای کلاس motivational

و برای کلاس nonMotivational به صورت زیر است:

```
   pipline('politic is')[0]['generated_text']

Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.

'politic is about social justice, but does work well for social justice as far as society is concerned. This has tak
en my life, I have had a difficult time doing so much as I'm ashamed and even angry,' and I'
```

شکل ۹: خروجی مدل برای کلاس nonMotivational

همانطور که از تصاویر مشخص می‌باشد این مدل نسبت به مدل قسمت قبل بسیار بهتر عمل می‌کند و علاوه بر اینکه کلمات رو مرتبط با برچسب می‌دهد، جملات از نظر ساختاری و معنایی نیز منطقی و درست می‌باشند.

۲.۵ classification

در این قسمت برای classification با استفاده از bert ، ابتدا bert base uncased که یک pretrained tokenizer می‌باشد، load می‌شود.
سپس داده ورودی به دو دسته آموزش و تست دسته بندی می‌شود(داده ورودی داده پردازش شده و clean شده می‌باشد).

سپس tokenizer گفته شده برای tokenize کردن دو داده آموزش و تست استفاده می‌شود.
در مرحله بعد دو داده تست و آموزش به فرمت خاص برای مراحل بعد تبدیل می‌شوند. سپس pretrained bert classifier به عنوان مدل آموزش در نظر گرفته می‌شود و آموزش روی آن انجام می‌شود. نتیجه آموزش به صورت زیر می‌باشد:

```
{'epoch': 20.0,
 'eval_loss': 0.0958588570356369,
 'eval_runtime': 1.0078,
 'eval_samples_per_second': 27.782,
 'eval_steps_per_second': 3.969}
```

شکل ۱۰: ارزیابی مدل

۶ نکاتی درباره کد

برای قسمت ۴ و ۵ از آنجایی که کد در کولب بود، دیتای موردنیاز دستی آپلود می‌شد(در کد py. آدرس‌ها درست است اما در نوت‌بوک باید آپلود شود). به علت حجم بالای مدل‌ها امکان اپلود آن ها نبود به همین علت لینک درایو آن‌ها را قرار دادم.
مدل‌ها در پوشه models ذخیره شده‌اند(برای مدل‌هایی که حجم بالایی داشتند لینک آن‌ها قرار داده شده است). برای قسمت parsing از آنجایی که مدل قبلا در تمرین آموزش دیده بود، دوباره آموزش داده نشده و آن قسمت از کد که مربوط به آموزش مدل بود از کد حذف شده است.
مدل‌های قسمت اول نیز در پوشه models قرار دارند.

مراجع

[1] https://www.kaggle.com/pierremegret/gensim-word2vec-tutorial

[2] https://stackoverflow.com/questions/43776572/visualise-word2vec-generated-from-gensim

[3] https://towardsdatascience.com/a-beginners-guide-to-word-embedding-with-gensim-word2ve

[4] https://medium.com/analytics-vidhya/a-comprehensive-guide-to-build-your-own-language-m

[5] https://machinelearningmastery.com/how-to-develop-a-word-level-neural-language-model-i

[6] https://huggingface.co/transformers/training.html

[7] https://towardsdatascience.com/train-gpt-2-in-your-own-language-fc6ad4d60171https://colab.research.google.com/github/philschmid/fine-tune-GPT-2/blob/master/Fine_tune_a_non_English_GPT_2_Model_with_Huggingface.ipynb#scrollTo=hKBSyNLgqF9K