



بسم الله الرحمن الرحيم

گزارش فاز نهایی پروژه درس NLP
 نرگس سادات حسینی
 لینک دسترسی به پروژه : [NLP Project](#)

۲ تیر ۱۴۰۰

فهرست مطالب

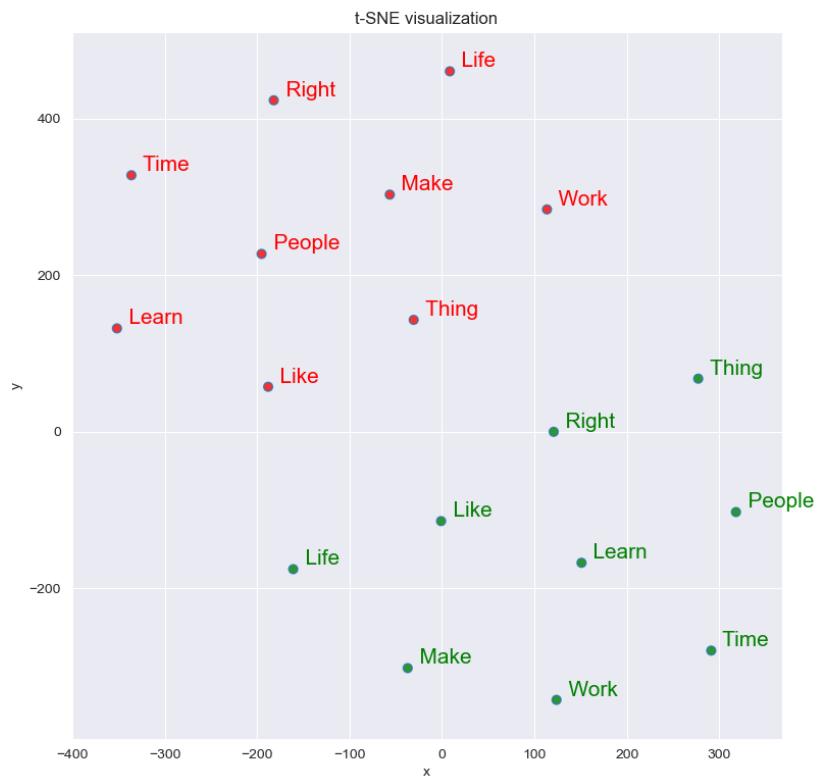
صفحه	عنوان
۳	۱ بخش word2vec
۶	۲ بخش language model
۷	۳ بخش fine tuning
۷	۴ نکاتی درباره کد

۱ بخش word2vec

برای به دست آوردن بردار کلمات در این قسمت از کتابخانه gensim استفاده شده است. علت استفاده از این کتابخانه و عدم استفاده از تمرین A۲ این بود که در تمرین A۲ داده ورودی به حالت خاصی بود که تغییر داده‌های موجود به دست آمده از فاز قبل به شکل داده‌های تمرین ۲ کار وقت گیری بود و به همین علت از کتابخانه آماده برای آموزش مدل استفاده شد.

برای آموزش مدل، ابتدا مدل اولیه توسط تابع Word2Vec ساخته می‌شود و پارامترهای آن مانند اندازه بردار کلمات مشخص می‌شوند. سپس با استفاده از داده ورودی، کلمات مدل ساخته می‌شوند و در آخر 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 شده است. مدل‌های آموزش دیده در پوشه models ذخیره می‌شوند.

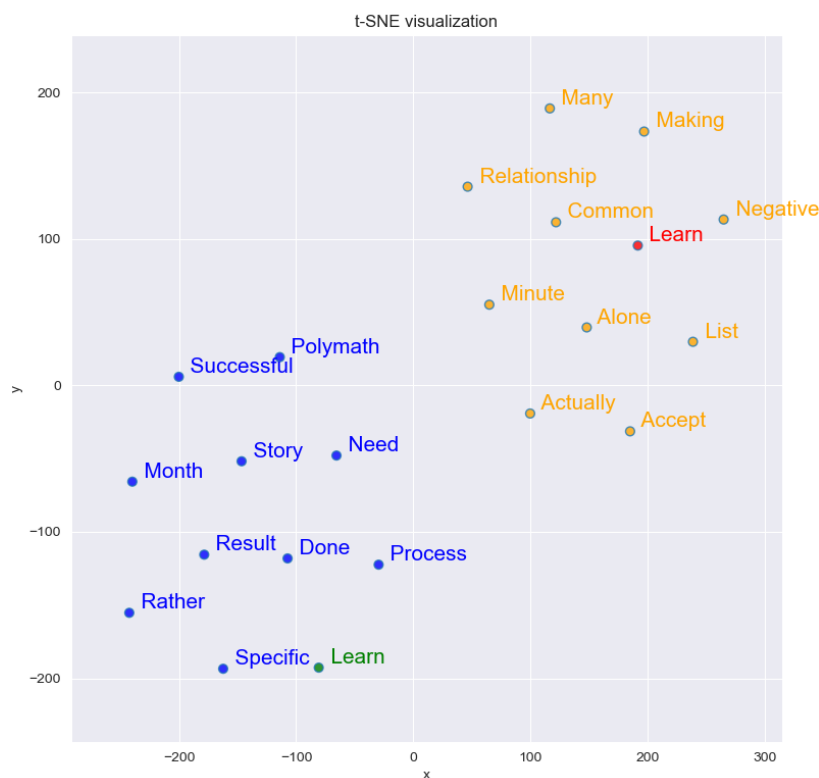
پس از آموزش مدل، بردار تعدادی از کلمات پرتکرار مشترک بین دو برجسب رسم شده است که به صورت زیر می‌باشد:



شکل ۱: بردار کلمات مشترک

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

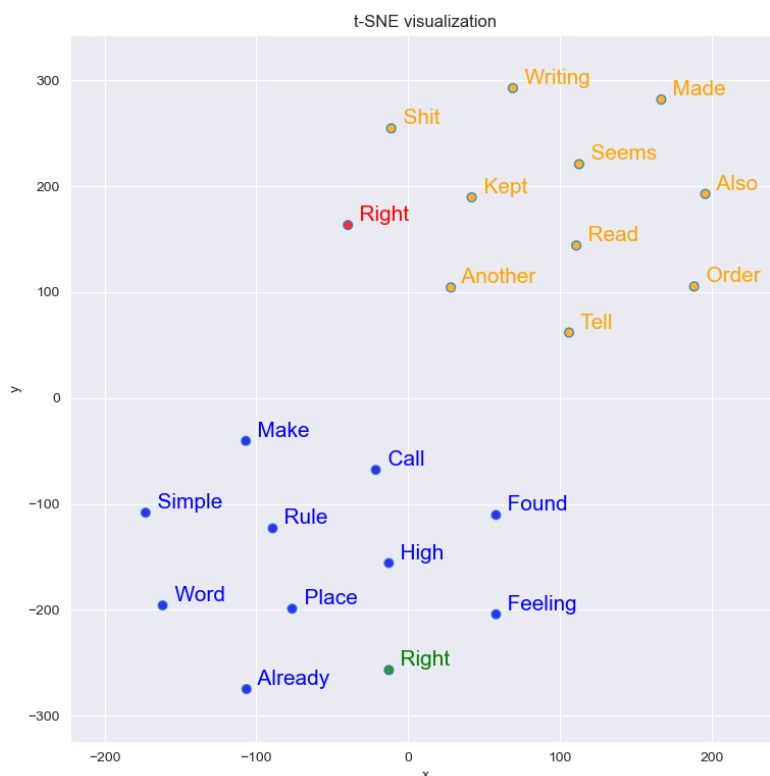
در هر برچسب در یک context خاص آمده و به همین علت بردار آن‌ها متفاوت می‌باشد. برای بررسی بیشتر دو کلمه به طور جداگانه بررسی شدند و همچنین برای فهم بهتر کلمات مشابه با آن کلمات نیز استفاده شده‌اند. در مثال اول بردار کلمه learn و کلمات مشابه آن برای هر برچسب رسم شده‌است (برای تشابه از تابع most similar استفاده شده‌است). نقاط آبی و سبز برای برچسب motivational و نقاط نارنجی و قرمز برای برچسب nonMotivational 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 nonmotivational 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 می باشد که بردار کلمات به صورت زیر می باشد:



شکل ۳: بردار right

در این مثال چیزی که قابل توجه است این می باشد که دیگر در این حالت نمی توان ادعا داشت که یک کلمه بار مثبت و دیگری منفی دارد (برخلاف مثال قبل). از نظر من علت این که کلمه right برای هر دو برچسب تقریباً یک معنی دارد (هرچند بردار آن ها با توجه به متفاوت بودن context و این که هر کدام از آن ها رو یک مدل آموزش دیده اند، متفاوت است) این می باشد که بعضی کلمات تنها یک معنی دارند و نمی توانند بار مثبت یا منفی بگیرند. به طور مثال کلمه learn که در مثال قبل بود، می تواند به معنی آموزش موضوعات جدید برای پیشرفت باشد یا می تواند به معنی درس گرفتن از یک اتفاق ناخوشایند پیش آمده باشد، اما تمامی معانی کلمه right به گونه ای هستند که نمی توان این نوع دسته بندی برای معانی آن در نظر گرفت.

برای بررسی بایاس در مدل از تابع most similar استفاده شده است. یک مثال از بایاس به صورت زیر می باشد:

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 همه ی داده ها از یک موضوع نبودند و جمع چندین موضوع متفاوت بودند، پیدا کردن بایاس کار سختی بود و بعضاً خروجی مناسبی تولید نمی شد (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 حدوداً باهمین مفهوم هستند) که به نظر من این نوعی بایاس می‌باشد که یک بدبینی نسبت به male در برابر female می‌باشد (مثلاً در موضوع assault) که این بایاس در افکار جامعه و در نتیجه در متن جمع‌آوری شده نیز وجود دارد.

۲ بخش language model

برای اجرا این بخش با توجه به این‌که به GPU نیاز داشتیم کد را در نوت‌بوک زدیم 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 به 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 ۴: 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 re-
spected israel imposed mandated dimension since dimension soviet since since
since sold continues racism egypt russia rejected

و برای 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 تقریباً 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.

۳ بخش fine tuning

این قسمت نیز همانند قسمت ۴ در نوت‌بوک نوشته شده است. 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 ۵: ارزیابی مدل

۴ 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 آپلود آن‌ها نبود به همین علت لینک درایو آن‌ها را قرار دادم.

لینک مدل‌های سوال ۴ : [language model](#)

لینک مدل سوال ۵ : [bert classifier model](#)

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-word2vec>
- [4] <https://medium.com/analytics-vidhya/a-comprehensive-guide-to-build-your-own-language-model>
- [5] <https://machinelearningmastery.com/how-to-develop-a-word-level-neural-language-model-in-pytorch/>
- [6] <https://huggingface.co/transformers/training.html>