بسم اللّه الرحمن الرحیم

پروژه پایانی پردازش زبان طبیعی

غزاله محمودی

۵ تیر ۱۴۰۰

فهرست مطالب

فهرست تصاویر

فهرست جداول

word2vec ۱

در این بخش قصد داریم با استفاده از ماژول gensim بردار word2vec را برای هر کلمه حساب کنیم.
برای به دست آوردن word2vec به کمک Gensim تعدادی پارامتر قابل تنظیم دارد که در ادامه به بررسی آن ها می پردازم.

- Size

این پارامتر تعیین کننده سایز vector برای نمایش هر word یا token است. هر چه دیتاست محدود تر و کوچکتر باشد این عدد نیز کوچک تر در نظر گرفته می شود و هر چه دیتاست بزرگتر باشد(کلمات unique بیشتری داشته باشد) باید اندازه vector بزرگتر در نظر گرفته شود. تجربه نشان داده اندازه بین ۱۰۰ تا ۱۵۰ برای دیتاست های بزرگ مقدار مناسبی است.

- Windows

این پارامتر تعیین کننده بیشترین فاصله مابین کلمه اصلی و همسایه های آن می باشد.از لحاظ تئوری هر چه این سایز کوچکتر باشد کلماتی که بیشتر ارتباط را به یکدیگر دارند به عنوان خروجی برمی گرداند. اگر تعداد داده به اندازه کافی بزرگ باشد سایز پنجره اهمیت زیادی ندارد اما باید این نکته را در نظر گرفت که این سایز نباید خیلی بزرگ یا بیش از حد کوچک باشد. اگر درباره انتخاب آن اطمینان نداریم بهتر است از مقدار پیش فرض استفاده کنیم.

- Min count

این پارامتر حداقل تکرار کلمه در دیتاست را نشان می دهد که در صورتی که کلمه ای به این تعداد تکرار شود در word embedding مورد توجه قرار می گیرد و در غیر این صورت کنار گذاشته می شود. تعیین این عدد در دیتاست های بزرگ برای کنار گذاشتن کلمات کم اهمیت که غالبا کم تکرار می شوند مناسب است. همچنین در مصرف بهینه ممووری و حافظه هم تاثیر دارد.

۱.۱ بررسی bias

در بخش بعدی آزمایش‌ها، احتمال وجود bias در دیتاست مورد بررسی قرار گرفت. برای این منظور مشابهت‌های مقابل مورد بررسی قرار گرفت.

woman as doctor similar as man as ?

که خروجی به صورت شکل۱ شد.

	word	score
0	psychiatrist	0.543868
1	sent	0.527563
2	symptom	0.509729
3	nearly	0.466752
4	appointment	0.451832
5	medication	0.442493
6	therapist	0.433560
7	ward	0.427322
8	adhd	0.424467
9	med	0.389320

شکل ۱: woman as doctor similar as man as ?

در ادامه برای پیدا کردن bias احتمالی بین زن و مرد در این دیتاست به ازای ورودی زیر اجرا را تکرار کردیم. خروجی به صورت شکل۲ شد.

man as doctor similar to woman as ?

	word	score
0	mum	0.529463
1	dad	0.484532
2	said	0.484175
3	therapist	0.466411
4	sister	0.459618
5	mad	0.443832
6	psychiatrist	0.441509
7	phone	0.441025
8	back	0.433116
9	grandma	0.433048

شکل ۲: man as doctor similar to woman as ?

همان‌طور که در گزارشات مطرح شده به وضوح مشخص است این دیتا برای جنسیت خانم‌ها و آقایان bias دارد. در مورد اول اگر شغل زن را دکتر فرض کنیم، مدل برای مرد شغل روان‌پزشک که شاخه‌ای از پزشکی است را انتخاب می‌کند. اما در آزمایش دوم هنگامی که شغل مرد را دکتر در نظر می‌گیریم، برای شغل زن mum را انتخاب می‌کند. گرچه مادری از جایگاه بالایی برخوردار است اما در اینجا نشان‌دهنده bias بر روی جنسیت می‌باشد.

۲.۱ بررسی بردارهای کلمات مشابه در کلاس‌های مختلف

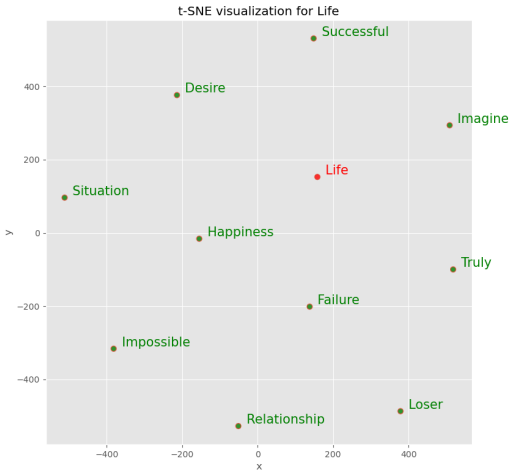
در این بخش بردارهای کلمات مشترک در دو دسته depresion و happiness را مورد بررسی قرار دادیم. در این آزمایش برای بردارها cosine similarity را محاسبه کردیم. اگر دو بردار کاملا یکسان باشند مقدار cosine similarity برابر با ۱ یا نزدیک به ۱ می‌شود و در غیر این صورت مقادیر کوچک تر از ۱ می‌باشد. نتیجه محاسبه cosine similarity برای چند کلمه مشترک بین دسته‌ها به صورت شکل۳ است.

	word	cosine similarity value
0	working	0.236995
1	time	-0.156845
2	able	0.060809
3	good	-0.052721
4	depression	-0.097719
5	life	-0.175595
6	believe	0.161606
7	anxiety	-0.010481
8	human	0.053481
9	beautiful	-0.067325

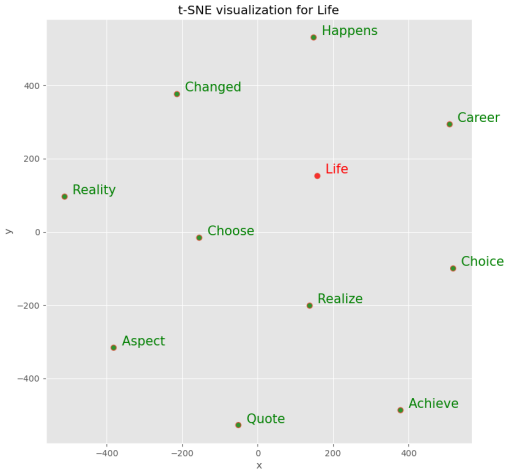
شکل ۳: cosine similarity common words

همان‌طور که مشخص است اکثر کلمات یکسان در دسته‌های مختلف بردارهای متفاوتی دارد. دلیل این امر این است که با توجه به context که کلمه در هر کلاس آمده، بردار مورد نظر به دست آمده است. با توجه به اینکه کلمات در متن‌های متفاوتی هستند پس بردارهای متفاوتی برای آن‌ها وجود دارد.

در ادامه کلمات مشترک کلاس‌های مختلف 10 most similarr words را برای کلمه life بررسی کردیم و بردار ۶۴ بعدی را در نمودار ۲ بعدی نمایش دادیم. ۱۰ کلمه برتر کلاس depresion در شکل۴ و ۱۰ کلمه برتر کلاس happiness در شکل۵ می‌باشد.



شکل ۴: بردار ۱۰ کلمات مشابه life



شکل ۵: بردار ۱۰ کلمات مشابه life

۲ tokenization

در این قسمت ابتدا پنج vocab size برای tokenize کردن داده انتخاب می‌کنیم. دیتا را به پنج بخش تقسیم کرده و در هر مرحله آزمایش، یک بخش به عنوان داده ارزیابی و چهار بخش باقی مانده را به عنوان داده آموزشی در نظر می‌گیریم. به ازای هر vocab size در این بخش tokenize در مرحله word اجرا می‌شود و id توکن unk عدد سه در نظر گرفته شده است. پنج بار آزمایش انجام می‌دهیم. مدل رت روی داده‌های آموزشی train کرده ودر انتها تعداد توکن‌های unk را به ازای vocab size های مختلف بر روی داده ارزیابی بررسی می‌کنیم. نتایج به دست آمده در شکل۶ قابل مشاهده است.

	token count	1	2	3	4	5	average unk token percent
0	60	42.39	42.50	42.58	42.49	42.31	42.454
1	500	25.67	25.47	25.49	25.16	25.40	25.438
2	2000	11.91	11.88	11.76	11.65	11.69	11.778
3	5000	5.71	5.60	5.57	5.50	5.62	5.600
4	10067	2.54	2.47	2.46	2.36	2.36	2.438

شکل ۶: tokenization result

همچنین نتایج به صورت متن در reports/tokenization.txt و log برنامه در logs/tokenization.log موجود است. برای اجرا آزمایش‌ها و ذخیره مدل نهایی در پوشه مورد نظر کافیست python3 tokenization.py اجرا شود.

همان‌طور که انتظار می‌رفت با افزایش تعداد vocab size تعداد توکن‌های unk به مقدار قابل توجهی کاهش می‌یابد.

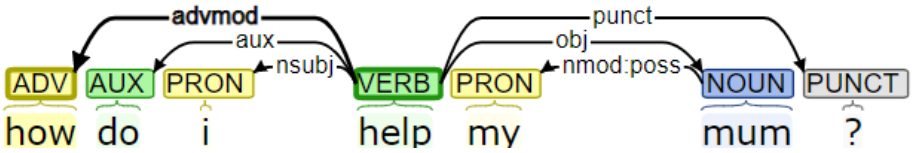
parsing ۳

در این قسمت به کمک کد تمرین ۳ مدل dependency parser را بر روی زبان انگلیسی آموزش داده و تعدادی جمله از دیتاست را انتخاب کرده و parse متناظر با آن‌ها را به صورت دستی نوشته و به عنوان فایل تست، قرار می‌دهیم. فایل تست تولید شده در src/parsing/data/project_data_test.conll قابل مشاهده است (شکل۷).

```
1   1   how _   ADV WRB _   4   aux _   _
2   2   do  _   AUX VBP _   4   nsubj   _   _
3   3   i   _   PRON    PRP _   4   nsubj   _   _
4   4   help    _   VERB    VB  _   0   root    _   _
5   5   my  _   DET VB  _   6   PRP _   _
6   6   mum _   NOUN    NN  _   4   dobj    _   _
7   7   ?   _   PUNCT   .   _   4   punct   _   _
8
```

شکل ۷: correct dependency parser

به عنوان مثال مدل برای جمله موجود در شکل۸ dependency parser را به طور کامل درست تشخیص داده است.



شکل ۸: correct dependency parser

نکته‌ای که در ساخت فایل conll بسیار مهم است و باید بدان توجه شود این است که بین موارد نوشته شده باید یک tab فاصله باشد و در صورت عدم رعایت این فاصله جملات توسط parser به صورت اشتباه خوانده می‌شوند.

۴ language model

در این بخش برای آموزش language model ابتدا داده تمیز را به صورت مناسب آماده می‌کنیم. سپس به ازای هر کدام از دسته‌های depresion و happiness داده را به شبکه داده تا مدل زبانی آموزش ببیند. در معماری تعریف شده ایتدا یک لایه embedding قرار داده شده و در ادامه لایه LSTM با 100 hidden state قرار دارد. لایه دیگری bidirectional LSTM و در ادامه یک لایه dense قرار دارد. لایه انتهایی یک لایه dense با تابع فعال‌سازی softmax می‌باشد که به تعداد همه کلمات موجود نورون دارد. در این لایه به ازای ورودی شبکه مشخص می‌شود چه کلمه باید بعد از عبارت ورودی شبکه بیاید. مدل تعریف شده به صورت شکل۹ می‌باشد. دقت و loss برای کلاس‌های مختلف به صورت شکل۱۰ و شکل۱۱ است.

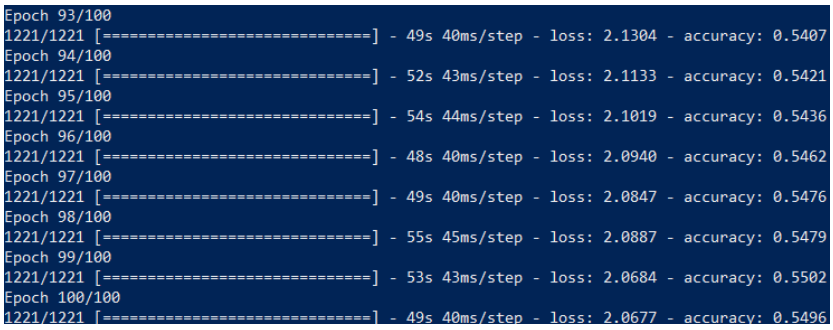
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 10, 50)            499250
_________________________________________________________________
lstm (LSTM)                  (None, 10, 100)           60400
_________________________________________________________________
bidirectional (Bidirectional (None, 200)               160800
_________________________________________________________________
dense (Dense)                (None, 100)               20100
_________________________________________________________________
dense_1 (Dense)              (None, 9985)              1008485
=================================================================
Total params: 1,749,035
Trainable params: 1,749,035
Non-trainable params: 0
```

شکل ۹: شبکه language model

```
Epoch 92/100
1221/1221 [==============================] - 48s 40ms/step - loss: 2.1851 - accuracy: 0.5296
Epoch 93/100
1221/1221 [==============================] - 51s 41ms/step - loss: 2.1756 - accuracy: 0.5306
Epoch 94/100
1221/1221 [==============================] - 55s 45ms/step - loss: 2.1517 - accuracy: 0.5362
Epoch 95/100
1221/1221 [==============================] - 49s 40ms/step - loss: 2.1539 - accuracy: 0.5357
Epoch 96/100
1221/1221 [==============================] - 49s 40ms/step - loss: 2.1635 - accuracy: 0.5344
Epoch 97/100
1221/1221 [==============================] - 53s 43ms/step - loss: 2.1472 - accuracy: 0.5371
Epoch 98/100
1221/1221 [==============================] - 54s 44ms/step - loss: 2.1362 - accuracy: 0.5380
Epoch 99/100
1221/1221 [==============================] - 48s 40ms/step - loss: 2.1314 - accuracy: 0.5390
Epoch 100/100
1221/1221 [==============================] - 49s 40ms/step - loss: 2.1096 - accuracy: 0.5418
```

شکل ۱۰: LSTM accuracy and Loss language model on class depression

با توجه به حذف stopwords و punctuation از جمله و کم‌بودن دیتا برای آموزش یک مدل زبانی مناسب، مشکلاتی در نحوه جمله بندی و قواعد گرامری جمله ساخته شده توسط مدل وجود دارد. عدم وجود I ، am ، are و حروف اضافه‌ای همچون ،thst is در زمان تمیز کردن دیتا باعث شده چنین جملاتی



شکل ۱۱: LSTM accuracy and Loss language model on class happiness

به وجود بیایند. از طرفی شبکه آموزش دیده pretrain نبوده و برای اولین بار آموزش می بیند و برای آموزش بهتر نیاز به دیتا بسیار بیشتری از حجم دیتا فعلی دارد.

- happiness

take break outside lunch feel like everyone office hate think boring since forced office romantic vacation friend friend whole money laying

- happiness

kill prayed god make accident happen year old relationship family know remember im told reminds friend something really eventually wa using

- happiness

i feel so .. like losing grow shell remember hate suck suck fucked cant

- depression

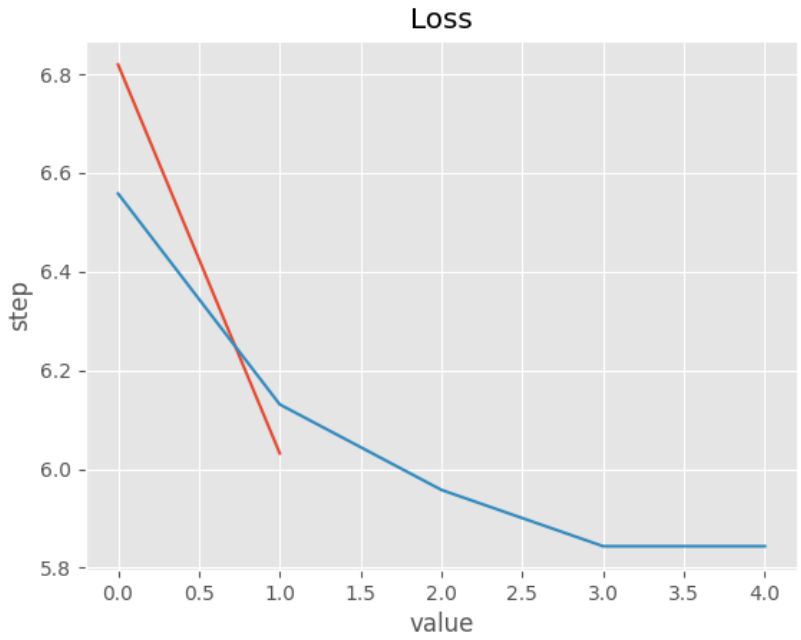
i feel so .. like edgy movie mind hold supportive defense trash sea man

finu tuning ۵

language model ۱.۵

در این بخش با توجه به مطالعات انجام شده بر روی مدل‌های GPT2 در language model و کیفیت خروجی مدل برای این تسک، از distilgpt2 به عنوان مدل pretrain استفاده کرده و مدل را بر دیتاست موجود finetune کردم.

برای اجرا این بخش کافیست python3 fine_tuning.py GPT2 را اجرا کنید. به ازای هر کدام از کلاس‌های depresion و happiness مدل را finetune کرده و وزن‌های حاصله به صورت اتوماتیک در models/depressionGPT2_lm و mosels/happinessGPT2_lm ذخیره می‌شوند. همچنین log در حین اجرا در logs/fine_tuning_GPT2.log قابل مشاهده هست. نمودار تغییرات loss مدل در زمان finetune برای کلاس depression به صورت شکل۱۲ و برای کلاس happiness به صورت شکل۱۳ می‌باشد.



شکل ۱۲: distilgpt2 Loss language model on class depression

شکل ۱۳: distilgpt2 Loss language model on class happiness

جملات تولید شده با مدل از قبل آموزش دیده distilgpt2 بسیار با کیفیت‌ترو معنی‌دارتر از جملات تولید شده در قسمت قبل می‌باشد. به ازای هر کدام از کلاس‌ها دو جمله تولید شده که به صورت زیر می‌باشد. نوشته به رنگ آبی توسط مدل تولید شده است.

- happiness

Hi, we have created a success forum for since forced office romantic vacation friend friend whole money laying

- happiness

people interested interested know Hi, we have created a success forum for going want succeed going want fail people want know go past time never successful people want know go past time know succeeding people want know go past time life lived time lived past moment

- happiness

I'm so depressed. I have nothing to live remember im told reminds friend something really eventually wa using

- depression

like edgy movie mind hold sup- I'm so depressed. I have nothing to live portive defense trash sea man

۲.۵ classification

در بخش دوم مدل bert-base-uncased از قبل آموزش دیده را برای classification داده‌ها بر روی داده‌های موجود finetune می‌کنیم. برای اجرا این بخش کافیست python3 fine_tuning.py Bert را اجرا کرد. وزن‌های مدل finetune شده در models/bert_classification_lm ذخیره می‌شوند. همچنین log در حین اجرا در logs/fine_tuning_Bert.log می‌باشد. نمودار تغییرات loss در شکل۱۴ قابل مشاهده است.

شکل ۱۴: bert-base-uncased Loss classification

۶ منابع

https://www.kaggle.com/pierremegret/gensim-word2vec-tutorial
https://radimrehurek.com/gensim/models/word2vec.html
https://www.philschmid.de/
fine-tune-a-non-english-gpt-2-model-with-huggingface
https://machinelearningmastery.com/
how-to-develop-a-word-level-neural-language-model-in-keras/
https://gmihaila.github.io/tutorial_notebooks/pretrain_
transformers_pytorch/
https://stackoverflow.com/questions/52277384/
calculation-of-cosine-similarity-of-a-single-word-in-2-different-word2vec-models