

بسم الله الرحمن الرحيم

پروژه پایانی پردازش زبان طبیعی

غزاله محمودی

۵ تیر ۱۴۰۰

## فهرست مطالب

۵	word2vec	۱
۶	tokenization	۲
۷	parsing	۳
۸	language model	۴
۱۰	finu tuning	۵
۱۰	language model	۱.۵
۱۲	classification	۲.۵
۱۳	منابع	۶

## فهرست تصاویر

۶	correct dependency parser	۱
۷	correct dependency parser	۲
۷	correct dependency parser	۳
۸	شبکه language model	۴
۹	LSTM accuracy and Loss language model on class depression	۵
۹	LSTM accuracy and Loss language model on class happiness	۶
۱۰	distilgpt2 Loss language model on class depression	۷
۱۱	distilgpt2 Loss language model on class happiness	۸
۱۲	bert-base-uncased Loss classification	۹

## فهرست جداول

## ۱ word2vec

در این بخش قصد داریم با استفاده از ماژول gensim بردار word 2 vec را برای هر کلمه حساب کنیم. ابتدا لازم است دیتاست را برای استفاده آماده کنیم. به کمک pandas و نظرات موجود را استخراج می کنیم. در نهایت DataFrame به صورت زیر حاصل می شود.

در ادامه به ازای تمام نظرات (جملات) موجود در دیتاست کلمات موجود در آن ها را استخراج می کنیم. این کار با استفاده از tokenizer ماژول HAZM انجام می دهیم.

برای به دست آوردن word2vec به کمک Gensim تعدادی پارامتر قابل تنظیم دارد که در ادامه به بررسی آن ها می پردازیم.

### • 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 غالباً کم تکرار می شوند مناسب است. همچنین در مصرف بهینه مموری و حافظه هم تاثیر دارد.

### • Workers

این پارامتر تعداد thread هایی که در عملیات اجرا مورد استفاده قرار می گیرد را مشخص می کند. برای عملیات بهینه سازی و افزایش سرعت اجرا در سیستم هایی که قابلیت پردازش موازی دارند.

### • Iter

تعداد epoch یا دفعاتی که الگوریتم اجرا می شود و مدل آموزش می بیند.

### • seed

برای مقدار دهی رندوم اولیه استفاده می شود.

## 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 پنج بار آزمایش را تکرار می‌کنیم. که در هر مرحله یک بخش به عنوان داده ارزیابی و چهار بخش باقی مانده را به عنوان داده آموزشی در نظر می‌گیریم. در این بخش tokenize در مرحله word اجرا می‌شود و id توکن unk به دستی عدد سه در نظر گرفته شده است. در انتها تعداد توکن‌های unk را به ازای vocab size های مختلف بررسی می‌کنیم. نتایج به دست آمده در شکل ۱ قابل مشاهده است. با هر بار اجرا مجدد کد در صورتی که تغییری در نتایج به وجود بیاید شکل ۱ به صورت اتوماتیک آپدیت می‌شود.

	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 ۱: correct dependency parser

همچنین نتایج به صورت متن در reports/tokenization.txt و log برنامه در logs/tokenization.log موجود است. همان‌طور که مشاهده می‌شود در

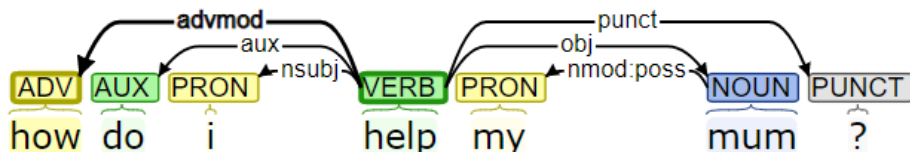
## ۳ parsing

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 در شکل ۲ قابل مشاهده است.

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 ۲: correct dependency parser

به عنوان مثال مدل برای جمله موجود در شکل ۳ dependency parser را به طور کامل درست تشخیص داده است.



شکل ۳: 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 depression و happiness داده را به شبکه داده تا مدل زبانی آموزش ببیند. در معماری تعریف شده ابتدا 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 یا تابع فعال‌سازی softmax می‌باشد که به تعداد همه کلمات موجود نوروں دارد. در این لایه به ازای ورودی شبکه مشخص می‌شود چه کلمه باید بعد از عبارت ورودی شبکه بیاید. مدل تعریف شده به صورت شکل ۴ می‌باشد. دقت و loss برای کلاس‌های مختلف به صورت شکل ۵ و شکل ۶ است. با توجه به حذف stopwords و punctuation از جمله و کم‌بودن مشکلاتی در جمله بندی مدل آموزش دیده وجود دارد که در جملات ساخته شده به وضوح مشخص است. عدم وجود I ، am ، are و حروف اضافه‌ای همچون thst is، در زمان تمیز کردن دیتا باعث شده چنین جملاتی به وجود بیایند.

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 ۴: شبکه language model

- 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



```

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 ۵: LSTM accuracy and Loss language model on class depression

```

Epoch 93/100
1221/1221 [=====] - 49s 40ms/step - loss: 2.1304 - accuracy: 0.5407
Epoch 94/100
1221/1221 [=====] - 52s 43ms/step - loss: 2.1133 - accuracy: 0.5421
Epoch 95/100
1221/1221 [=====] - 54s 44ms/step - loss: 2.1019 - accuracy: 0.5436
Epoch 96/100
1221/1221 [=====] - 48s 40ms/step - loss: 2.0940 - accuracy: 0.5462
Epoch 97/100
1221/1221 [=====] - 49s 40ms/step - loss: 2.0847 - accuracy: 0.5476
Epoch 98/100
1221/1221 [=====] - 55s 45ms/step - loss: 2.0887 - accuracy: 0.5479
Epoch 99/100
1221/1221 [=====] - 53s 43ms/step - loss: 2.0684 - accuracy: 0.5502
Epoch 100/100
1221/1221 [=====] - 49s 40ms/step - loss: 2.0677 - accuracy: 0.5496

```

شکل ۶: LSTM accuracy and Loss language model on class happiness

depression •

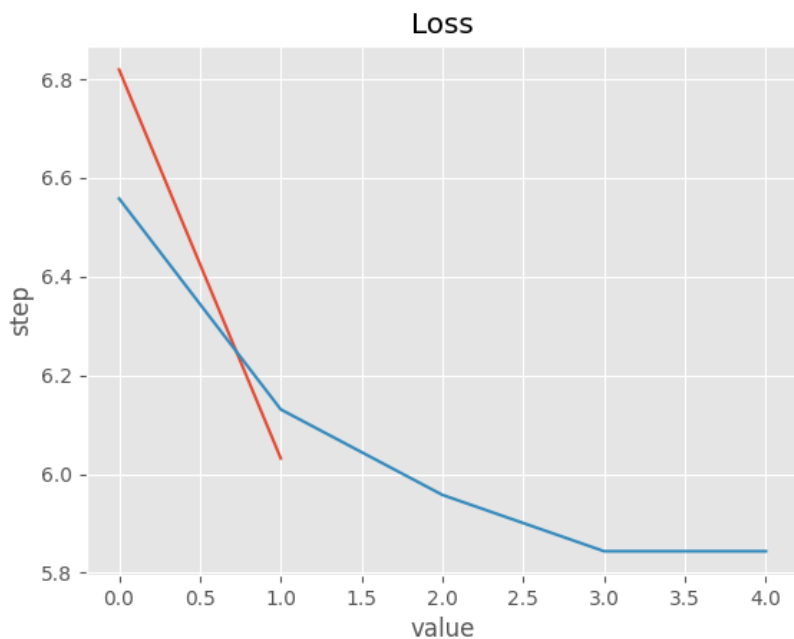
i feel so .. like edgy movie mind hold supportive defense trash sea man

## ۵ finu tuning

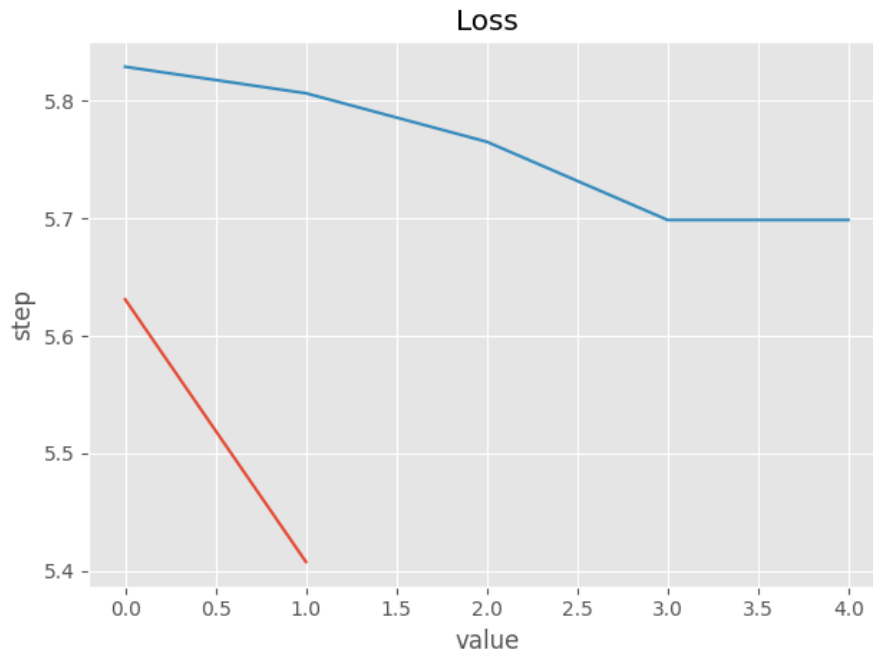
### ۱.۵ language model

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 کافیسیت GPT2 python3 fine\_tuning.py را اجرا کنید. به ازای هر کدام از کلاس‌های depression و happiness مدل را finetune کرده و وزن‌های حاصله به صورت اتوماتیک در mosels/happinessGPT2\_lm و models/depressionGPT2\_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 شکل ۷ و برای کلاس happiness به صورت شکل ۸ می‌باشد.



شکل ۷: distilgpt2 Loss language model on class depression



شکل ۸: distilgpt2 Loss language model on class happiness

جملات تولید شده با مدل از قبل آموزش دیده distilgpt2 بسیار با کیفیت‌تر و 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 کافیسست Bert `python3 fine_tuning.py` 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 شکل ۹ قابل مشاهده است.



شکل ۹: bert-base-uncased Loss classification

## ٦ منابع

<https://radimrehurek.com/gensim/models/word2vec.html>  
[https://www.philschmid.de/  
fine-tune-a-non-english-gpt-2-model-with-huggingface](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://machinelearningmastery.com/how-to-develop-a-word-level-neural-language-model-in-keras/)  
[https://gmihaila.github.io/tutorial\\_notebooks/pretrain\\_  
transformers\\_pytorch/](https://gmihaila.github.io/tutorial_notebooks/pretrain_transformers_pytorch/)