

بسم الله الرحمن الرحيم

پروژه پایانی پردازش زبان طبیعی

غزاله محمودی

۶ تیر ۱۴۰۰

فهرست مطالب

۴	word2vec	۱
۴ بررسی bias	۱.۱
۶ بررسی بردارهای کلمات مشابه در کلاس‌های مختلف	۲.۱
۸	tokenization	۲
۹	parsing	۳
۱۰	language model	۴
۱۲	finu tuning	۵
۱۲ language model	۱.۵
۱۴ classification	۲.۵
۱۵	نحوه اجرا پروژه	۶
۱۶	منابع	۷

فهرست تصاویر

۵	woman as doctor similar as man as ?	۱
۵	man as doctor similar to woman as ?	۲
۶	cosine similarity common words	۳
۷	بردار ۱۰ کلمات مشابه life	۴
۷	بردار ۱۰ کلمات مشابه life	۵
۸	tokenization result	۶
۹	correct dependency parser	۷
۹	correct dependency parser	۸
۹	correct dependency parser	۹
۱۰	شبکه language model	۱۰
۱۰	..	LSTM accuracy and Loss language model on class depression	۱۱
۱۱	..	LSTM accuracy and Loss language model on class happiness	۱۲
۱۲	distilgpt2 Loss language model on class depression	۱۳
۱۳	distilgpt2 Loss language model on class happiness	۱۴
۱۴	bert-base-uncased Loss classification	۱۵

۱ word2vec

در این بخش قصد داریم با استفاده از ماژول gensim بردار word2vec را برای هر کلمه حساب کنیم. 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 سائز vector برای نمایش هر word یا token است. هر چه دیتاست محدود تر و کوچک تر باشد این عدد نیز کوچک تر در نظر گرفته می شود و هر چه دیتاست بزرگ تر باشد (کلمات unique بیشتری داشته باشد) باید اندازه vector بزرگ تر 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 سائز کوچک تر باشد کلماتی که بیشترین ارتباط معنایی را به یکدیگر دارند به عنوان خروجی برمی گرداند. اگر تعداد داده به اندازه کافی بزرگ باشد سائز پنجره اهمیت زیادی ندارد اما باید این نکته را در نظر گرفت که این سائز 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.

۱.۱ بررسی bias

در این بخش احتمال وجود bias در دیتاست مورد بررسی قرار گرفت. برای این منظور مشابهت های مقابل مورد بررسی قرار گرفت.

woman as doctor similar as man as ?

که خروجی به صورت شکل ۱ شد.

	word	score
0	psychiatrist	0.539148
1	therapist	0.494201
2	sent	0.490427
3	appointment	0.451637
4	symptom	0.440813
5	medication	0.419075
6	nearly	0.415111
7	ward	0.413921
8	adhd	0.403657
9	med	0.399072

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 شکل ۲ شد.

man as doctor similar to woman as ?

	word	score
0	mum	0.499389
1	sister	0.489097
2	dad	0.462778
3	upset	0.453062
4	therapist	0.451369
5	mad	0.449687
6	phone	0.448865
7	anyway	0.448734
8	psychiatrist	0.442708
9	appointment	0.441602

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.

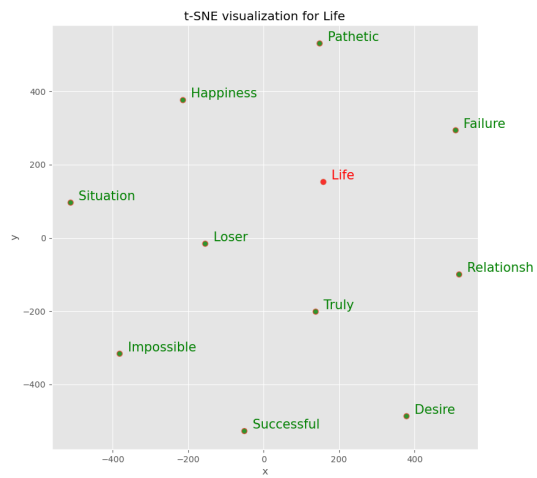
۲.۱ بررسی بردارهای کلمات مشابه در کلاس‌های مختلف

در این بخش بردارهای کلمات مشترک در دو دسته happiness و depression را مورد بررسی قرار دادیم. در این آزمایش برای بردارها cosine similarity را محاسبه کردیم. اگر دو بردار کاملاً یکسان باشند مقدار cosine similarity برابر با ۱ یا نزدیک به ۱ می‌شود و در غیر این صورت مقادیر کوچک‌تر از ۱ می‌باشد. نتیجه محاسبه cosine similarity برای چند کلمه مشترک بین دسته‌ها به صورت شکل ۳ است.

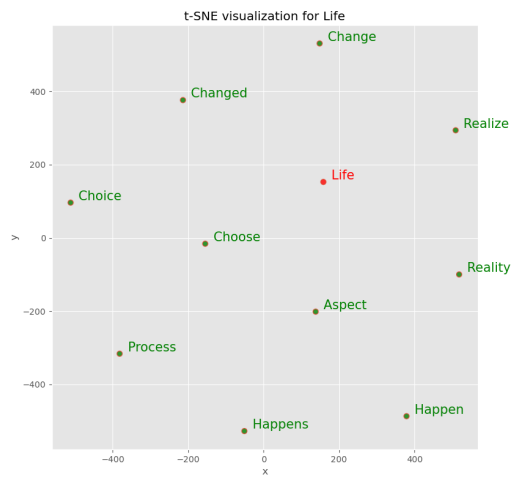
	word	cosine similarity value
0	working	0.195335
1	time	-0.091545
2	able	0.124327
3	good	-0.018955
4	depression	-0.106198
5	life	-0.169888
6	believe	0.118834
7	anxiety	-0.049866
8	human	0.090089
9	beautiful	-0.075464

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 بررسی 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 depression در شکل ۴ و ۱۰ کلمه برتر کلاس happiness در شکل ۵ 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 و در انتها تعداد توکن‌های unk را به ازای vocab size های مختلف بر روی داده ارزیابی بررسی می‌کنیم. نتایج به دست آمده در شکل ۶ 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
python tokenization/tokenization.py
اجرا شود.
همان‌طور که انتظار می‌رفت با افزایش تعداد vocab size تعداد توکن‌های unk به مقدار قابل توجهی کاهش می‌یابد.

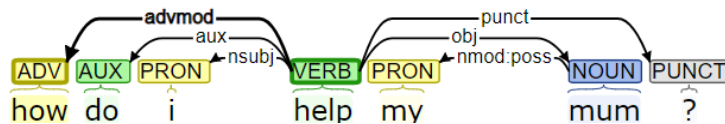
۳ parsing

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 src/parsing/run.py دستور python میسر می‌باشد.

1	1	how	ADV	WRB	4	aux	_	_
2	2	do	AUX	VP	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								

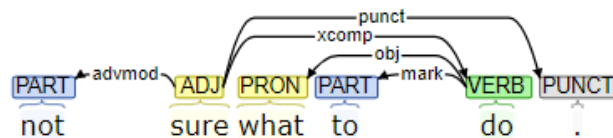
شکل ۷: correct dependency parser

به عنوان مثال مدل برای جمله موجود در شکل ۸ 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. پس از اجرا تست UAS: 80.00 test به دست آمد. مدل در جملات پیچیده و گاهی با دو فعل (و یا جملات ساده‌ای که در داده‌های آموزشی نباشند) در تشخیص root دچار مشکل شده و اشتباهات به صورت سلسله‌وار به وجود می‌آید. به عنوان مثال در جمله "not sure what to do." واژه not را به عنوان root در نظر گرفته در صورتی که پارس صحیح جمله به صورت شکل ۹ است.



شکل ۹: correct dependency parser

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 با 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 ۱۰ می‌باشد. دقت و loss برای کلاس‌های مختلف به صورت شکل ۱۱ و شکل ۱۲ 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 ۱۰: 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 ۱۱: 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 is، that در زمان تمیز کردن دیتا باعث شده چنین جملاتی

```

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 ۱۲: 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 •

think about death serving obvious interesting tends sleeper know go with-
out please ha

depression •

think about death amp feeding oct girl last last two including rejoined
member

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

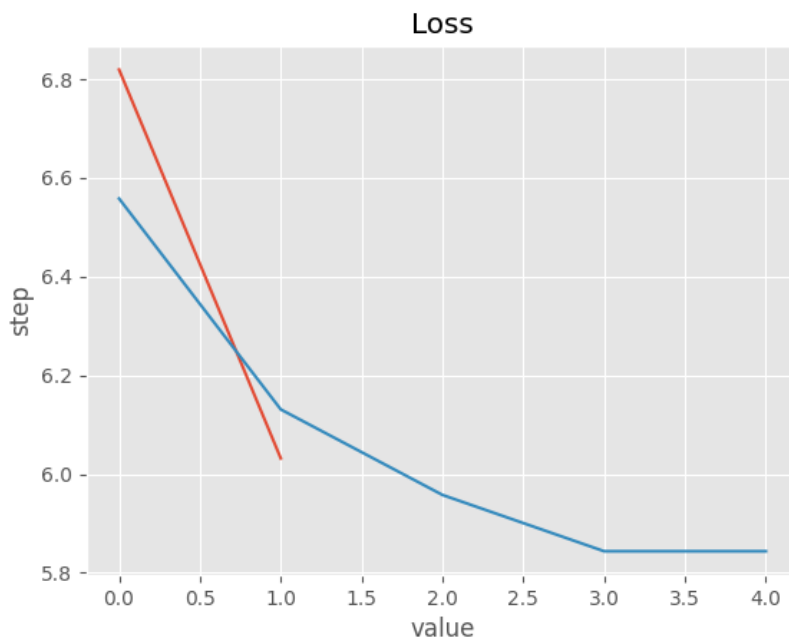
برای اجرا این بخش کافیت python language_model.py را اجرا کنید. به ازای هر کدام از کلاس‌های happiness و depression مدل را finetune کرده و وزن‌های حاصله به صورت اتوماتیک در models/language_model/depression_language_model.h5 و models/language_model/happiness_language_model.h5 ذخیره می‌شوند. همچنین log در حین اجرا در logs/language_model.log قابل مشاهده هست.

۵ finu tuning

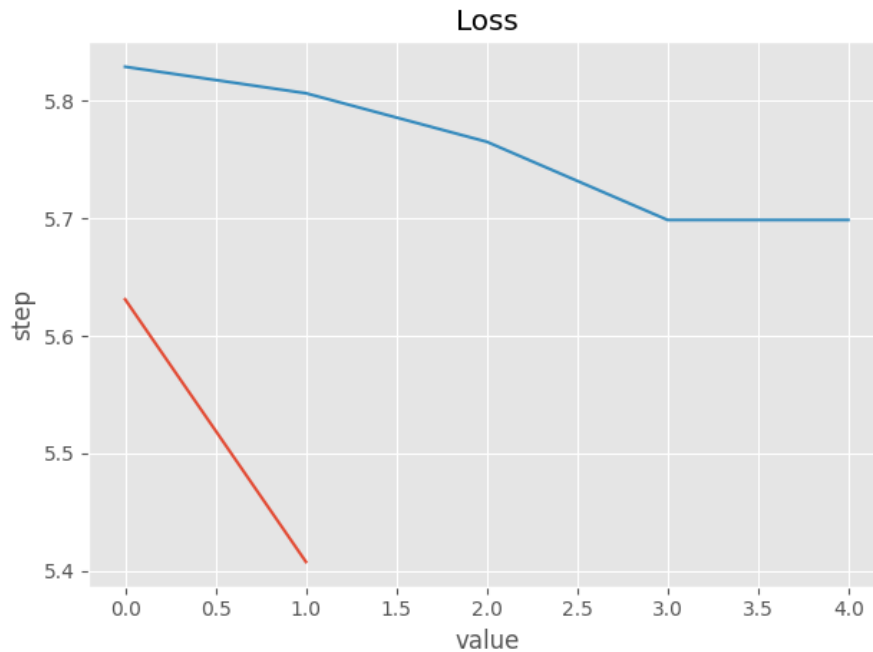
۱.۵ language model

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 python 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 ۱۳ و برای کلاس happiness به صورت شکل ۱۴ می‌باشد.



شکل ۱۳: distilgpt2 Loss language model on class depression



شکل ۱۴: 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 people interested interested know
 help support success success want like like get rich life know want give
 give want give back give want give know want give take

depression •

Hi, we have created a success forum for people interested interested know
 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 livepeople like that time ago and
 everything felt normal today really worked like ever since even though
 everyone else would take back everything back everything right moment
 every morning would feel like day today felt normal today feel

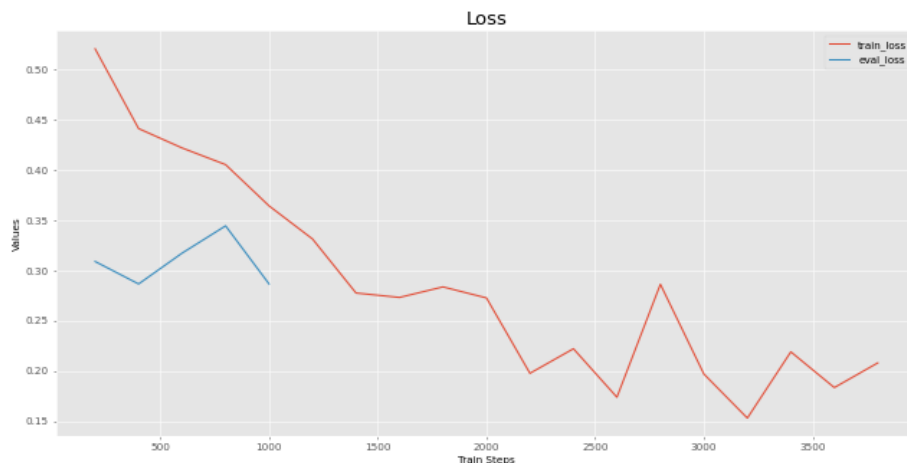
depression •

I'm so depressed. I have nothing to live like two people say like crazy thing
say everything said nothing said anything anything said anything actually
meant anything else meant anything meant nothing meant anything any-
thing meant anything else never truly meant anything know want need
understand needed know

نکته قابل تامل این است که حجم دیتا depression نسبت به happiness اندکی بیشتر است و به نظر می‌رسد این موضوع نحوه آموزش تاثیر گذاشته است. همچنین عملکرد مدل GPT برای ساخت جملات طولانی بهتر از LSTM می‌باشد.

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 کافیت Bert python 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 ۱۵ قابل مشاهده است.



شکل ۱۵: bert-base-uncased Loss classification

نمونه خروجی classification به ازای ۴ جمله ورودی در logs/fine_tuning_Bert.log موجود می‌باشد.

۶ نحوه اجرا پروژه

برای آماده‌سازی کامپیوتر شخصی برای اجرا پروژه ابتدا لازم است با اجرا دستور زیر envirement مورد نیاز را ساخته و آن را فعال کنیم.

```
conda env create -f local__env.yml  
conda activate project__env
```

در ادامه برای اجرا کلیه مراحل فاز پایانی پروژه کافیت کامند زیر را اجرا کنید. اجرا در سیستم عامل لینوکس با دستور

```
sh run.sh
```

و بر روی ویندوز با دستور

```
(windows) run.bat
```

می‌باشد. راهنمای اجرا هر بخش از پروژه در قسمت مربوط به آن موجود می‌باشد.

٧ منابع

<https://www.kaggle.com/pierremegret/gensim-word2vec-tutorial>
<https://radimrehurek.com/gensim/models/word2vec.html>
<https://www.philtschmid.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>
<http://stanza.run/>