

رامتین احسانی – سینا اسکندری

لینک گیت هاب: <https://github.com/ramtin-ehsani/Movie-Genre-NLP>

فاز نهایی پروژه درس مبانی پردازش زبان و گفتار

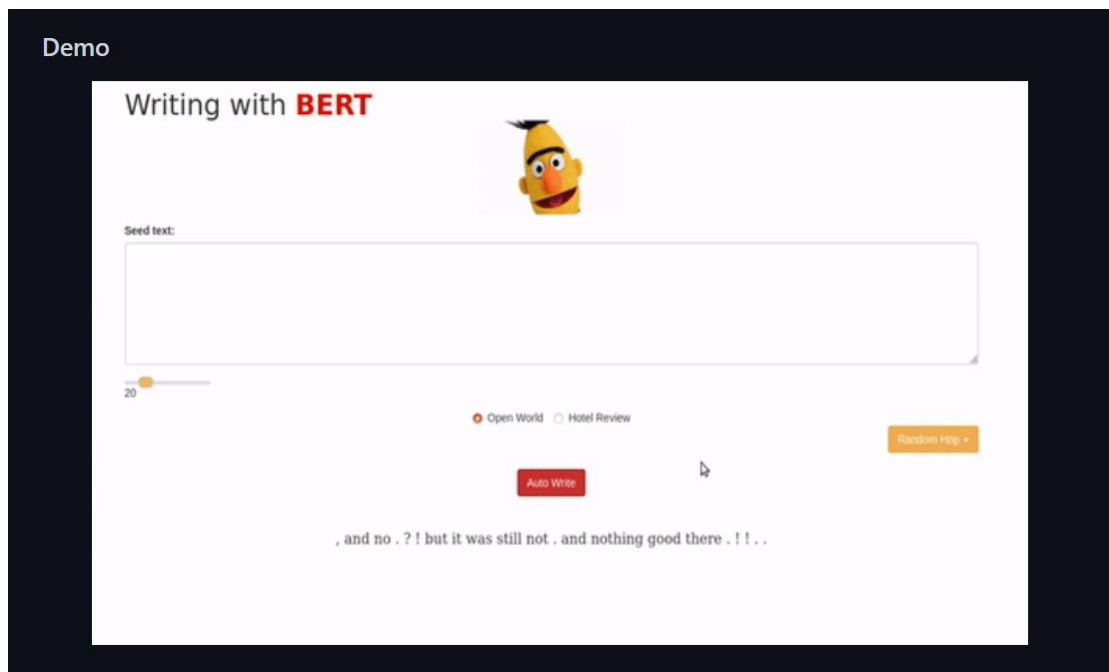
بخش اول: تولید جملات

برای این بخش، از آن جایی که دیتای ما به زبان انگلیسی میباشد از مدل pretrained شده BERT استفاده کردیم.

به این صورت که مدل را در کد لود کردیم و بر روی دیتای از پیش تهیه شده خودمان train کردیم. پس از به پایان رسیدن آموزش و تست این مدل، مدل نهایی را در drive ذخیره کردیم.

برای بخش تولید جملات از پروژه زیر کمک گرفتیم:

<https://github.com/prakhar21/Writing-with-BERT>



تمامی فایل های تولید شده توسط کد خودمان را به پروژه بالا میدهم و با استفاده از مدل trained شده ما، این کد شروع به تولید جملات میکند.

لینک مدل و سایر موارد تولید شده در این مرحله:

<https://drive.google.com/drive/folders/1ZEW6xOXzSCHRDrdU2RG4F5KzFy9CGwCM?usp=sharing>

لینک گوگل کولب:

<https://colab.research.google.com/drive/1vwxrLCMafSY0dWWNMeZ3EEUEjNAJVv6s?usp=sharing>

آموزش و evaluation مدل BERT:

```
[ ] !python3 finetune_on_pregenerated.py --pregenerated_data training/ --bert_model bert-base-uncased --do_lower_case --train_batch_size 16 --output_dir finetuned_lm/ --epochs 2

2022-07-08 21:10:29,330: device: cuda n_gpu: 1, distributed training: False, 16-bits training: False
2022-07-08 21:10:30,276: loading file https://s3.amazonaws.com/models.huggingface.co/bert/bert-base-uncased-vocab.txt from cache at /root/.cache/torch/pytorch_transformers/26bc
2022-07-08 21:10:31,273: https://s3.amazonaws.com/models.huggingface.co/bert/bert-base-uncased-config.json not found in cache or force_download set to True, downloading to /tmp
100% 433/433 [00:00<00:00, 401444.22B/s]
2022-07-08 21:10:32,239: copying /tmp/tmpkafgo8vi to cache at /root/.cache/torch/pytorch_transformers/4dad0251492946e18ac39290fcfe91b89d370fee250efe9521476438fe8ca185.7156163d5
2022-07-08 21:10:32,240: creating metadata file for /root/.cache/torch/pytorch_transformers/4dad0251492946e18ac39290fcfe91b89d370fee250efe9521476438fe8ca185.7156163d5fdc189c301
2022-07-08 21:10:32,240: removing temp file /tmp/tmpkafgo8vi
2022-07-08 21:10:32,240: loading configuration file https://s3.amazonaws.com/models.huggingface.co/bert/bert-base-uncased-config.json from cache at /root/.cache/torch/pytorch_t
2022-07-08 21:10:32,240: Model config {
  "architectures": [
    "BertForMaskedLM"
  ],
  "attention_probs_dropout_prob": 0.1,
  "finetuning_task": null,
  "hidden_act": "gelu",
  "hidden_dropout_prob": 0.1,
  "hidden_size": 768,
  "initializer_range": 0.02,
  "intermediate_size": 3072,
  "layer_norm_eps": 1e-12,
  "max_position_embeddings": 512,
  "model_type": "bert",
  "num_attention_heads": 12,
  "num_hidden_layers": 12,
  "num_labels": 2,
  "output_attentions": false,
  "output_hidden_states": false,
```

```
trainer.train()

/usr/local/lib/python3.7/dist-packages/transformers/optimization.py:310: FutureWarning: This implementation
FutureWarning,
***** Running training *****
  Num examples = 24930
  Num Epochs = 5
  Instantaneous batch size per device = 8
  Total train batch size (w. parallel, distributed & accumulation) = 8
  Gradient Accumulation steps = 1
  Total optimization steps = 15585

[15585/15585 1:10:26, Epoch 5/5]



| Epoch | Training Loss | Validation Loss | F1       | Roc Auc  | Accuracy |
|-------|---------------|-----------------|----------|----------|----------|
| 1     | 0.294200      | 0.250590        | 0.674595 | 0.780176 | 0.253109 |
| 2     | 0.249000      | 0.201003        | 0.761928 | 0.837103 | 0.358002 |
| 3     | 0.201800      | 0.162779        | 0.823978 | 0.881059 | 0.474569 |
| 4     | 0.170500      | 0.130574        | 0.869138 | 0.908086 | 0.580064 |
| 5     | 0.145900      | 0.119198        | 0.886015 | 0.921223 | 0.626314 |



***** Running Evaluation *****
  Num examples = 24930
  Batch size = 8
  Saving model checkpoint to bert-finetuned-sem_eval-english/checkpoint-3117
```

نتایج حاصل شده:

20

☐ Open World ☒ Hotel Review

Random Hop ▾

Auto Write

available online . available . . available with an all online access free . a streaming video version online only .

20

☐ Open World ☒ Hotel Review

Random Hop ▾

Auto Write

review site star reviews , was also considered for being considered a online magazine / social content dating portal " .

دقت شود که اگر با گزینه های جایگزین مانند GPT2 و LM مبتنی بر LSTM/GRU مقایسه شود، BERT برای کارهایی مانند NLG مناسب نخواهد بود، فقط به این دلیل که در وهله اول به سبک معمولی رگرسیون خودکار آموزش داده نشده است. BERT به عنوان مدل زبان ماسک شده (MLM) در سبک دو جهته آموزش داده شده است. در MLM به جای پیش بینی هر نشانه بعدی، درصدی از نشانه های ورودی به صورت تصادفی پوشانده می شوند و تنها آن نشانه ها بر اساس کلمات باقی مانده در سمت چپ و راست آن پیش بینی می شوند، که به آن زمینه دو سویه غنی می دهد.

برای اجرای کد، فایل مدل ها (nlp) را از درایو دانلود کرده و در دایرکتوری قسمت BERT قرار دهید. سپس app.py را اجرا کرده و با استفاده از فایل index.html به سرور request بزنید و جمله تولید کنید.

بخش دوم:

لینک گوگل کولب:

<https://colab.research.google.com/drive/1SWUvSaA2pqashAqa7xxalvcuSKBRWxOj?usp=sharing>

برای این بخش از یک مدل ساده dense استفاده کردیم که در شکل زیر قابل مشاهده می باشد:

```
[19] print(model.summary())
```

Model: "sequential"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 512)	15771136
dense_1 (Dense)	(None, 256)	131328
dense_2 (Dense)	(None, 29)	7453

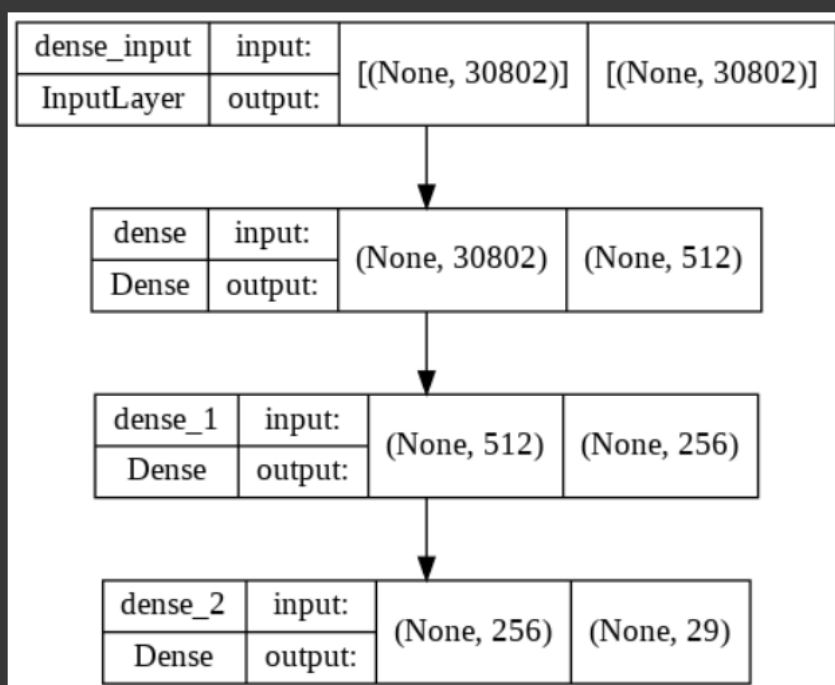
Total params: 15,909,917

Trainable params: 15,909,917

Non-trainable params: 0

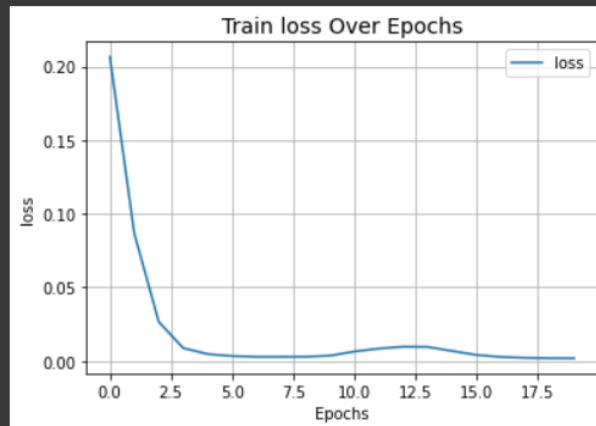
None

```
[22] from keras.utils.vis_utils import plot_model
      plot_model(model, to_file='model_plot4a.png', show_shapes=True, show_layer_names=True)
```

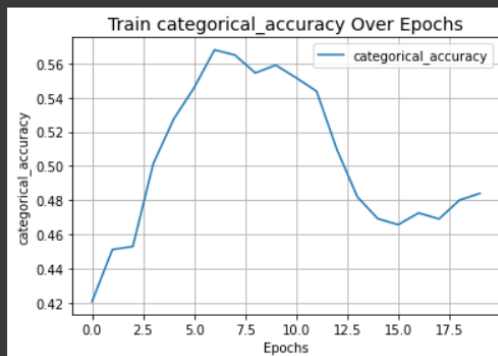


از آن جایی که از دیتای **crawl** شده و تهیه شده توسط خودمان در فاز های قبلی استفاده کردیم، و همینطور به دلیل اینکه کلاس های کتگوری برای این پروژه زیاد هستند و دیتای ما محدود، به طبع انتظار زیادی از این مدل در **Evaluation** نداریم. با این حال، این مدل حدود 35 درصد **accuracy** به ما میدهد:

```
[23] plt.plot(history.history['loss'], label='loss')
      plt.xlabel("Epochs")
      plt.ylabel('loss')
      plt.title("Train {} Over Epochs".format('loss'), fontsize=14)
      plt.legend()
      plt.grid()
      plt.show()
```



```
[24] plt.plot(history.history['categorical_accuracy'], label='categorical_accuracy')
      plt.xlabel("Epochs")
      plt.ylabel('categorical_accuracy')
      plt.title("Train {} Over Epochs".format('categorical_accuracy'), fontsize=14)
      plt.legend()
      plt.grid()
      plt.show()
```



```
_, categorical_acc = model.evaluate(test_dataset)
print(f"Categorical accuracy on the test set: {round(categorical_acc * 100, 2)}%")
```

```
49/49 [=====] - 1s 10ms/step - loss: 0.7119 - categorical_accuracy: 0.3571
Categorical accuracy on the test set: 35.71%.
```

کدهای این بخش در فایل Part2 قرار دارند.

بخش سوم:

برای بهبود مدل طراحی شده، دو راه پیاده سازی را پیش گرفتیم. استفاده از LSTM، و استفاده از CNN.

روش LSTM:

```
[15] print(model.summary())

Model: "sequential"

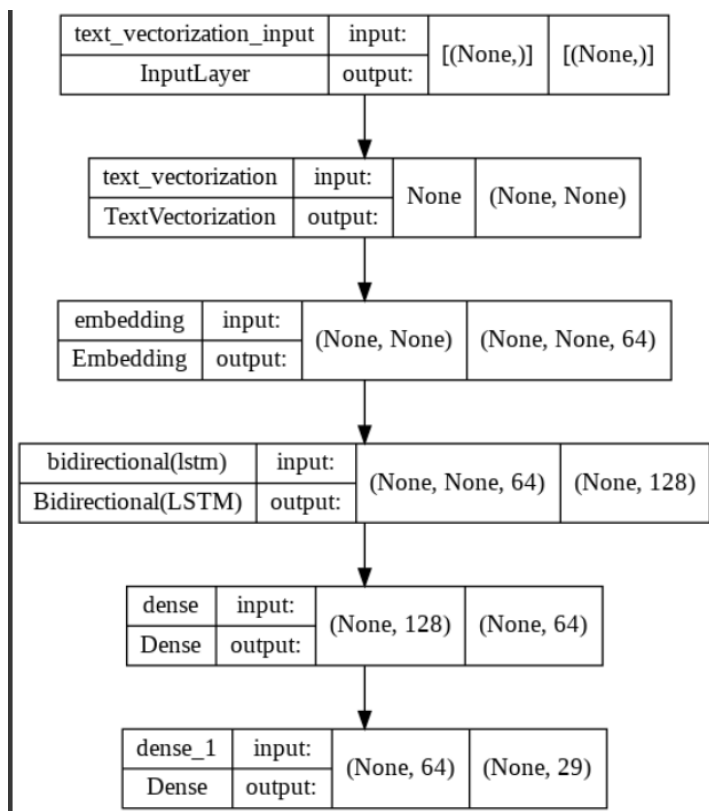
```

Layer (type)	Output Shape	Param #
text_vectorization (TextVectorization)	(None, None)	0
embedding (Embedding)	(None, None, 64)	1966400
bidirectional (Bidirectional)	(None, 128)	66048
dense (Dense)	(None, 64)	8256
dense_1 (Dense)	(None, 29)	1885

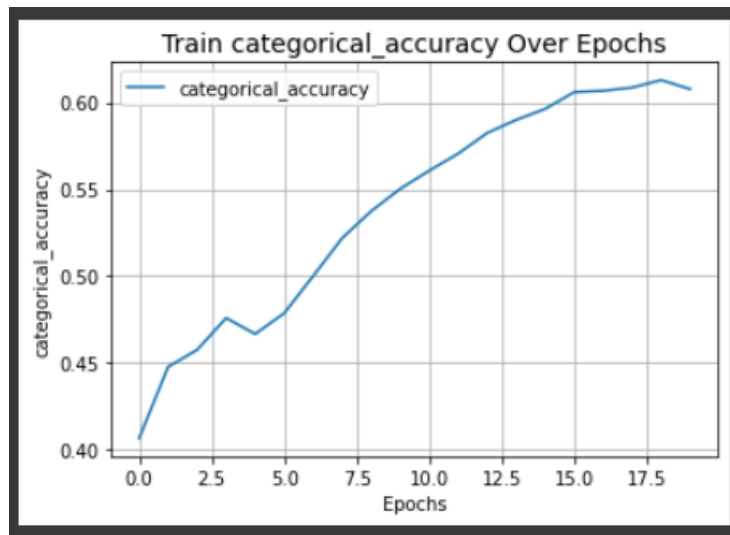
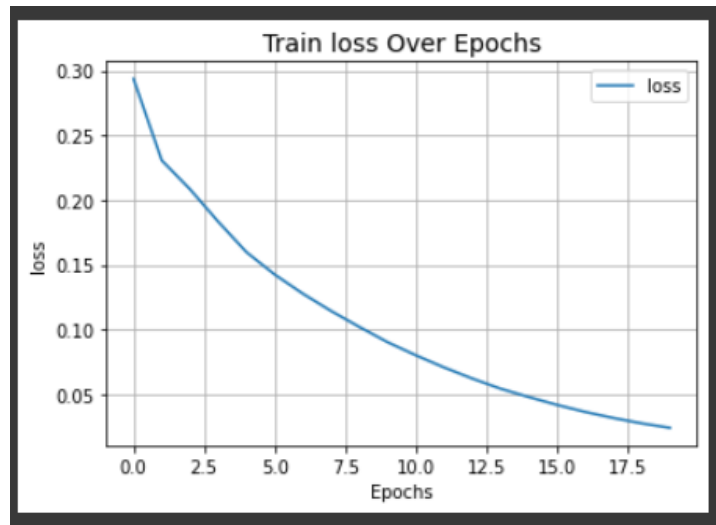
```

Total params: 2,042,589
Trainable params: 2,042,589
Non-trainable params: 0
None

```



نتایج بدست آمده از مدل طراحی شده با lstm:



```
loss_metric, categorical_acc, f1_metric, precision_metric, recall_metric = test_metrics  
print(f"Categorical accuracy on the test set: {round(categorical_acc, 2)}")  
print(f"f1 metric on the test set: {f1_metric}")  
print(f"precision on the test set: {precision_metric}")  
print(f"recall on the test set: {recall_metric}")
```

```
49/49 [=====] - 3s 8ms/step -  
Categorical accuracy on the test set: 34.85%.  
f1 metric on the test set: 0.46216341853141785.  
precision on the test set: 0.48089316487312317.  
recall on the test set: 0.4451489746570587.
```

همانطور که مشخص است همچنان پیشرفت خاصی بر روی نتایج مدل مشاهده نمیکنیم.

فایل گوگل کولب برای روش lstm:

<https://colab.research.google.com/drive/1Uq-vcH1fv-ePdpEeW-T-xkdhTIN6OiQR?usp=sharing>

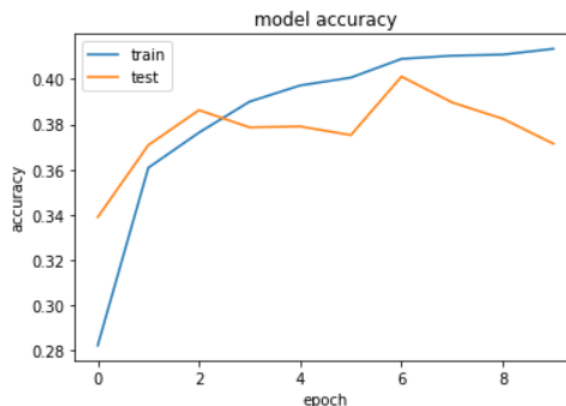
روش CNN:

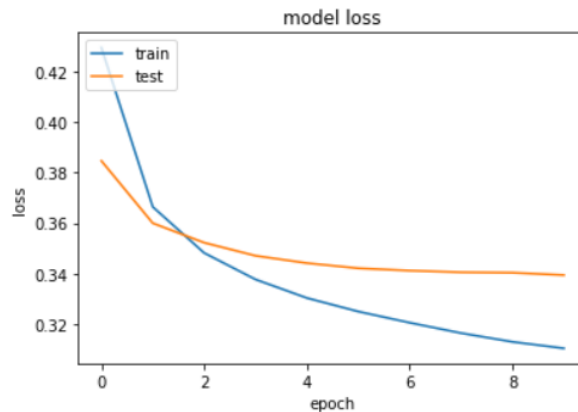
در این روش ابتدا ابتدا لایه Embedding را به یک لایه Conv1D وارد می کنیم و سپس از Average Pooling و Max Pooling همزمان استفاده می کنیم و این ۲ را در انتها به هم متصل می کنیم. نتیجه آخرین لایه concatenate (None, 13) است که به لایه خروجی داده می شود.

Model: "model_2"

Layer (type)	Output Shape	Param #	Connected to
input_3 (InputLayer)	[(None, 200)]	0	
embedding_2 (Embedding)	(None, 200, 100)	3727600	input_3[0][0]
conv1d_2 (Conv1D)	(None, 198, 64)	19264	embedding_2[0][0]
global_average_pooling1d_2 (GlobalAveragePooling1D)	(None, 64)	0	conv1d_2[0][0]
global_max_pooling1d_2 (GlobalMaxPooling1D)	(None, 64)	0	conv1d_2[0][0]
concatenate_2 (Concatenate)	(None, 128)	0	global_average_pooling1d_2[0][0] global_max_pooling1d_2[0][0]
dense_2 (Dense)	(None, 13)	1677	concatenate_2[0][0]

Total params: 3,748,541
Trainable params: 20,941
Non-trainable params: 3,727,600





195/195 [=====] - 0s 2ms/step - loss: 0.3404 - acc: 0.3640
 Test Score: 0.34044456481933594
 Test Accuracy: 0.3640301525592804

باز هم پیشرفت خوبی در نتایج حاصل نمیشود.

دلیل اصلی این موضوع زیاد بودن کتگوری های ژانر های فیلم ها و محدود بودن دیتا در این زمینه میباشد.
 چیزی حدود حداقل ۱۳ تا ژانر متفاوت در دیتا وجود دارد و هر فیلم حداقل ۲ یا ۳ ژانر دارد و طبیعی است که مدل ما به دیتای بیشتری در این زمینه احتیاج داشته باشد.
 کدهای این بخش در فایل Part3 قرار دارند.