




# گزارش پروژه

Deep learning

یاسمین مدنی

پریسا ظفیری



## فهرست

۲	..... مقدمه
۳	.....Fine tune BERT model
۳	..... Word Embedding
۵	.....Data Set
۶	..... Model
۷	..... Inference

## مقدمه

در این پروژه قرار است یک سیستم پرسش و پاسخ هوشمند (single turn FAQ) با توجه به داده های شرکت همراه اول طراحی شود که بتواند سوالات مشتریان را به نحو مناسبی پاسخ دهد. یک سری سوال ورودی و خروجی وجود دارد که جواب هر سوال مشخص است. اگر سوال جدید پرسیده شد باید تشخیص داده شود که به کدام سوال نزدیکتر است و جواب همان سوال به عنوان جواب سوال جدید نیز در نظر گرفته شود. مثالی از مجموعه داده ها:

## سوالات ورودی:

چگونه می توان از موفقیت آمیز بودن خرید اینترنتی مطمئن گردید؟

وضعیت خرید اینترنتی رو چگونه باید مشاهده کرد؟

## پاسخ:

پس از انجام خرید سیم کارت از طریق فروشگاه آنلاین همراه اول، می توانید با شماره ای که با آن اقدام به خرید سیم کارت نموده اید وارد حساب کاربری خود شده و در بخش سفارشات من وضعیت خرید را بررسی نمایید.

از آنجا که نمیتوانیم جملات را به طور ناگهانی آنها را به شبکه بدهیم نیاز به پیش پردازش هایی برای تبدیل آنها به بردار های عددی داریم. برای به دست آوردن embedding جملات از مدلها و روشهای مختلف از جمله parsBert میتوان استفاده کرد. ابتدا پکیج های مورد را نصب و import کرده و سپس دیتای داده شده که شامل یک فایل اکسل از اطلاعات مورد نیاز بود را میخوانیم.

```
▼ Load Data

1 path = '/content/drive/MyDrive/Extension1.xlsx'
2 df = pd.read_excel(path, header=None, names=['question', 'answer'])

Run cell (Ctrl+Enter)
cell executed since last change

[6] = df['answer'].unique()
executed by yasmin madani
2:23 PM (5 minutes ago)
```

از آنجا که تعداد جواب ها است که در این سوال محدود است ابتدا جواب های احتمالی را به صورت زیر به دست میآوریم.

```
candidate_answers = df['answer'].unique()
```

به عبارتی به تعداد اعضای این آرایه کلاس داریم.

## Fine tune BERT model

مدل برت روی داده‌هایی عمومی‌ترین شده است از این رو ما نیاز داریم تا آنرا بر روی داده‌های خودمان مقداری منطبق کنیم. از این رو جملات سوال و جواب را به هم پیوند داده و برای این مرحله نیاز به توکنایز کردن این مجموعه‌ای که ساختیم داریم. که به صورت زیر انجام می‌دهیم.

```
1 tokenizer = BertTokenizer.from_pretrained(model_checkpoint)
2 tokenized_dataset = dataset.map(lambda x: tokenizer(x["text"]), batched=True, num_proc=4, remove_columns=["text"])
3 model = AutoModelForMaskedLM.from_pretrained(model_checkpoint)
4 data_collator = DataCollatorForLanguageModeling(tokenizer=tokenizer, mlm_probability=0.15)
```

```
✓ 5m ▶ 1 args = TrainingArguments(
2     "test-clm",
3     learning_rate=2e-5,
4     weight_decay=0.01,
5     save_total_limit=20,
6     evaluation_strategy = "epoch",)
7 trainer = Trainer(
8     model=model,
9     args= args,
10    train_dataset=tokenized_dataset,
11    eval_dataset=tokenized_dataset,
12    data_collator=data_collator,
13 )
14 # train
15 trainer.train()
```

## Word Embedding

بعد از سیو مدل‌مان نوبت به استفاده از آن می‌رسد که برای اینکار نیاز داریم تا جملات به نحوی تبدیل شوند که parsBert متوجه آنها بشود. از این رو در این تابع متن را به گونه‌ای تبدیل می‌کنیم که قابل فهم bert باشد.

```

1 def Preprocess_For_Bert(text, tokenizer):
2     marked_text = "[CLS] " + text + " [SEP]"
3     tokenized_text = tokenizer.tokenize(marked_text)
4     indexed_tokens = tokenizer.convert_tokens_to_ids(tokenized_text)
5     segments_ids = [1]*len(indexed_tokens)
6     # convert inputs to tensors
7     tokens_tensor = torch.tensor([indexed_tokens])
8     segments_tensor = torch.tensor([segments_ids])
9     return tokenized_text, tokens_tensor, segments_tensor
10

```

مرحله بعد استخراج word embed ها توسط bert است که به صورت زیر انجام میگیرد.

```

11
12 def Etract_Bert_Embeds(tokens_tensor, segments_tensor, model):
13     with torch.no_grad():
14         # obtain hidden states
15         outputs = model(tokens_tensor, segments_tensor)
16         hidden_states = outputs[2]
17         # concatenate the tensors for all layers
18         # use "stack" to create new dimension in tensor
19         token_embeddings = torch.stack(hidden_states, dim=0)
20         # remove dimension 1, the "batches"
21         token_embeddings = torch.squeeze(token_embeddings, dim=1)
22         # swap dimensions 0 and 1 so we can loop over tokens
23         token_embeddings = token_embeddings.permute(1,0,2)
24         # intialized list to store embeddings
25         token_vecs_sum = []
26         # "token_embeddings" is a [Y x 12 x 768] tensor
27         # where Y is the number of tokens in the sentence
28         # loop over tokens in sentence
29         for token in token_embeddings:
30             # "token" is a [12 x 768] tensor
31             # sum the vectors from the last four layers
32             sum_vec = torch.sum(token[-4:], dim=0)
33             token_vecs_sum.append(sum_vec)
34         return token_vecs_sum

```

حالا با استفاده از این توابع امکان تبدیل سوالات و جوابها به تنسور را داریم.

برای نمونه امبد کردن جواب ها به صورت زیر صورت میگیرد.

#### ▼ Embed answers

```

1 answer_embeds = {}
2 for ans in candidate_answers:
3     tokenized_text, tokens_tensor, segments_tensors = Preprocess_For_Bert(ans, tokenizer)
4     list_token_embeddings = Etract_Bert_Embeds(tokens_tensor, segments_tensors, bert_model)
5     answer_embedding = torch.stack(list_token_embeddings, dim=0).sum(dim=0)
6     answer_embeds[ans] = answer_embedding

```

## Data Set

بعد از اتمام این مراحل نوبت به آماده سازی دیتاست برای ورودی دادن به شبکه است. ایده ی ساخت این دیتاست اینطوری است که جدا از جفت سوال جواب هایی که به عنوان سمپل مثبت وجود دارند نیاز داریم تا هر سوال را با مجموعه ای از جواب ها که پاسخ این پرسش نیستند به عنوان نمونه منفی در نظر بگیریم که به دلیل جایگشت بالا تعداد نمونه منفی بالا میرود در این مرحله به جای آنکه همه ی آنها را در نظر بگیریم مانند نگتیو سمپلینگ عمل میکنیم که در اینجا ما در ازای هر نمونه + ۴ عدد نمونه منفی تولید کرده ایم.

```
1 X = []
2 y = []
3 neg_sample_num = 4
4 for idx, row in df.iterrows():
5     tokenized_text, tok_tensor, seg_tensors = Preprocess_For_Bert(row['question'], tokenizer)
6     tok_embeds = Extract_Bert_Embeds(tok_tensor, seg_tensors, bert_model)
7     question_embedding = torch.stack(tok_embeds, dim=0).sum(dim=0)
8     answer_embedding = answer_embeds[row['answer']]
9     x = torch.cat((question_embedding, answer_embedding), 0).numpy()
10    #positive samples
11    X.append(x)
12    y.append(1)
13    #negative samples
14    for i in range(neg_sample_num):
15        r = random.randint(0, len(candidate_answers) - 1)
16        while candidate_answers[r] == row['answer']:
17            r = random.randint(0, len(candidate_answers) - 1)
18        wrong_answer_embed = answer_embeds[candidate_answers[r]]
19        x = torch.cat((question_embedding, wrong_answer_embed), 0).numpy()
20        X.append(x)
21        y.append(0)
```

پیش از تعریف مدل متریک **f1 score** را تعریف کرده ایم تا دید بهتری از آنچه مدل پیش بینی میکند داشته باشیم.

```
1 from keras import backend as K
2
3 def recall_m(y_true, y_pred):
4     true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
5     possible_positives = K.sum(K.round(K.clip(y_true, 0, 1)))
6     recall = true_positives / (possible_positives + K.epsilon())
7     return recall
8
9 def precision_m(y_true, y_pred):
10    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
11    predicted_positives = K.sum(K.round(K.clip(y_pred, 0, 1)))
12    precision = true_positives / (predicted_positives + K.epsilon())
13    return precision
14
15 def f1_m(y_true, y_pred):
16    precision = precision_m(y_true, y_pred)
17    recall = recall_m(y_true, y_pred)
18    return 2*((precision*recall)/(precision+recall+K.epsilon()))
```

## Model

در این پروژه از یک مدل سیکوئانشیال به شکل زیر استفاده کرده ایم:

Model: "sequential"

Layer (type)	Output Shape	Param #
flatten (Flatten)	(None, 1536)	0
dense (Dense)	(None, 1024)	1573888
dropout (Dropout)	(None, 1024)	0
dense_1 (Dense)	(None, 512)	524800
dense_2 (Dense)	(None, 256)	131328
dense_3 (Dense)	(None, 2)	514

=====  
Total params: 2,230,530  
Trainable params: 2,230,530  
Non-trainable params: 0  
=====

در ادامه با استفاده از `svd` حجم ورودی این مدل را کاهش دادیم اما تغییر چشمگیری در وضعیت نسبت به حالت قبل مشاهده نشد از این رو کار را با تعداد اولیه فیچر ها ادامه دادیم.

```
1 from sklearn.decomposition import TruncatedSVD
2 svd = TruncatedSVD(n_components)
3 svd.fit(X)
4 transformed = svd.transform(X)
```

```
[ ] 1 X.shape
(6480, 1536)
```

```
[ ] 1 transformed.shape
(6480, 1000)
```

نتایج برای ایپاک آخر

4ms/step - loss: 0.2283 - accuracy: 0.8767 - f1\_m: 0.3404 - val\_loss: 3.5820 - val\_accuracy:  
0.6782 - val\_f1\_m: 0.3398

## Inference

بخش بعدی نوشتن تابعی است که از ما سوال ورودی بگیرد و با استفاده از مدل جواب احتمالی را باز گرداند. کد مربوطه در زیر آورده شده است در اینجا ابتدا جمله ورودی را امبد میکنیم و پس از آن `predict` مدل را با این جمله اندازه میگیریم از میان احتمال های موجود برای هر جواب یکتا، جواب مربوط به بیشترین احتمال را باز میگردانیم.

```
1 def inference(question):
2     X_test = []
3     tokenized_text, tok_tensor, seg_tensors = Preprocess_For_Bert(question, tokenizer)
4     tok_embeds = Etract_Bert_Embeds(tok_tensor, seg_tensors, bert_model)
5     q_embed = torch.stack(tok_embeds, dim=0).sum(dim=0)
6     for probebel_ans in answer_embeds.items():
7         x = torch.cat((q_embed, probebel_ans[1]), 0).numpy()
8         X_test.append(x)
9
10    X_test = np.array(X_test)
11    #print(X_test)
12    pred=mlp_model.predict(X_test)
13    print(pred)
14    max = pred[0][1]
15    index=0
16    for idx,p in enumerate(pred):
17        if p[1] > max :
18            max = p[1]
19            index=idx
20    return candidate_answers[index]
```

در زیر نیز تست دو جمله را میبینیم که یکی از سوال های موجود در اکسل و دیگری سوال نا موجود و مشابه است که در هر دو حالت جواب درستی بازگردانده شده است.

`inference("چگونه می توان از موفقیت آمیز بودن خرید اینترنتی مطمئن گردید؟")`

پس از انجام خرید سیم کارت از طریق فروشگاه آنلاین همراه اول، می توانید با شماره ای که با آن اقدام به خرید سیم کارت نموده اید وارد حساب کاربری خود شده و در بخش سفارشات من وضعیت خرید را بررسی نمایید.

`inference('چند است؟ 0912 قیمت سیم کارت عادی با پیش شماره')`

قیمت سیم کارت دائمی همراه اول، طرح فیروزه ای درجه یک ۱۴۰۰-۰۰۰ تومان، فیروزه ای درجه دو ۸۰۰-۰۰۰ تومان، فیروزه ای درجه سه ۶۰۰-۰۰۰ تومان، زمردی ۱۶۶-۸۰۰ تومان (با ۴۰٪ تخفیف ۱۰۰-۰۰۰ تومان) و سیم کارت عادی (با پیش شماره ۰۹۱۰) ۱۳۰-۸۰۰ تومان می باشد.