گزارش ساخت مدل تبدیل تاریخ متنی به رسمی

پارسا بختیاری

چکیده

هدف این پروژه ساخت مدل هوش مصنوعی تبدیل کننده تاریخ های غیر رسمی متنی به تاریخ های عددی و رسمی میباشد. در ابتدا دیتاست مورد نیاز با الهام از موارد پیشنهادی ChatGPT شکل گرفت و برای ده سال 1400 تا 1410 ساخته شد. مجموعا 24091 نمونه تاریخ تولید شد. در نهایت مدل BERT در فریم ورک پایتورچ با ۱۵ میلیون پارامتر به مدت حدودا 3 ساعت بروی دیتاست آموزش دیده و به دقت 66٪ رسیده است. همچنین معیار Perpelexity نیز به عدد 1.01 رسیده است که نشان میدهد مدل به خوبی میتواند تبدیل را انجام دهد.

فهرست مطالب

1 دیتاست

این بخش به بررسی نحوه ساخت دیتاست میپردازیم که دو نوع تاریخ رسمی و غیر رسمی را در نظر میگیرد. در نهایت دیتاست تولید شده به فرمت csv. ذخیره میگردد.

تابع convert_year_to_persian(year) سال را به فار سی تبدیل میکند به طوریکه هزار ها و صد ها به شکل فار سی نگا شته میشود. همچنین دو رقم اخر نیز با توجه به دیکشنری اعداد فار سی نیز ترجمه میشود. در حا حاضر این تابع فقط سال های 1400 را در نظر میگیرد اما امکان توسعه آن به سال های قبل و بعد نیز وجود دارد.

تابع generate_date_mappings_with_persian_year(start_year, end_year) تاریخ را برای یک بازه تولید میکند که شامل سه حلقه برای روز ماه و سال میباشد.

همچنین تعداد روز های ماه ها برای نیمه اول و دوم نیز در نظر گرفته میشود به طوری که ماه های نیمه اول سال ۳۱ روز و نیمه دوم به جز اسفند ۳۰ روز میباشد. (اسفند نیز ۲۹ روز)

تاریخ‌ها به دو فرمت:

- غیررسمی: شامل فرمت‌های مختلفی از جمله (روز X ماه سال) و (اول/دوم/... ماه سال)
- رسمی: به فرمت YYYY-MM-DD ذخیره می‌شود.

در نهایت در دو لیست Informal_dates و formal_dates ذخیره میگردد و با استفاده از کتابخانه Pandas تبدیل به یک دیتافریم شده و به فرمت csv. ذخیره میگردد.

نمونه تاریخ تولید شده:

بیست و هشتم اسفند هزار و چهار صد و ده،۱۴۱۰-۱۲-۲۸

روز ۲۸ اسفند هزار و چهار صد و ده،۱۴۱۰-۱۲-۲۸

بیست و نهم اسفند هزار و چهار صد و ده،۱۴۱۰-۱۲-۲۹

روز ۲۹ اسفند هزار و چهار صد و ده،۱۴۱۰-۱۲-۲۹

۲۹ اسفند هزار و چهار صد و ده،۱۴۱۰-۱۲-۲۹

روز ۱ فروردین ۱۴۰۰,۱۴۰۰-۰۱-۰۱

۱ فروردین ۱۴۰۰,۱۴۰۰-۰۱-۰۱

اول فروردین ۱۴۰۰,۱۴۰۰-۰۱-۰۱

روز ۲ فروردین ۱۴۰۰,۱۴۰۰-۰۱-۰۲

دوم فروردین ۱۴۰۰,۱۴۰۰-۰۱-۰۲

۲ فروردین ۱۴۰۰,۱۴۰۰-۰۱-۰۲

۳ فروردین ۱۴۰۰,۱۴۰۰-۰۱-۰۳

سوم فروردین ۱۴۰۰,۱۴۰۰-۰۱-۰۳

روز ۳ فروردین ۱۴۰۰,۱۴۰۰-۰۱-۰۳

۴ فروردین ۱۴۰۰,۱۴۰۰-۰۱-۰۴

2 ساختار مدل

معماری مدل در ابتدا به شـکل Encoder و Decoder طراحی شـد اما به دلیل اهمیت بالای دقت مدل به معماری Encoder only تغییر کرد و از 4 لایه Encoder به همراه یک Fully connected بهره میبرد.

پارامتر های زیر برای معماری در نظر گرفته شده :

" vocab_size" : 25003
" context_length" : 32
" emb_dim" : 256
" n_heads" : 4
" n_layers" : 4
" drop_rate" : 0.1

نکته : در ابتدا برای مدل 3 سر برای روز ماه و سال در نظر گرفته شد که هر کدام وظیفه پیش بینی متغییر نظیر را داشته اند اما در ادامه این سر ها حذف شدند.

تعداد پارامتر های مدل برابر با 14.933.931 میلیون میباشد.

```
Transformer(
  (embedding): Embedding(25003, 256)
  (positional_encoding): Embedding(32, 256)
  (en): TransformerEncoder(
    (layers): ModuleList(
      (0-3): 4 x TransformerEncoderLayer(
        (self_attn): MultiheadAttention(
          (out_proj): NonDynamicallyQuantizableLinear(in_features=256, out_features=256, bias=False)
        )
        (linear1): Linear(in_features=256, out_features=512, bias=False)
        (dropout): Dropout(p=0.1, inplace=False)
        (linear2): Linear(in_features=512, out_features=256, bias=False)
        (norm1): LayerNorm((256,), eps=1e-05, elementwise_affine=True)
        (norm2): LayerNorm((256,), eps=1e-05, elementwise_affine=True)
        (dropout1): Dropout(p=0.1, inplace=False)
        (dropout2): Dropout(p=0.1, inplace=False)
      )
    )
  )
  (fc_train): Linear(in_features=256, out_features=25003, bias=True)
  (fc_year): Linear(in_features=256, out_features=10, bias=True)
  (fc_day): Linear(in_features=256, out_features=31, bias=True)
  (fc_month): Linear(in_features=256, out_features=12, bias=True)
)
```

Figure ۱ معماری شبکه BERT

1.2 توکنایزر

در این مدل از یک توکنایزر فارسی به نام بلبل زبان که به شکل وزن باز در Huggingface موجود است استفاده شده است.

این توکنایزر لغت نامه ای با اندازه 25000 هزار توکن را دارد. طی ارزیابی های انجام شده توکنایزر اعداد را رقم به رقم Encode میکند.

3 آموزش مدل

بعد از طراحی معماری شبکه و جمع آوری دیتاست مورد نیاز نوبت به آموزش مدل میرسد. دیتاست به 3 بخش تقسیم میشود. 80 درصد برای آموزش، 10 درصد برای اعتبار سنجی و 10 درصد برای تست که تعداد نمونه ها به شکل زیر میشود:

Training set length: 19272

Validation set length: 2409

Test set length: 2409

برای دیتاســت نیاز اســت که از روش Masking اســتفاده کنیم. به روش که ما بخش رســمی از تاریخ را ماسک میکنیم تا مدل با پیش بینی آن بتواند فرایند یادگیری را انجام دهد.

```
input_ids = torch.tensor(self.tokenizer.encode(
    self.informal[idx] +" "+"[MASK][MASK][MASK][MASK]-[MASK][MASK]-[MASK][MASK]"))

labels = torch.tensor(self.tokenizer.encode(self.informal[idx] +" "+ self.formal[idx]))
```

تصویر ۱ ماسک کردن تاریخ های رسمی

در نهایت بعد از ساخت دیتا ست و دیتالودر آموزش مدل را آغاز میکنیم. تعداد Batch های آموز شی را 2048 در نظر گرفتیم. و همچنین تعداد Epoch آموزشی را 60 ایپاک در نظر گرفتیم.

برای آموزش از روش Cosine Schaduler استفاده کردیم.

نرخ یادگیری : 0.00005

اپتیمایزر : AdamW

weight_decay = 0.2

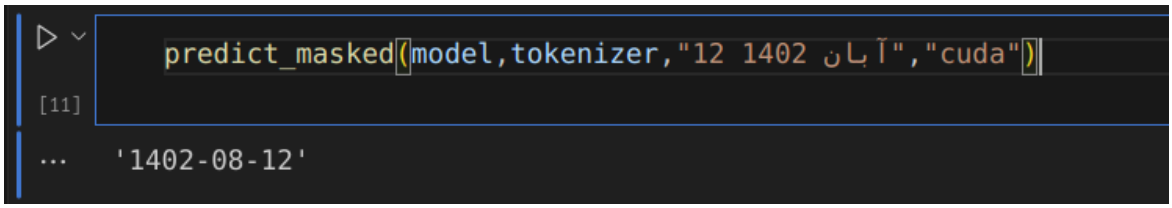
در ابتدا خطای آموزشی برابر با 10.3 است. که در 60 ایپاک به 0.005 رسید. همچنین خطای آموزشی روی دیتای آموزشی نیز از 10.1 به 0.010 رسیده است.

دقت محا سبه شده روی دیتا تست برابر با 66 در صد ا ست. در این دقت مدل باید عینا خروجی نظیر را تولید کند و در صورت وجود یک خطا نیز منفی میگردد.

همچنین میزان Perplexity نیز برابر 1.01 است.

4 استنتاج

در فایل Inference.ipynb مدل به همراه توکنایزر لود می شود و در نهایت ۳ تابع نو شته شده ا ست . یک تابع برای تبدیل Token_ids به متن و یک تابع برای عکس آن اســتفاده میشــود. در نهایت در تابع predict_masked میتوان ورودی را به مدل داد و خروجی مد نظر را تحویل گرفت.



پیوست‌ها

پیوست الف: کتابخانه های مورد نیاز

کتابخانه ها :

PyTorch

Pandas

Numpy

Transformers

پایان