دانشگاه صنعتی امیرکبیر
(پلی‌تکنیک تهران)
دانشکده مهندسی کامپیوتر

پروژه هشتم درس رایانش عصبی

نگارش
سیدمهدی میرفندرسکی
مدرس
دکتر رضا صفابخش
بهمن ۱۴۰۱

فهرست مطالب

۱ اصول اولیه

۱.۱ سوال اول

به صورت کلی مکانیزم توجه مورد استفاده در معماری ترنسفورمر یک مکانیزم کلیدی است که به مدل اجازه می‌دهد تا توالی‌های ورودی با طول‌های مختلف را بدون از دست دادن اطلاعات مهم به طور موثر پردازش کند. مکانیسم توجه، وزن‌هایی را بین عناصر مختلف یک دنباله ورودی و خروجی ارائه شده محاسبه کرده و اختصاص می‌دهد. با این کار تعیین می‌شود که کدام عناصر (از ورودی و خروجی) باید تاثیر بیشتری در تولید خروجی بعدی داشته باشند.

برای پیاده‌سازی این مکانیزم در ترنسفورمرها، ابتدا بازنمایی عددی دنباله ورودی تولید می‌شود (Input embedding). سپس چون مکان‌های هر ورودی در توالی اهمیت دارد طبق مرحله‌ای (positional encoding) مکان هر ورودی در بازنمایی عددی آن تنیده می‌شود. حال تا به اینجا ورودی multi-head attention فراهم آورده شد. سپس سه مقدار کوئری (همان بازنمایی عددی تولید شده مرحله قبل که علاقه‌مند به تعیین معنای آن هستیم)، کلید (بازنمایی عددی از تمام کلمات در دنباله ورودی) و مقدار (بازنمایی عددی معنا برای هر توکن) تعریف می‌شوند. سپس حاصل ضرب داخلی بردار کوئری با بردار کلید محاسبه می‌شود و در نتیجه برای هر توکن در هر موقعیت یک امتیاز (score) بدست محاسبه می‌شود. سپس از این امتیازها برای محاسبه وزن‌های توجه استفاده می‌شود که میزان تأثیر هر عنصر ورودی بر نمایش خروجی را تعیین می‌کند. در نهایت، وزن توجه به بردار مقادیر اعمال می‌شود تا خروجی توجه را تولید کند. سپس این خروجی با توالی ورودی ترکیب می‌شود تا بازنمایی نهایی را برای پیش بینی تولید کند. (البته آنچه تا به حال از این مکانیزم گفته شد خلاصه‌ای بود. مراحلی مانند نرمال‌سازی، اتصالات باقی‌ماندگی و ... در جزئیات وجود دارند. همچنین مکانیزم مشابهی در قسمت کدگشا وجود دارند که کلیات آن به همین منوال است اما چون خروجی‌های آینده وجود ندارد، خروجی امتیازهای مرتبط ماسک می‌شوند.)

۲.۱ سوال دوم

خود-توجه و توجه متقاطع دو نوع مکانیزم توجه هستند که در مدل‌های یادگیری عمیق از جمله معماری ترنسفورمر استفاده می‌شوند.

خود-توجه به مکانیزم توجهی اشاره دارد که در آن توالی ورودی به خود ارجاع داده می‌شود، به این معنی که بردارهای کوئری، کلید و مقدار همه از یک دنباله ورودی هستند. در خود-توجه ، مدل امتیازات توجه را بین هر جفت عنصر در دنباله ورودی محاسبه می‌کند و به طور موثر رابطه بین هر عنصر و هر عنصر دیگر را تعیین می‌کند.

از طرفی، توجه متقاطع به مکانیزم توجهی اشاره دارد که در آن توالی‌های ورودی به خود ارجاع داده نمی‌شوند، بلکه به یکدیگر ارجاع داده می‌شوند. در توجه متقاطع، بردارهای کوئری، کلید و مقدار از توالی‌های ورودی مختلف هستند و امتیازات توجه بین عناصر یک دنباله ورودی با همه عناصر دنباله ورودی دیگر محاسبه می‌شود.

۲ پیش آموزش بدون نظارت

۱.۲ سوال اول

BERT یک تکنیک قبل از آموزش برای ترنسفورمرها است که تأثیر زیادی در زمینه پردازش زبان طبیعی داشته است. ایده اصلی BERT این است که یک شبکه ترنسفورمر دو طرفه عمیق را از قبل بر روی مجموعه بزرگی از داده‌های متنی آموزش دهیم، سپس شبکه از پیش آموزش دیده را برای وظایف خاص‌تر NLP دوباره آماده کنیم.

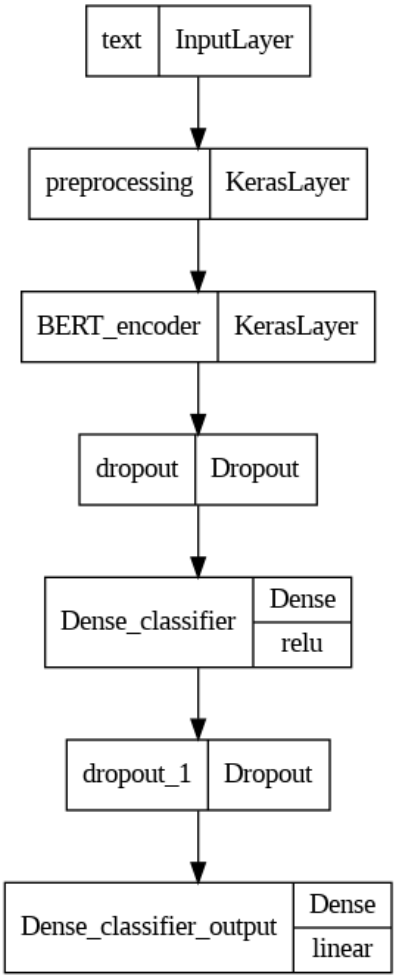
نوآوری کلیدی BERT استفاده از توجه دو طرفه است که به مدل اجازه می‌دهد هم معنای گذشته و هم آینده هر کلمه را در دنباله ورودی در نظر بگیرد. در مدل‌های سنتی یک جهته، مکانیسم توجه فقط می‌تواند معنای قبل از یک کلمه معین را درنظر بگیرد، در حالی که در BERT، مکانیسم توجه می‌تواند هم معنای قبل و هم بعد از یک کلمه را درنظر داشته باشد.

همچنین BERT تکنیکی به کار می‌برد که برخی از کلمات در دنباله ورودی به‌طور تصادفی ماسک شده و مدل باید کلمات ماسک شده را با توجه به معنا کلمات قبلی و بعدی پیش‌بینی کند. این تکنیک به مدل اجازه می‌دهد تا درک عمیقی از روابط بین کلمات در یک جمله و معنای آن‌ها بدست آورد.

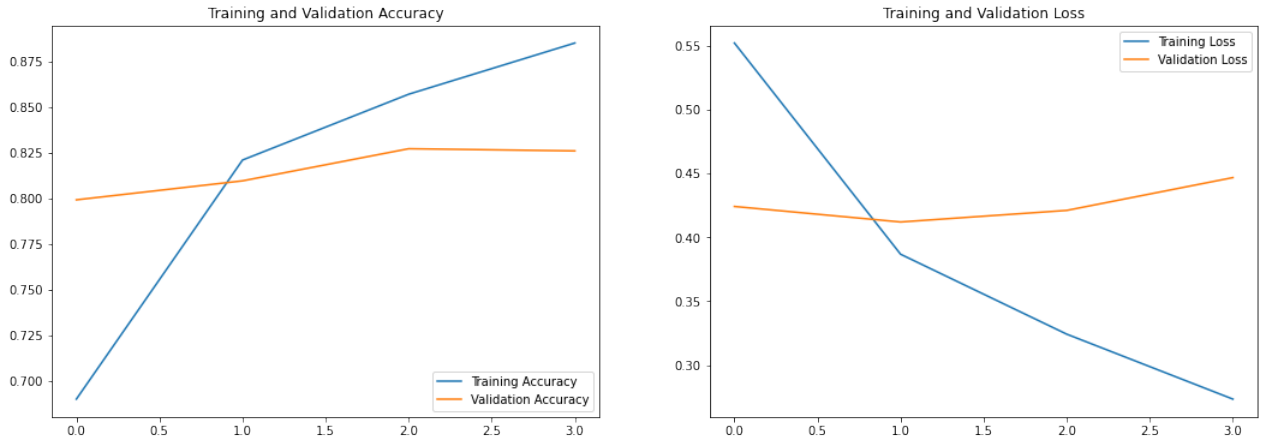
۲.۲ سوال دوم

با توجه به راهنمایی، معماری کل مدل بدین صورت خواهد بود که ابتدا یک لایه ورودی خواهیم داشت. سپس یک لایه یا ماژول پیش‌پردازش نیاز است که تا نظرات قابلیت ورود به شبکه را داشته باشند. بخشی از این لایه پیش‌پردازش شامل مواردی است که در پروژه ششم انجام شد (پیش‌پردازش خود متن و embedding). بعد از آن نوبت به استفاده از یک نوع مدل BERT می‌رسد. سپس بعد از انتخاب یک نوع مدل BERT خواهد رسید. در لینک راهنمایی نام این لایه به گونه‌ای انتخاب شده است که گویی تنها از قسمت کدگذاری ترنسفورمر استفاده می‌شود. این بدان دلیل است که اصلا نیازی به بخش کد گشایی آن نیست (بخش استخراج معنای متن و یک دسته‌بند کافی خواهد بود). همچنین اساسا BERT از بخش مشابه کدگذار ترنسفورمر استفاده می‌کند. در نهایت بعد از قرار دادن مدل BERT یک لایه Drop-out قرار می‌دهد. و در نهایت یک واحد نورون برای تشخیص برچسب قرار گرفته می‌گیرد. مدل خود لینک راهنما با ۵ ای‌پاک تست شد، صحت داده‌های آموزشی نزدیک به ۹۳ دصد و داده‌های آزمون نزدیک به ۸۵ درصد به دست آمد.

اما در این سوال چون نیاز به سعی و خطا برای صحت بهتر نیستیم، مدل دیگری از BERT و همچنین یک لایه کامل Dense به همراه Drop-out استفاده شد. اما تغییری در نحوه ساخت optimizer داده نشد زیرا تغییر و استفاده از موارد گذشته نتایج خوبی حاصل نشد. همچنین برای تقسیم‌بندی داده‌های اعتبارسنجی با توجه به برابر بودن کل آموزش و آزمون، ۰.۱ داده‌های آموزشی برای اعتبار سنجی در نظر گرفته شد. اندازه دسته نیز همان مقدار ۳۲ در نظر گرفته شد. مدل BERT استفاده شده دو لایه با سایز پنهان ۵۱۲ به همراه مقدار ۸ برای attention heads دارد. تعداد ای‌پاک اجرا شده برابر با ۷ است (بدلیل زمان اجرای طولانی هر ای‌پاک). درنهایت نیز از یک es-callback مشابه پروژه‌های قبلی استفاده شد (توقف در ای‌پاک ۴). معماری مدل آموزش داده شده به همراه نمودارهای هزینه و صحت برای داده‌های آموزشی و اعتبار سنجی در ادامه مشاهده می‌شود. همچنین صحت مدل برای داده‌های آموزشی (شامل اعتبارسنجی) ۰.۸۶۳ درصد و برای داده‌های آزمون ۰.۸۱۹ درصد بدست آمد.



شکل ۱: گراف مصور مدل



شکل ۲: نمودار صحت و خطا

همانطور که مشاهده می‌شود es-callback باعث می‌شود که آموزش مدل به محض شروع بیش‌برازش متوقف شود. در حالتی که خطای اعتبارسنجی شروع به افزایش می‌کند، صحت آموزشی (بدون اعتبارسنجی) نزدیک ۸۸ درصد است. درحالی که صحت اعتبارسنجی نزدیک ۸۲ درصد است.

۳ شبکه ترنسفورمر برای مسائل دیگر

۱.۳ سوال اول

از آنجایی که بحث چت‌بات‌ها مدتی داغ است. به عنوان اولین مسئله با استفاده از این ماژول یک برنامه تولید متن نوشته شد. خروجی آن به ازای سه جمله دلخواه به همراه کد در ادامه مشاهده می‌شود.

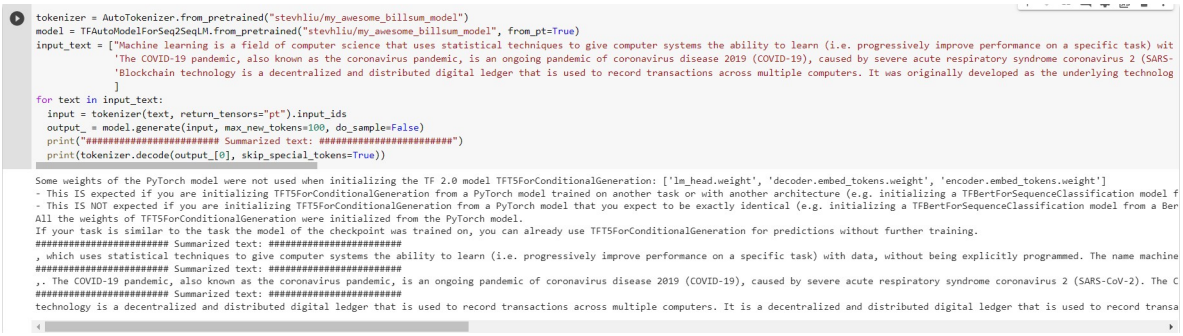
```
tokenizer = GPT2Tokenizer.from_pretrained("gpt2")
model = TFGPT2LMHeadModel.from_pretrained("gpt2")
texts = ["In a shocking turn of events, scientists have discovered that",
         "I am a graduate computer engineering student",
         "Hi, I'm happy"
         ]
# print('########################')
# print(tf.constant(tokenizer.encode(texts, return_tensors="tf"))[None, :])
# print('########################')
# input_ids = tf.constant(tokenizer.encode(texts, return_tensors="tf"))[None, :]
# print(input_ids)
for text in texts:
  input_ids = tokenizer.encode(text, return_tensors="tf")
  print('########################')
  sequence = model.generate(input_ids, pad_token_id=tokenizer.eos_token_id)
  generated_text = tokenizer.decode(sequence[0], skip_special_tokens=True)
  print(generated_text)
```

```
All model checkpoint layers were used when initializing TFGPT2LMHeadModel.

All the layers of TFGPT2LMHeadModel were initialized from the model checkpoint at gpt2.
If your task is similar to the task the model of the checkpoint was trained on, you can already use TFGPT2LMHeadModel for predictions without further training.
########################
In a shocking turn of events, scientists have discovered that the bacteria that cause the disease are actually the
########################
I am a graduate computer engineering student at the University of California, Berkeley. I am currently working on
########################
Hi, I'm happy to announce that I'm going to be joining the team of the new team
```

شکل ۳: مدل تولید ادامه متن

به عنوان مسئله دوم از حوزه پردازش زبان طبیعی، مسئله خلاصه‌سازی متن تست شد. خروجی آن در ادامه مشاهده می‌شود.



شکل ۴: خلاصه‌سازی متن