
بسم الله الرحمن الرحیم

دانشگاه شهید باهنر کرمان

دانشکده فنی و مهندسی

گروه مهندسی کامپیوتر

پروژه کارشناسی

آشنایی با ترنسفورمرها

استاد راهنما: دکتر مهدی افتخاری

توسط: امیرحسن امیرماهانی و غلامرضا دار

تاریخ : ۸/۶/۱۴۰۰

چکیده

چندین سال است که LSTM ها و GRU ها روش های اصلی برای انجام تقریبا تمام کارهای مربوط به پردازش زبان طبیعی و درک زبان[۱] هستند. مانند هر روشی، این روش ها هم مشکلات و ضعف های خود را دارند.

در این پروژه در مورد پیدایش ترنسفورمر[۲]ها، یک روش بر مبنای توجه[۳] به عنوان جایگزین بسیار عالی برای LSTM ها و GRU ها بحث کردیم. ترنسفورمر ها نشان داده اند که یک معماری بسیار گسترش پذیر و قدرتمند هستند. در زمان نوشتن این پروژه هر چند ماه یکبار بهبودی اساسی در ترنسفورمرها رخ می‌داد. بررسی کردن این تغییر ها نیز بخشی از بحث ما بود.

در این پروژه در مورد نسخه‌های مختلف ترنسفورمر ها ازجمله Linformer,Performer, ... صحبت کردیم و اینکه چگونه کارایی ترنسفورمرها را چندین برابر می‌کنند. در مورد ایده های این نسخه های مختلف صحبت کردیم و همچنین نقاط ضعف و قوت هر کدام را نیز بیان کردیم.

در نهایت به طور عملی با کارکردن با ترنسفورمرها آشنا شدیم و طی چندین مثال قابلیت های آنها برای حل مشکلات روزمره با قدرت بسیار زیاد را نشان دادیم. همچنین راجع به آینده شاخه ی پردازش زبان به طور کلی بحث شد و اینکه چگونه ترنسفورمرها می‌توانند همه چیز را تغییر دهند!

---

[۱] Language Understanding

[۲] Transformer

[۳] Attention

فهرست مطالب

فهرست اشکال

فهرست جدول ها

۱- فصل اول: مقدمه و پیشگفتار

۱-۱- پیشگفتار

از زمان اختراع اولین کامپیوتر همیشه یک چالش موجود بود که کامپیوتر ها بتوانند آسان تر با انسان ها ارتباط برقرار کنند. در این زمینه تلاش های زیادی شد و سعی شد که با تعریف نمادهای بصری ارتباط بین کامپیوتر و انسان را تسهیل کنند؛ اما هنوز ارتباط زبانی انسان با ماشین به عنوان یک مسئله و چالش باقی مانده بود. ازین رو برنامه نویسان و طراحان الگوریتم تلاش های زیادی کردند. بعد از ۱۹۵۸ که شبکه های عصبی پا به عرصه وجود گذاشتند. تلاش های بسیاری برای ارتباط زبانی با کامپیوتر از طریق شبکه های عصبی و یادگیری ماشین، صورت گرفت. هر کدام ازین روش ها مزایا و معایبی داشتند که در ادامه به بررسی آن ها می پردازیم.

منابع این گزارش به صورت مجموعه‌ای از فیلم ها جمع آوری شده است تا علاقمندان این حوزه براحتی و با صرف زمان کمتر بتوانند با مسائل روز دنیا آشنا شوند و در حل آن ها تلاش ورزند[۴].

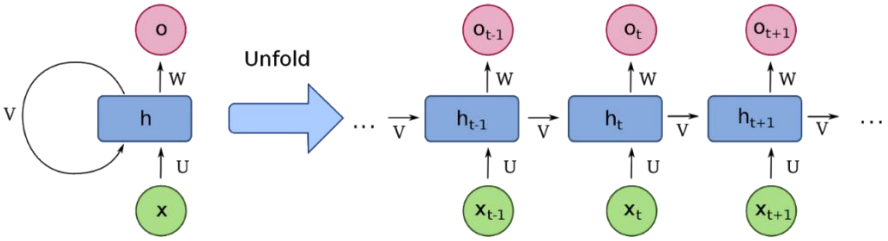
۱-۲-شبکه های RNN

در سال ۱۹۸۶ میلادی یک شبکه عصبی به نام RNN[۵] و برای کاربردهای زیر معرفی شد[1]:

۱. ترجمه زبانی

۲. خلاصه سازی متن

۳. پیش بینی بخش بعدی متن



شکل ۱-۱: ساختار شبکه عصبی RNN

در این تصویر نحوه کارکرد این شبکه، به تصویر کشیده شده است.

[۴] Transformer and language models - YouTube

[۵] Recurrent neural network

این شبکه، از چندین سلول تشکیل میشود؛ که هر سلول دانش اکتسابی اش را به سلول بعدی انتقال میدهد. بدین صورت میتوان تاثیر توالی کلمات و اثر وابستگی کلمات به کلمات ماقبل خودشان را در نتیجه نهایی مشاهده کرد. اما این روش ایرادتی نیز داشت[1] .

۱. نیاز هر سلول به اطلاعات سلول قبل

۲. زمان اجرای زیاد

۳. عدم امکان استفاده از قدرت موازی سازی

۴. مشکل محوشدگی گرادیان

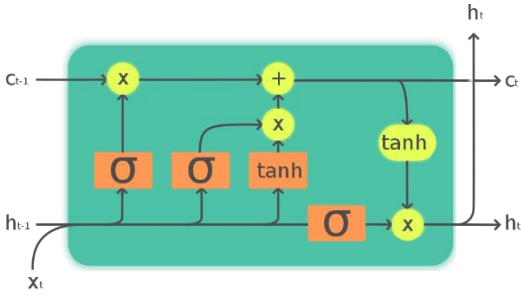
۱-۳-شبکه های LSTM

بعد از یازده سال و در سال ۱۹۹۷ میلادی شبکه های عصبی Lstm [۶]معرفی شدند[2].

این شبکه ها در سال ۲۰۰۷ توانستند انقلابی در زمنیه تشخیص گفتار آغاز کنند.

LSTM ها رکورد های بهبود ترجمه ماشینی مدل سازی زبان و پردازش چند زبانه را شکستند همچنین ترکیبشان با شبکه های CNN[۷] زیرنویس خودکار تصاویر را بهبود بخشید[2]. این شبکه ها از RNN ها سریعتر بودند اما همچنان با مشکل زمان اجرای زیاد دست و پنجه نرم میکردند، ازین رو برای تسریع آن ها از شتاب دهنده های سخت افزاری استفاده میشد.

LSTM ها تا حدی توانستند مشکل محوشدگی گرادیان (به صفر رسیدن تاثیر خطا در لایه های اولی) را نیز حل کنند؛ زیرا واحد های LSTM اجازه می دهند که شیب ها جریان یابند. با این حال این واحد ها هنوز با مشکل انفجار گرادیان و مشکل زمان زیاد آموزش روبرو بودند [2].



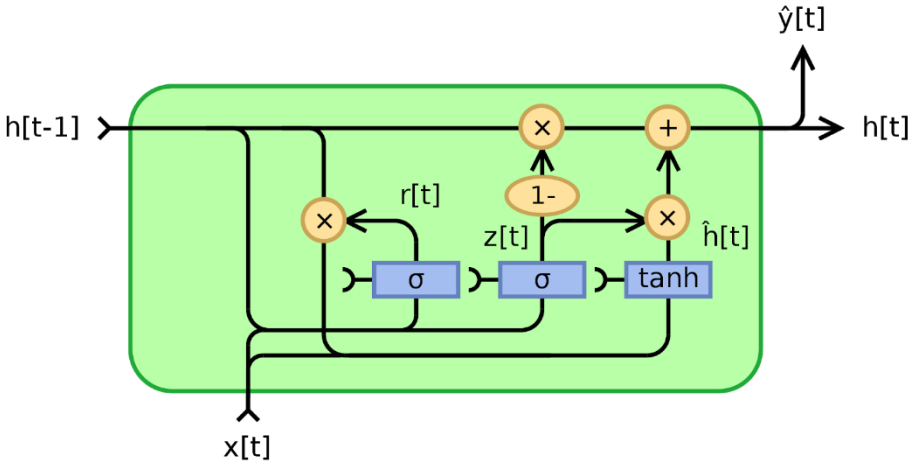
شکل ۱-۲:ساختار یک واحدLSTM

[۶] Long Short-Term Memory

[۷] Convolutional Neural Network

۱-٤-شبکه های GRU

در سال ۲۰۰۴ شبکه های GRU[۸] معرفی شدند[3]. ای شبکه ها بسیار به LSTM شبیه اند با این تفاوت که علاوه بر کاهش تعداد پارامترها، برای جلوگیری از انفجار گرادیان، یک گیت فراموشی به سلول های LSTM افزودند. کارایی این شبکه ها بسیار شبیه شبکه های LSTM اند ولی تحقیقات نشان داده اند که GRU در برخی مجموعه های کوچکتر و کم‌تکرار عملکرد بهتری دارد [3].



شکل ۱-۳:ساختار یک واحد GRU

۱-۵-نگرشی نو

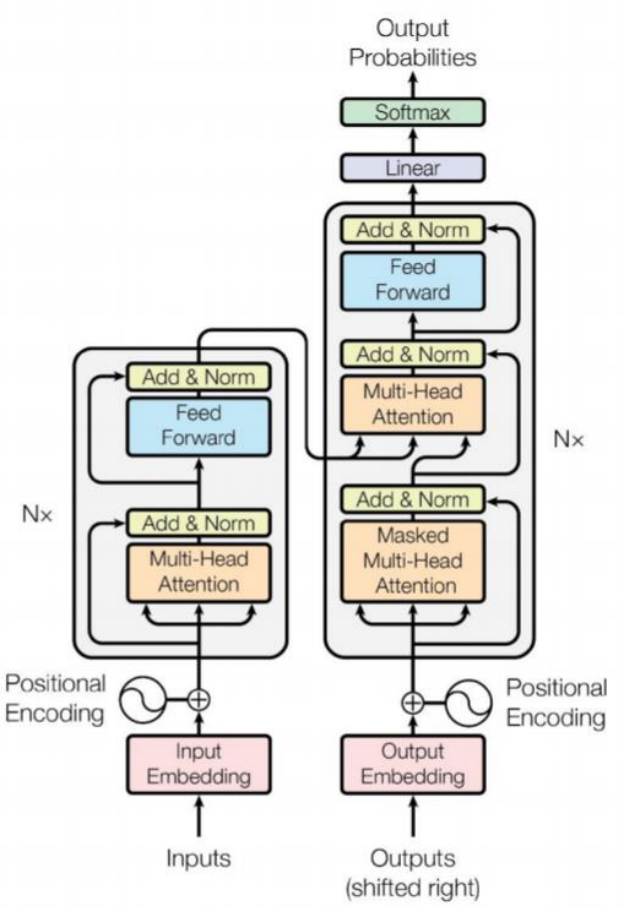
در سال ۲۰۱۷ یک معماری جدید برای شبکه های عصبی معرفی شد که مسئله ارتباط کلمات یک جمله را با دید دیگری بررسی میکرد، دیدی که باعث یک انقلاب جدید در مدل های زبانی شد و توانست انسان را به آرزوی ارتباط زبانی با کامپیوتر و حتی تسهیل ارتباط با انسان های دیگر نزدیک تر کند[4]. در ادامه این گزارش به بررسی ساختار این معماری، کاربردهای آن و مشکلات آن خواهیم پرداخت.

[۸] Gated recurrent unit

۲- فصل دوم: نگاهی عمیق‌تر به معماری ترنسفورمرها

۲-۱-معماری ترنسفورمرها

در فصل قبل با ترنسفورمرها و کاربردهای آنها تا حدی آشنا شدیم. هدف این فصل بررسی کردن دقیق تر معماری و ساختار داخلی ترنسفورمرها است.



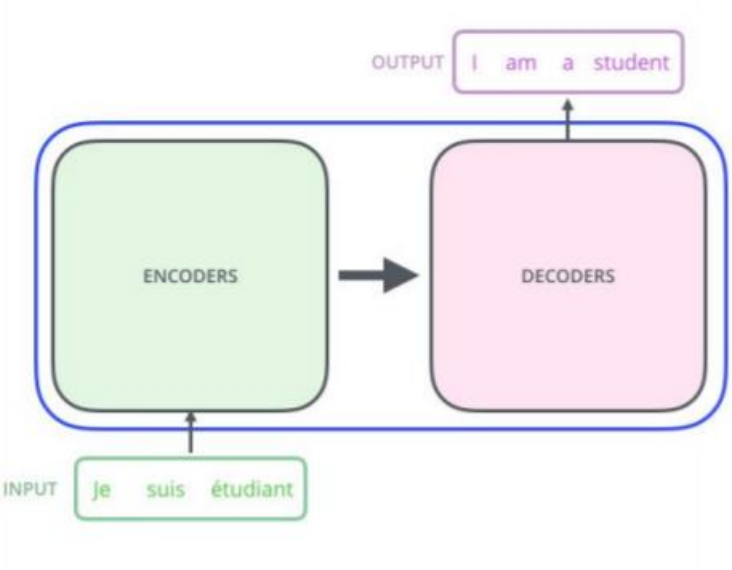
شکل ۲-۱:ساختار معماری ترنسفورمر

همانطور که در شکل ۲-۱ مشاهده می‌شود، ترنسفورمر از تعدادی بلاک و عملیات مختلفی بین این بلاک ها تشکیل شده است[4]. در ادامه این فصل به شرح جزییات مربوط به این بلاک‌ها میپردازیم.

از دید بیرون و به عنوان مثال در کار ترجمه یا Machine Translation ترنسفورمر یک بلاک است که بر اساس ورودی که به آن میدهیم، کلمات متناظر خروجی را یکی پس از دیگری حدس میزند[5].

شکل ۲-۲: شمای کلی ترنسفورمر

اگر کمی بیشتر وارد ترنسفورمر شویم متوجه میشویم که ترنسفورمر از تعدادی Encoder و تعدادی Decoder که به هم متصل اند ساخته شده است. واحد Decoders در واقع استکی از انکدرها است که به صورت سری به هم متصل شده اند، این انکدر ها از نظر ساختار داخلی دقیقا مشابه یکدیگر هستند اما وزن های مستقلی دارند.به طور مشابه دیکدرز هم استکی از دیکدر های مشابه به هم متصل است. اتصال بین Encoders و Decoders به این صورت است که خروجی آخرین یا بالاترین انکدر به تک تک دیکدر ها ارسال شده.



شکل ۲-۳: نگاهی جزئی تر به ساختمان ترنسفورمر

تعداد واحد های انکدر و دیکدر در مقاله Attention is all you need برابر ۶ در نظر گرفته شده است[4]. اما همانطور که میدانید این عدد یک هایپر پارامتر[۹] است و قابل تغییر می‌باشد.

اگر درون یک واحد انکدر را نگاه کنیم، خواهیم دید که شامل یک لایه Self Attention و یک لایه Feed Forward است. با مفهوم Attention در بخش های قبل آشنا شدیم و در این بخش بیشتر با Self Attention آشنا خواهیم شد. سلف

[۹] Hyperparameter

اتنشن این اجازه را می‌دهد که در حالی که یک کلمه ورودی در حال انکد شدن است، بتوانیم به سایر کلمات ورودی نگاه کنیم. خروجی لایه سلف اتنشن به لایه Feed Forward پاس داده خواهد شد.

یک واحد Decoder نیز بسیار شبیه به یک واحد انکدر است با این تفاوت که یک لایه میانی Encoder Decoder Attention داردکه به دیکدر کمک میکند توجه خود را روی قسمت مناسبی از خروجی متمرکز کند.

خوب است کمی بیشتر با نحوه کار انکدر ها آشنا شویم. اولین انکدر کلمات ورودی را به صورت لیستی از بردار ها دریافت میکند، اما ورودی انکدر های بعدی خروجی انکدر قبل از خودشان است که از لحاظ ابعاد با ورودی برابر هستند اما هر کدام معنی درونی خود را دارند.

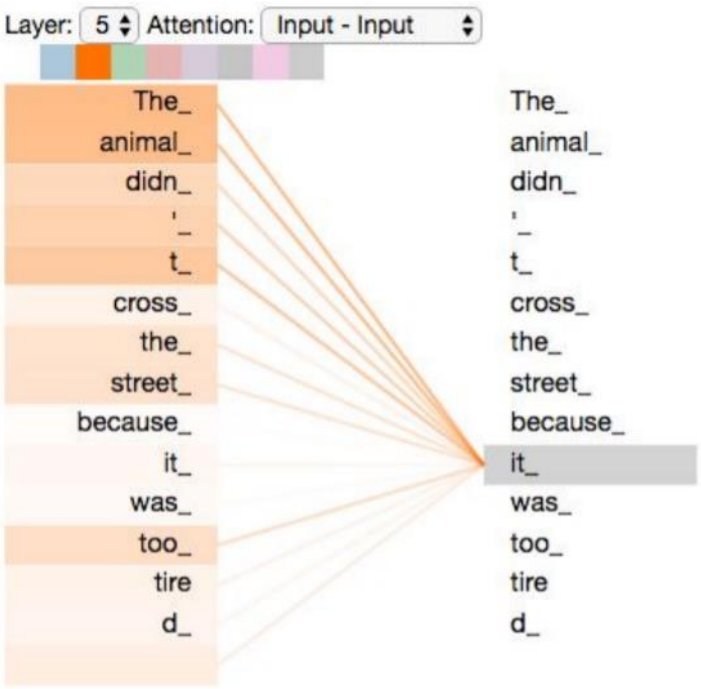
این کلمات ورودی به یک لایه Self Attention میروند و تعدادی ارتباطات بین کلمات بوجود می آید ولی بعد از خروج از این لایه مسیر مستقل خود را به واحد های کوچکتر Feed Forward پیش میگیرند. در این قسمت کلمات مختلف به هم وابسته نیستند که یکی از ویژگی های خیلی مهم ترنسفورمر ها است که قابلیت Parallelize شدن را به آنها میدهد [4].

۲-۲-سلف اتنشن[۱۰]

نوبت آن است به صورت عمیق تر مفهوم سلف اتنشن را بررسی کنیم. فرض کنید جمله The animal didn't cross the street because it was too tired را داریم و میخواهیم بدانیم کلمه "it" به چه اشاره میکند. ما به عنوان انسان خیلی راحت میتوانیم متوجه شویم که منظور animal است و این قسمت از جمله در واقع Animal was too tired بوده است. اما این کار برای ماشین مخصوصا اگر فاصله بین animal و it زیاد باشد اصلا ساده نیست.

حال کاربرد سلف اتنشن این است که زمانی که مدل در حال پردازش یک کلمه است، میتواند به سایر کلمات نگاه بی‌اندازد و با توجه به آنها انکدینگ بهتری از کلمه مورد نظر تولید کند. بنابراین مشابه Hidden State ها در RNN ها ، سلف اتنشن به ترنسفورمر کمک میکند درک بهتری از بقیه کلمات مرتبط درون کلمه فعلی Bake کند[5].

[۱۰] Self Attention



شکل ۲-۴: نحوه ی کار مکانیزم سلف اتنشن

در شکل ۲-۴ میبینیم که ستون سمت چپ و راست هر دو جمله ورودی هستند و در ستون سمت راست کلمه it انتخاب شده است و در ستون سمت چپ میزان توجه به سایر کلمات ورودی دیده می‌شود. همانطور که انتظار میرفت در این مثال بیشترین توجه به کلمه Animal بوده است.

تصویری که در صفحه قبل دیدید توسط یک ابزار به اسم Tensor 2 Tensor تولید شده است که توسط تیم Google Brain توسعه داده شده و یک لایبرری مفید برای انجام تحقیقات در زمینه Machine learning می‌باشد.

۲-۳-نحوه عملکرد سلف اتنشن

نحوه عملکرد سلف اتنشن به این شکل است که علاوه بر فضای توصیفی[۱۱] که از کلمات ورودی تولید شده است، سه بردار دیگر به ازای هر کلمه به اسامی Query,Key,Value در نظر میگیرد[4].

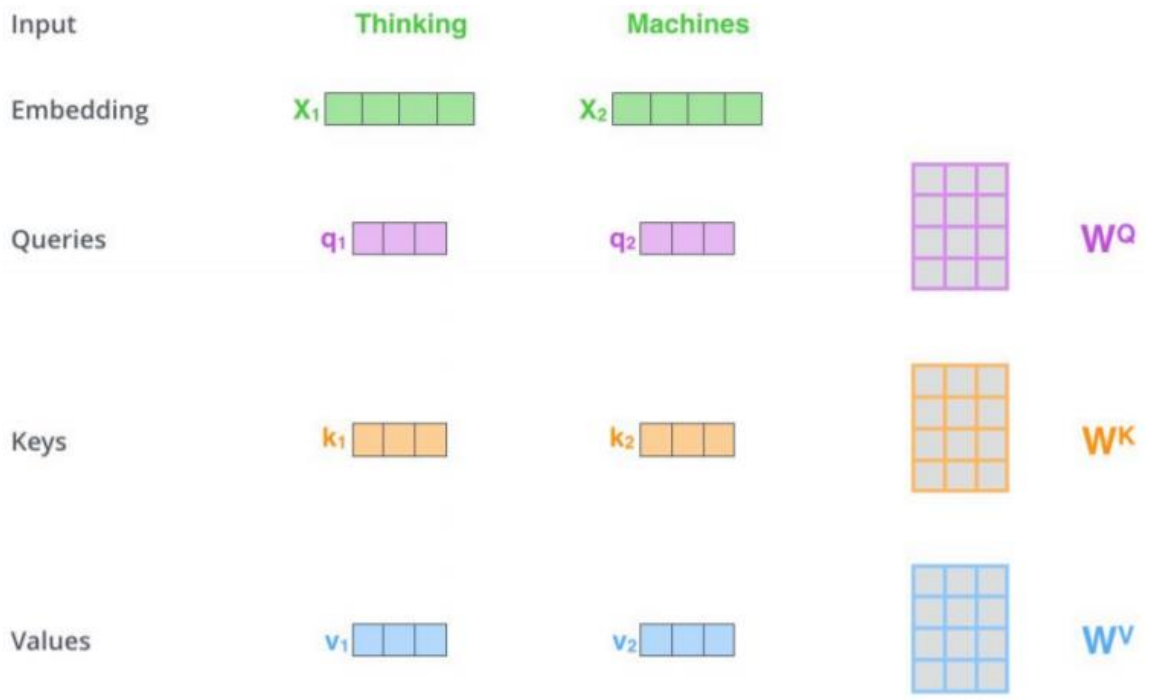
در مقاله Attention is all you need اندازه ورودی ۵۱۲ در نظر گرفته شده است [4] و اندازه این سه بردار ۶۴ است. نکته حائز اهمیتی که در مقاله ذکر شده به آن اشاره شده این است که این سه بردار لزوما نیاز نیست از اندازه ورودی کمتر باشند اما

[۱۱] Embedding

کمتر بودن اندازه آنها کمک میکند در مراحل بعد Multi head self attention به صورت Constant time محاسبه شود. راجع به Multi head attention در ادامه توضیح خواهیم داد.

۲-۳-۱-بدست آوردن بردار های Q,K,V

برای محاسبه این سه بردار، سه ماتریس به نام های Wv, Wk, Wq در نظر میگیریم که در ابتدا با مقادیر تصادفی initialize شده اند. با ضرب کردن این ماتریس ها در بردار ورودی به ترتیب Q,K,V بدست می‌آیند. در مرحله Train شدن شبکه مقادیر این ماتریس ها آپدیت میشود وشبکه سعی میکند رفته رفته مقادیر مناسب تری برای این ماتریس ها پیدا کند. در واقع میتوان گفت این ماتریس ها ماتریس های وزن هستند.



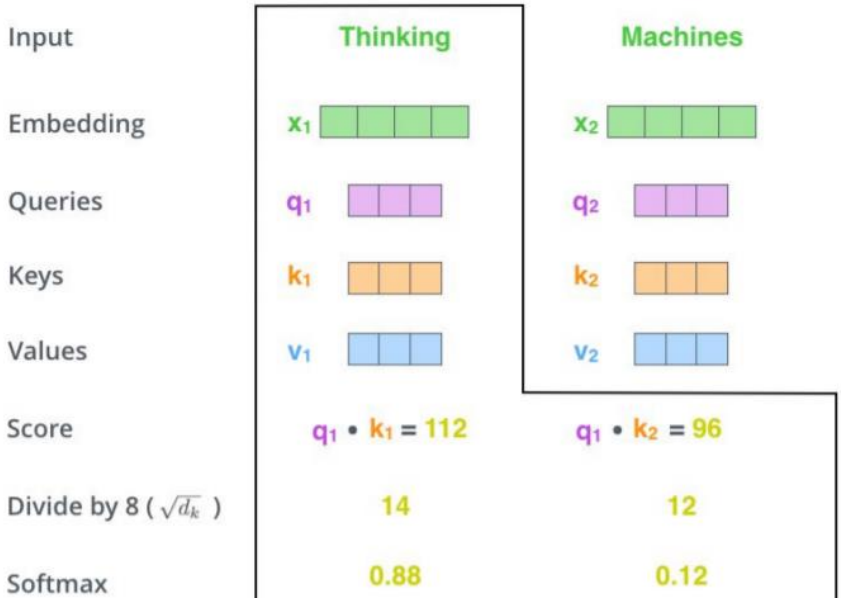
شکل ۲-۵: بدست آوردن بردار های Q,K,V

مرحله بعد برای محاسبه سلف اتنشن بدست آوردن یک Score است.اسکور برای هر کلمه ورودی با توجه به یک کلمه خاص، از ضرب کردن Query آن کلمه خاص در Key هر کلمه بدست می‌آید (در اینجا منظور از ضرب ضرب داخلی است).

اگر به شکل ۲-۵ دقت کنید بهتر متوجه این موضوع می‌شویم. در این مثال Query کلمه it در Key همه کلمات ضرب می‌شود و Score آن کلمه با توجه به it محاسبه می‌شود.

مرحله بعد تقسیم کردن اسکور بر یک عدد است، در مقاله پیشنهاد شده این عدد جزر dimensionality سه بردار کمکی باشد[4] قائدتا این عدد هم میتواند تغییر کند اما نتیجه استفاده از این عدد پایدار شدن گرادیان ها بوده است[5].

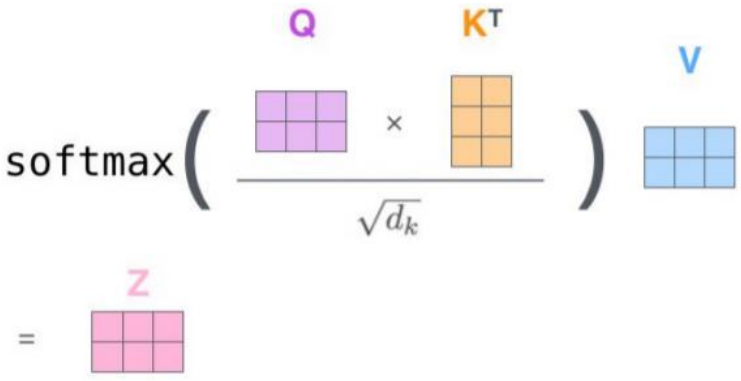
مرحله بعد عبور دادن این مقدار از تابع سافت مکس است که باعث می‌شود اعدادمان در بازه ۰ و ۱ قرار بگیرند و جمع همه آنها برابر ۱ شود که برای مقصود ما بسیار مناسب است.



شکل ۲-۶: مراحل بعدی محاسبه سلف اتنشن

در نهایت برای ساده تر کردن محاسبات ، فرمول محاسبه سلف اتنشن را به صورت ماتریسی در می‌آوریم.

فرمول نهایی در نهایت یک خروجی به نام Z می‌دهد و ما این خروجی را به لایه Feed forward می‌دهیم [5].



شکل ۲-۷: استفاده تابع سافت مکس در محاسبه ی سلف اتنشن

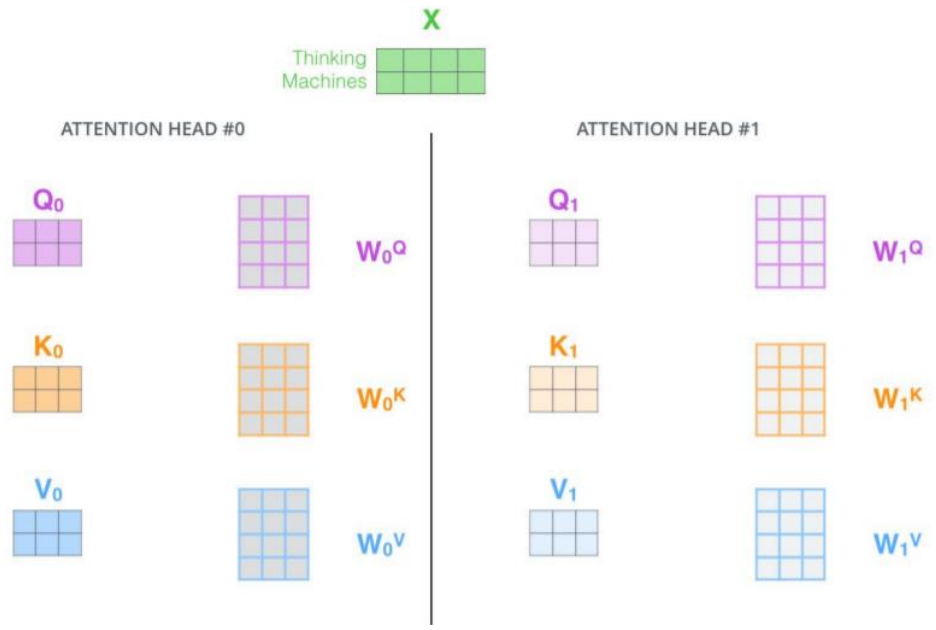
۲-٤-اتنشن چند سره[۱۲]

مولتی هد اتنشن یک مکانیزم برای بهبود سلف اتنشن می‌باشد. از نظر ساختاری مولتی هد اتنشن همان سلف اتنشن است که به تعدادی کپی شده است در نتیجه این عمل بجای داشتن یک ماتریس K,Q,V چندین ماتریس K,Q,V داریم. به هر کدام از این کپی ها یک Attention Head گفته می‌شود و برای ترنسفورمر در این مقاله تعداد ۸ تا انتخاب شده است[4].

مولتی هد اتنشن دو کاربرد اصلی دارد:

۱. قدرت تمرکز مدل روی مکان های مختلف را تقویت میکند. بدون این روش هم مدل روی مکان های مختلف تمرکز میکرد اما معمولا تاثیر آنها بسیار کمتر بود و توسط کلمه اصلی Dominate میشدند[4].
۲. تعداد بیشتری Representation Subspace به مدل می‌دهد و باعث وسیع تر شدن مدل می‌شود[4].

از آنجایی در این روش ۸ هد داریم باید محاسبات مربوط به Score را هم ۸ بار به صورت مستقل از هم و با ماتریس های وزن متفاوت انجام دهیم. به ازای هر هد یک ماتریس Z بدست خواهد آمد پس در نهایت ۸ ماتریس Z خواهیم داشت.

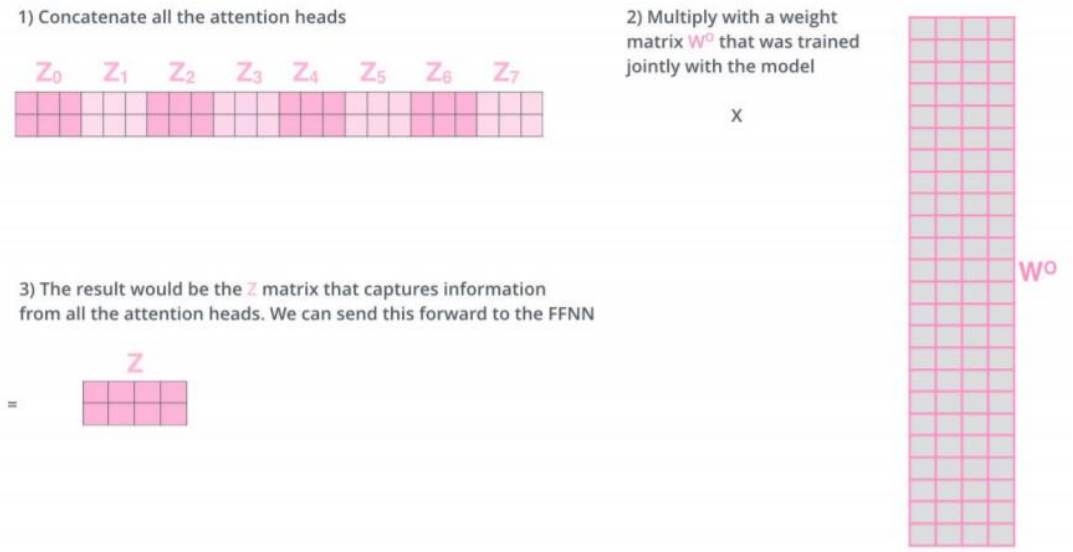


شکل ۲-۸: نحوه محاسبه ی ماتریس ها در اتنشن چند سره

اما همانطور که گفتیم ما برای ارسال به لایه Feed Forward تنها به یک خروجی نیاز داریم پس باید این ۸ خروجی را با هم ترکیب کنیم. برای اینکار همراه با Train شدن شبکه یک ماتریس وزن جدید به نام Wo نیز ترین میکنیم. نحوه ترکیب

[۱۲] Multi Head Attention

کردن این خروجی ها بدین صورت است که ابتدا مانند شکل ۲-۹ ۸ خروجی را به هم Concatenate میکنیم سپس در ماتریس Wo ضرب میکنیم، حاصل Z نهایی ما است که آماده ارسال به لایه Feed Forward می‌باشد[5].



شکل ۲-۹: نحوه ی محاسبه ماتریس Z

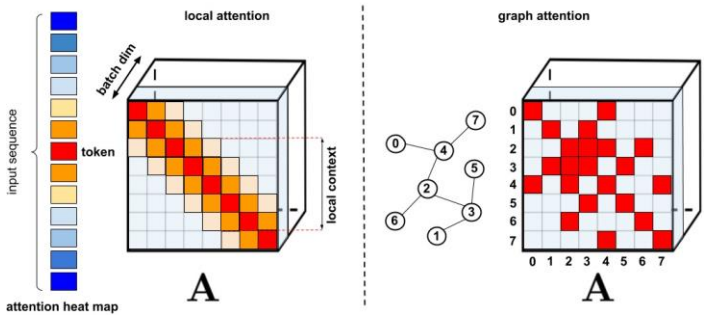
۳- فصل سوم: چالش های معماری ترنسفورمر

یکی از مشکلات اصلی معماری ترنسفورمر مکانیزم self-attention آن هاست[6]. که بدین صورت عمل میکند: مقدار بازنمایی هر توکن با بازدید از همه ی توکن ها بروز رسانی میشود تا بتواند داده ها را برای مدت زیادی ذخیره و نگه داری کند البته این امر باعث میشود تا پیچیدگی زمانی این الگوریتم از مرتبه نمایی و وابسته به اندازه ورودی باشد[6]. نمایی بودن پیچیدگی ترنسفورمر باعث افزایش زمان اجرا، افزایش مصرف برق و نیاز به سخت افزار های بیشتر میگردد. که از نقاط ضعف این الگوریتم است.

تلاش هایی برای بهبود پیچیدگی ترنسفورمرها شد که بطور خلاصه بررسی میکنیم:

۱. مکانیزم توجه اندک[۱۳]

روش کار این مکانیزم بدین صورت است که هر توکن بجای توجه و محاسبات بازنمایی برای همه توکن ها به تعدادی توکن توجه کند. اینکار پیچیدگی نمایی ترنسفورمر ها را به مرتبه $O(n\sqrt{n})$ رساند[6]. البته این روش به صرفه نبود زیرا در ازای کاهش ۲ درصدی کارایی تنها ۲۰ درصد افزایش سرعت داشت و نمیتوانست بخوبی مسئله را حل کند.



شکل ۳-۱: روش های توجه اندک از اساس کار شبکه های عصبی گرافی نیز ایده گرفتند که با روابط خاص بصورت گراف توجه ظاهر می‌شوند.

متأسفانه ، روشهای توجه اندک هنوز هم محدودیتهای زیادی دارند.

۱. به عملیات ضرب ماتریس پراکنده کارآمد نیاز دارند که در همه شتاب دهنده ها موجود نیست.
۲. معمولاً تضمین های نظری دقیق برای قدرت تمثال خود ارائه نمیدهند.
۳. فقط برای مدل های ترنسفورمر و pre-training بهینه شدند.
٤. معمولاً لایه های توجه بیشتری را برای جبران نمایش های پراکنده روی هم میگذارند ، که استفاده از آن ها را با سایر مدلهای pre-trained دشوار می کند، زیرا که نیاز به آموزش مجدد و مصرف قابل توجه انرژی دارند.

[۱۳] sparsity attention mechanism

اگر از همه محدودیت هایی گفتیم بگذریم باز دو مشکل اساسی این مکانیزم این است که گاهی اوقات به اندازه ای که نیاز است از ورودی انتخاب نمیکند و مشکل بوجود میاورد همچنین نمیتواند از سربار یکسری عملگر هایی که خیلی سربار دارن کم کند نمونش softmax که برای نرمال سازی یا سامانه های پیشنهادگر زیاد استفاده میشود

۲. Reformer

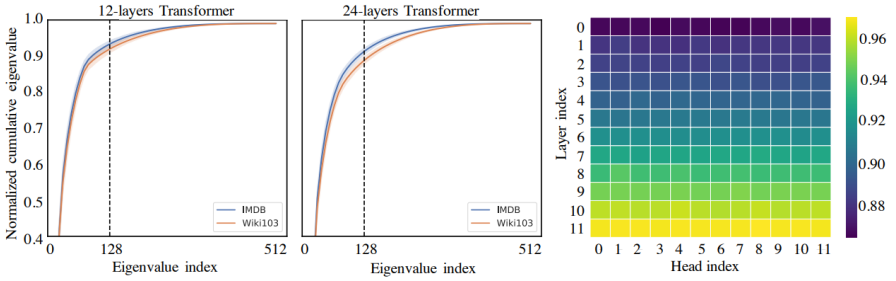
از دیگر تلاش های کاهش پیچیدگی میتوان مقاله reformer را نام برد[7]. این مقاله از روش Locally sensitive hashing برای کاهش پیچیدگی استفاده میشود و پیچیدگی را به مرتبه $O(n \log n)$ کاهش میدهد[7]. این روش نیز خالی از ایراد نبود؛ از ایرادات این روش این بود که تاثیر چندانی برای رشته های کوتاه نداشت و تاثیر آن فقط در رشته هایی با طول بلند تر از ۲۰۴۸ چشم گیر و قابل توجه بود. دیگر مشکل این روش این بود که با استفاده از روش های رمزنگاری پیچیده کارایی را کاهش میداد.

این مشکلات باعث شد تا محققان به دنبال راه حل دیگری باشند.

۳-۱- ترنسفورمرهای خطی[۱۴]

ترنسفورمر خطی توسط تیم فیسبوک در سال ۲۰۲۰ در قالب یک مقاله معرفی شد[6].

ایده اصلی این مقاله این بود که ماتریس توجه در ترنسفورمرها میتواند به کمک یک ماتریس درجه کمتر تخمین زده شود. در این مقاله از روش Singular value decomposition [8] برای ماتریس های Key , Value کمک گرفته شد تا بتوانند درجه ماتریس ها را کاهش و در نهایت پیچیدگی محاسبات را کاهش دهند. این مهم باعث شد تا ترنسفورمر های خطی هم در مرحله آموزش و هم در مرحله استنتاج پیچیدگی خطی داشته باشد. نام گذاری اینگونه ترنسفورمر ها نیز به دلیل مرتبه پیچیدگی آن ها بود (Linear + Transformer) Linformer .



شکل ۲-۳: نمودار های بالا نشان میدهند که می توان نتیجه محاسبات ترنسفورمر ها را با ماتریس های درجه کمتر تخمین زد. باید توجه داشت که اگر داده ها در ابعاد مختلف ماتریس بطور یکنواخت توزیع شده باشند یک نمودار خطی قابل انتظار بود. توی این شکل مشخص است که اکثر تاثیر داده ها در ۱۲۸ بعد نخست وجود دارد و در ۳۸۴ بعد، بعدی تاثیر کمی وجود دارد.

این نمودار نتیجه آزمایش برروی ترنسفورمر های ۱۲ و ۲۴ لایه و روی مجموعه دادگان IMDB و WIKI103 بوده است.

[۱۴] Linformer

درجه ماتریس:

در جبر خطی درجه یک ماتریس تعداد بعد فضای برداری است که توسط ستون های آن ماتریس پوشش داده میشود به بیان ساده تر درجه ماتریس تعداد سطرها یا ستون های مستقل خطی یک ماتریس است[9].

$$\begin{bmatrix} 1 & 0 & 1 \\ -2 & -3 & 1 \\ 3 & 3 & 0 \end{bmatrix} \quad \begin{bmatrix} 1 & -1 \\ 1 & -1 \\ 0 & 0 \\ 2 & -2 \end{bmatrix}$$

به عنوان مثال در ماتریس سمت راست ستون وسط را میتوان از تفریق ستون های چپ و راست بدست آورد. پس تنها ۲ ستون مستقل خطی دارد و درجه آن ۲ است.

یا در ماتریس سمت چپ هر دو ستون از روی یکدیگر با قرینه کردن درایه ها میتوان بدست آورد. پس درجه آن ۱ است.

۳-۱-۱-نتایج بنچمارک ها

جدول ۳-۱: بنچمارک‌های ترنسفورمر خطی

n	Model	SST-2	IMDB	QNLI	QQP	Average
512	Liu et al. (2019), RoBERTa-base	93.1	94.1	90.9	**90.9**	92.25
	Linformer, 128	92.4	94.0	90.4	90.2	91.75
	Linformer, 128, shared kv	**93.4**	93.4	90.3	90.3	91.85
	Linformer, 128, shared kv, layer	93.2	93.8	90.1	90.2	91.83
	Linformer, 256	93.2	94.0	90.6	90.5	92.08
	Linformer, 256, shared kv	93.3	93.6	90.6	90.6	92.03
	Linformer, 256, shared kv, layer	93.1	94.1	**91.2**	90.8	**92.30**
512	Devlin et al. (2019), BERT-base	92.7	93.5	91.8	89.6	91.90
	Sanh et al. (2019), Distilled BERT	91.3	92.8	89.2	88.5	90.45
1024	Linformer, 256	93.0	93.8	90.4	90.4	91.90
	Linformer, 256, shared kv	93.0	93.6	90.3	90.4	91.83
	Linformer, 256, shared kv, layer	93.2	**94.2**	90.8	90.5	92.18

نتیجه بنچمارک هایی که ترنسفورمر خطی بر روی آن ها اجرا شده است این مهم را نشان میدهد که ترنسفورمر خطی در همه مواقع سریعتر از ترنسفورمر معمولی عمل کرده است[6]؛ ولی معمولا نتایج بدتری به همراه داشته است. هرچند که در مواردی خاص توانسته از ترنسفورمر معمولی هم نتیجه بهتری داشته باشد. در این بنچمارک ها نشان داده شد که کارایی ترنسفورمر خطی به ابعاد وابسته نیست و به یک مقدار ثابت وابسته است که اثباتی بر خطی بودن آن است[6].

۳-۱-۲-ترنسفورمر خطی در مرحله استنتاج

ترنسفورمر خطی علاوه بر خطی عمل کردن در مرحله آموزش، در مرحله استنباط هم خطی عمل میکند و بهبود های زیادی را در زمینه استنتاج یا استفاده کردن از مدل آموزش داده شده میتوانیم شاهد باشیم .

جدول ۳-۲: ین جدول نشان دهنده یک عمل استنباط با استفاده از یک GPU تسلا وی ۱۰۰ با ۱۶ گیگ وی رم است. جدول سمت چپ نشان دهنده زمان صرفه جویی شده نسبت به استفاده از ترنسفورمر معمولی و جدول سمت راست نشون دهنده حافظه صرفه جویی شده نسبت به استفاده از ترنسفورمر معمولی است.

length n	projected dimensions k				
	128	256	512	1024	2048
512	1.5x	1.3x	-	-	-
1024	1.7x	1.6x	1.3x	-	-
2048	2.6x	2.4x	2.1x	1.3x	-
4096	3.4x	3.2x	2.8x	2.2x	1.3x
8192	5.5x	5.0x	4.4x	3.5x	2.1x
16384	8.6x	7.8x	7.0x	5.6x	3.3x
32768	13x	12x	11x	8.8x	5.0x
65536	20x	18x	16x	14x	7.9x

length n	projected dimensions k				
	128	256	512	1024	2048
512	1.7x	1.5x	-	-	-
1024	3.0x	2.9x	1.8x	-	-
2048	6.1x	5.6x	3.6x	2.0x	-
4096	14x	13x	8.3x	4.3x	2.3x
8192	28x	26x	17x	8.5x	4.5x
16384	56x	48x	32x	16x	8x
32768	56x	48x	36x	18x	16x
65536	60x	52x	40x	20x	18x

۳-۱-۳-کاربردهای ترنسفورمر خطی

از کاربرد ها و تاثیر های ترنسفورمر خطی در بلند مدت میتوانیم موارد زیر را نام ببریم[6]:

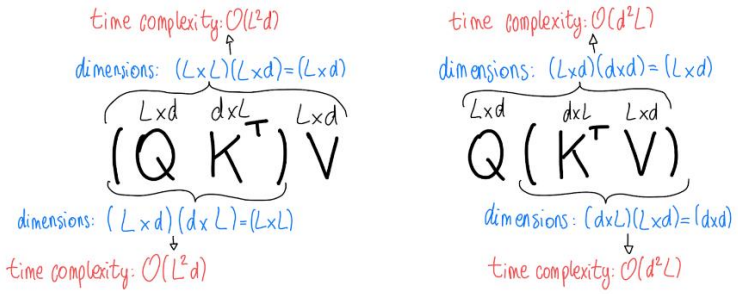
۱. کار کردن روی کارهای مربوط به تصویر
۲. انجام کارهایی که صرفا بخاطر هزینه بر بودن قابل انجام نبودن
۳. ذخیره انرژی و کمک به محیط زیست (به دلیل بهینگی و مصرف کمتر انرژی برق)

از معایب احتمالی این ساختار میتوان به مشکلات اخلاقی و استفاده نادرست از این قدرت اشاره کرد. مثل استفاده از این شبکه در شکستن رمز ها و ...[6]

۳-۲-پرفورمر[۱۵]

پرفورمر در تاریخ ۳۰ سپتامبر ۲۰۲۰ و توسط تیم تحقیقاتی گوگل به عنوان یک مقاله کنفرانسی ارائه شد[10].

شیوه کار پرفورمر[11]:



شکل ۳-۳: بررسی اساس کار ترانسفومر

اساس کار ترنسفورمر سه ماتریس key,query,value است که طبق شکل سمت چپ ضرب میشوند و با توجه به ابعادشون و بقیه مراحل سربار سنگینی بزای شبکه ایجاد میکنند محققین گوگل با برسی ابعاد این ماتریس ها به این نتیجه رسیدند که چون

[۱۵] performer

d از L کوچکتر است؛ اگر مثل تصویر سمت راست بتوانیم ضرب را انجام دهیم خیلی سربار کاهش پیدا میکند ولی در مرحله محاسبه softmax بخاطر ذات نمایی این الگوریتم تاثیر سربار چندین برابر میشد. پس رهایی از الگوریتم softmax و یک تخمین خوب برای مقدارش میتوانست مشکل را حل کند[10].

بعد از محاسبات ریاضی بسیار زیاد محققین توانستن با ضرب دوتا مقدار K' و Q' نتیجه تابع softmax را تقریب بزنند. اینکار سر بار نمایی الگوریتم به ازای هر درایه ماتریس را حذف میکرد.

(۱-۳)

$$\text{softmax}\left(\frac{\mathbf{QK}^\top}{\sqrt{d}}\right) = \mathbf{D}^{-1}\mathbf{A}$$

$$\text{with:} \quad \mathbf{A} = \exp\left(\mathbf{QK}^\top/\sqrt{d}\right), \quad \mathbf{D} = \text{diag}(\mathbf{A}\mathbf{1}_L)$$

(۲-۳)

$$\mathbf{A}(i,j) = \mathrm{K}(\mathbf{q}_i, \mathbf{k}_j) = \exp(\mathbf{q}_i\mathbf{k}_j^\top) = \phi(\mathbf{q}_i)^\top\phi(\mathbf{k}_j)$$

اگر a ماتریس نتیجه تابع softmax باشد، برای تخمین زدن آن میتوان از کرنل استفاده کرد که این کرنل مقداری متغییر است و میتواند از نوع گوسی یا مثلثاتی یا... باشد.

(۳-۳)

$$\phi(\mathbf{x}) = \frac{h(\mathbf{x})}{\sqrt{m}}\left(f_1(w_1^\top\mathbf{x}), ..., f_1(w_m^\top\mathbf{x}), ..., f_l(w_1^\top\mathbf{x}), ..., f_l(w_m^\top\mathbf{x})\right)$$

(۴-۳)

$$h(\mathbf{x}) = 1, l = 2, f_1 = \sin, f_2 = \cos, \mathcal{D} = \mathcal{N}(0, \mathbf{I}_d)$$

برای محاسبه ϕ ها این رابطه موجود است که اثبات آن در مقاله بطور کامل آورده شده است. و اگر در مورد ویژگی های فوریه تصادفی اطلاعاتی داشته باشید درک آن برایتان ساده تر خواهد بود. کاربرد ϕ ها این است که مسئله را از بُعد خطی به یک بُعد دیگر ببرد. همچنین می دانیم که اگر ϕ_q را در ϕ_k ضرب کنیم یک تقریب خوب از تابع softmax بدست خواهد آمد[11].

در رابطه محاسبه ϕ چندین امگا داریم که بردارهای تصادفی از یک توزیع احتمال مشخص مثل d اند که مثلا دراین رابطه از صفر تا i است که میتوان گفت یک توزیع نرمال است.

h یک تابع قطعی از ورودی است که با تقسیم بر $\sqrt{m}$ نرمال میشود. اینجا h را فعلا یک در نظر گرفته شده است و D نرمال سازی خطی بوده است. انتخاب h و f یا توابع نوع کرنل را مشخص میکنند که مثلا اینجا گوسی است[11].

$$h(\mathbf{x}) = \exp\left(\frac{||\mathbf{x}||^2}{2}\right) \tag{۵-۳}$$

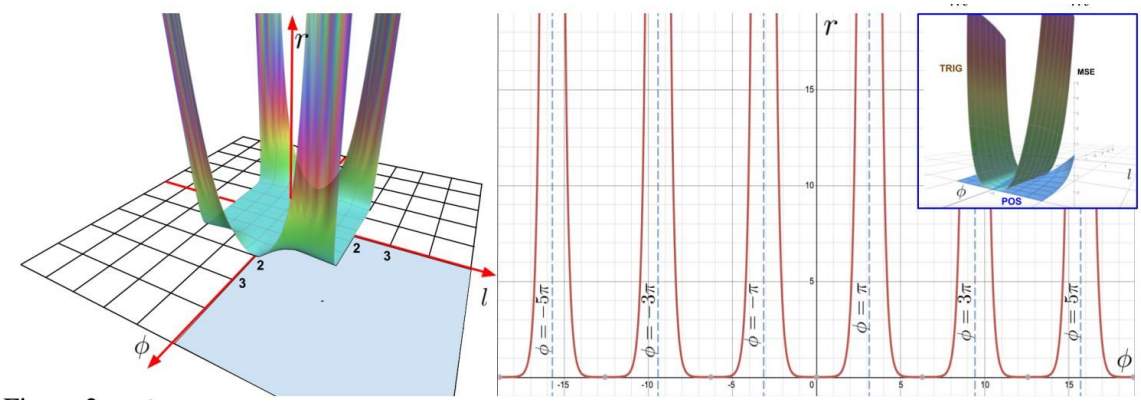
با کمی تغییر تابع h میتوانیم یک کرنل برای softmax بسازیم با مشکلاتی روبرو خواهیم شد، مهمترین مشکل آن این است که تابع softmax خروجی همیشه مثبت دارد در حالی که این تخمین ممکن است خروجی منفی نیز داشته باشد.

برای رفع این مشکل یک رابطه جدید برای کرنل softmax پیشنهاد شد اگر بخواهیم این کرنل را با همان رابطه ϕ که داشتیم بدست بیاوریم ثابت ها و توابع مجهول به صورت زیر در خواهند آمد[11].

$$\begin{aligned} \mathrm{K_{SM}}(\mathbf{x},\mathbf{y}) &= \exp(\mathbf{x}^\top \mathbf{y}) = \mathbb{E}_{w\sim\mathcal{N}(0,\mathbf{I}_d)}[\exp(w^\top \mathbf{x} - \frac{\|\mathbf{x}\|^2}{2})\exp(w^\top \|\mathbf{y}\| - \frac{\mathbf{y}^2}{2})] \\ &= \mathbb{E}_{w\sim\mathcal{N}(0,\mathbf{I}_d)}[\exp(-\frac{\|\mathbf{x}\|^2}{2})\exp(-\frac{\|\mathbf{y}\|^2}{2})\exp(w^\top \mathbf{x})\exp(w^\top \mathbf{y})] \end{aligned} \tag{۶-۳}$$

$$h(\mathbf{x}) = \exp(-\frac{||\mathbf{x}||^2}{2}), l = 1, f_1 = \exp, \mathcal{D} = \mathcal{N}(0, \mathbf{I}_d) \tag{۷-۳}$$

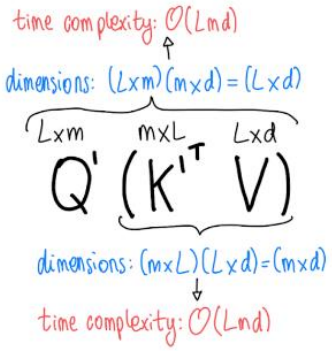
بعد از انجام محاسبات یک نمودار از نتیجه ترسیم شد که یک مشکل اساسی را نشان میداد که اگر با همون کرنل قبلی یا اصطلاحا کرنل مثلثاتی کار کنیم خطا در اطراف صفر به سمت بی نهایت میرود، در صورتی که برای کرنل جدید خطا به سمت صفر میرفت.



شکل ۳-۴: خطای مشاهده شده در کرنل مثلثاتی

در مقاله کرنل های دیگری نیز از تابع softmax اثبات شده اند[10].

بعد از انجام این محاسبات پیچیده سربار محاسباتی از شبکه حذف میشود و بجای تابع softmax خواهیم داشت:

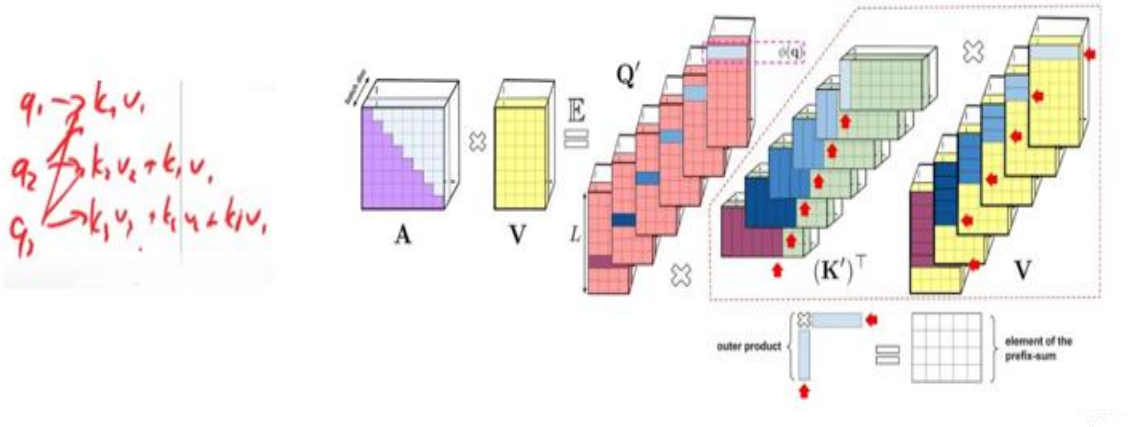


شکل ۳-۵: مرتبه پیچیدگی پرفورمر

که این تابع از مرتبه پیچیدگی $O(Lmd)$ است و براحتی میتوان گفت که پیچیدگی آن خطی است[11].

۳-۲-۱-توجه بر اساس پیشوندها

خب ممکن است که بخواهیم این روش را در مدل های عمومی تری مانند GPT پیاده کنیم؛ در این مدل ها چون ماتریس A خلوت[۱۶] است ما از روش جمع پیشوندی استفاده میکنیم که رابطه آن در شکل زیر آورده شده است .

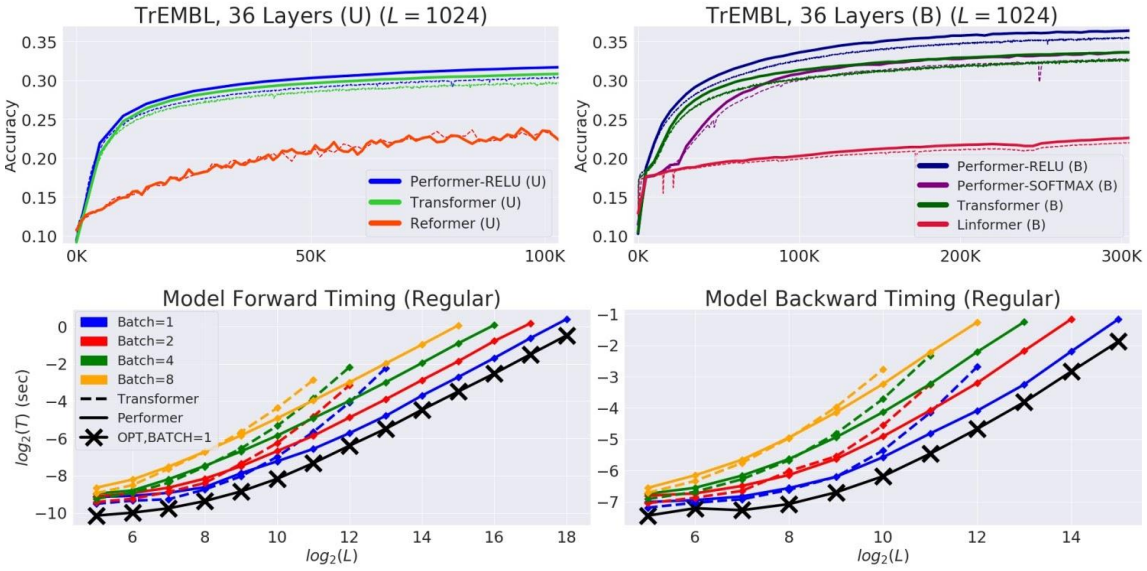


شکل ۳-۶: توجه براساس پیشوندها

برتری پرفورمر به نسبت ترنسفورمر خطی در این مورد این است که پرفورمر جداسازی ماتریس ها از یکدیگر را بهتر انجام داده است[10].

[۱۶] Sparse

۳-۲-۲-مقایسه ها

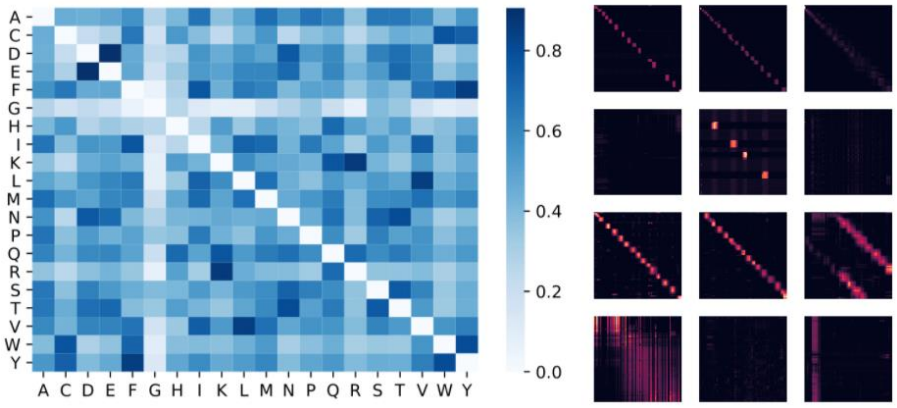


شکل ۳-۷: نمودار های دقت و زمان اجرای الگوریتم های مختلف

نمودار های بالا دقت و زمان اجرای الگوریتم های مختلف را باهم مقایسه کرده اند که در مجموع میتوان گفت که پرفورمر عملکرد بسیار خوبی از خودش نشان داده و زمان اجرای کمتری دارد .

۳-۲-۳-حوزه های کاربرد پرفورمر

۱. بیولوژی و داروسازی
۲. حفظ محیط زیست (زمان اجرای کمتر و مصرف کمتر انرژی)
۳. تحقیقات بر روی شبکه های ترنسفورمر
٤. سازگاری با روش ها و مدل های موجود



شکل ۳-۸: نتیجه استفاده پرفورمر در مبحث ژنتیک

یکی از زمینه های کاربرد پرفورمر که در مقاله بررسی شده است، مبحث بیولوژی و ژنتیک می باشد[10] که همیشه چالش طول بلند ژنوم ها در این حوزه خیلی از شبکه ها را رفوزه میکرد. پر فورمر با استفاده از تخمین و کرنل هایش توانسته به این چالش غلبه کند و به نتایج خوبی در این زمینه برسد .

٤- فصل چهارم: آشنایی عملی با ترنسفورمرها (و محصولات Hugging face)

٤-١-کمپانی هاگینگ فیس[۱۷]

در این بخش می‌خواهیم با پیاده سازی و استفاده عملی از ترنسفورمر ها بیشتر آشنا شویم. ابتدا خوب است کمی با کمپانی هاگینگ فیس و خدماتشان آشنا شویم. در بخش بعدی به صورت عملی یک مدل ترنسفورمر ترین شده را برای یک تسک خاص(کامنت های IMDB) فاین تون[۱۸](تنظیم) میکنیم.

شرکت هاگینگ فیس یکی از شرکت های اصلی فعال در زمینه NLP و بخصوص ترنسفورمر هاست. یکی از خدمات اصلی و مهم این شرکت در دسترس قرار دادن انواع مدل های ترنسفورمر از قبل ترین شده که توسط شرکت های بزرگی مانند google, facebook و ... منتشر شده اند است.

همچنین این شرکت سرویس های Hosted inference را هم برای مدل ها ارائه میدهد تا نیاز نداشته باشید خودتان به فکر دیپلوی کردن و پیکربندی محیط برای استفاده از نتیجه مدلتان باشید. این سرویس یک نسخه رایگان دارد که تا ۳۰۰۰۰ کاراکتر ورودی در ماه را پوشش میدهد و نسخه های غیر رایگان با امکانات بیشتر هم موجود است.

یکی دیگر از سرویس هایی که این شرکت ارائه میدهد سرویس نوپای Auto NLP است. هدف این سرویس این است که در آینده ای نه چندان دور به راحتی وارد کردن چند دستور بتوانیم مدل های NLP خود را به آسانی ترین و دیپلوی کنیم.

٤-٢-کتابخانه Transformers

اصلی ترین محصول این شرکت کتابخانه Transformer برای پایتون است. این لایبرری که از PyTorch و TensorFlow که دو فریم ورک اصلی ماشین لرنینگ در پایتون هستند پشتیبانی میکند، امکانات بسیار زیادی برای کار با ترنسفورمر ها در اختیار علاقه مندان قرار میدهد.

---

[۱۷] HuggingFace

[۱۸] Fine-Tune

بخشی از مدل های موجود در این لایبرری را در شکل ۴-۱ مشاهده میکنید.

Model	Tokenizer slow	Tokenizer fast	PyTorch support	TensorFlow support	Flax Support
ALBERT	✅	✅	✅	✅	❌
BART	✅	✅	✅	✅	✅
BERT	✅	✅	✅	✅	✅
Bert Generation	✅	❌	✅	❌	❌
BigBird	✅	✅	✅	❌	✅
BigBirdPegasus	❌	❌	✅	❌	❌
Blenderbot	✅	❌	✅	✅	❌
BlenderbotSmall	✅	❌	✅	✅	❌
CLIP	✅	✅	✅	❌	✅
CTRL	✅	❌	✅	✅	❌

شکل ۴-۱: بخشی از مدل های موجود در هاگینگ فیس

برای شروع به کار با کتابخانه Transformer نیاز است از طریق pip آن را نصب کنید. پس از نصب یکی از ساده ترین روش ها برای شروع به کار با ترنسفورمر ها استفاده از پایپلاین[۱۹] ها هستند. پایپلاین ها در واقع مسیر های از قبل تعیین شده ای هستند که برای تسک های پرکاربرد NLP در این کتابخانه قرار داده شده اند.

```
>>> from transformers import pipeline

>>> question_answerer = pipeline("question-answering")

>>> context = r"""
... Extractive Question Answering is the task of extracting an answer from a text given a question. An example of a
... question answering dataset is the SQuAD dataset, which is entirely based on that task. If you would like to fine-tune
... a model on a SQuAD task, you may leverage the examples/pytorch/question-answering/run_squad.py script.
... """
```

شکل ۴-۲: استفاده از کتابخانه ترنسفورمر

توصیه می‌شود فایل notebook [12] پیوست شده به گزارش را برای دیدن جزییات پیاده سازی و مشاهده خروجی کار بررسی کنید. چندین پایپلاین پرکاربرد در این notebook با پارامتر های مختلف بررسی شده اند. در ادامه چند مثال از پایپلاین را مشاهده میکنید.

[۱۹] Pipeline

```
In [ ]:
classifier = pipeline('sentiment-analysis')

In [ ]:
classifier('i love you')

Out[ ]:
[{'label': 'POSITIVE', 'score': 0.9998656511306763}]

In [ ]:
classifier('i hate you')

Out[ ]:
[{'label': 'NEGATIVE', 'score': 0.9991129040718079}]
```

شکل ۴-۳: استفاده از کتابخانه ترنسفورمر در مبحث آنالیز احساس

```
In [ ]:
mlm = pipeline("fill-mask")

In [ ]:
ml = mlm(f"HuggingFace is creating a {mlm.tokenizer.mask_token} that the

for i in ml:
  print(i["score"], "\t", i["token_str"])

0.17927460372447968    tool
0.1134939044713974    framework
0.05243545398116112    library
0.03493543714284897    database
0.02860247902572155    prototype
```

```
In [ ]:
mlpers = mlmpersian(f"خواهد شد [MASK] رتبه من در کنکور ارشد 1400 برابر با")

print("رتبه من در کنکور ارشد 1400 برابر با *** خواهد شد")
for i in mlpers:
  print(i["score"], "\t", i["token_str"])

رتبه من در کنکور ارشد 1400 برابر با *** میشود
0.0288712065666914    ۶۳
0.019150292500853354    ۲
0.018279979005455597    ۹۰
0.017009330913424492    ۱
0.01648857071995735    ۷
```

شکل ۴-۴: استفاده از کتابخانه ترنسفورمر در مبحث ماسکینگ

٤-٣-مثال FineTune کردن ترنسفورمر

همانطور که گفته شد ترنسفورمر ها از زبان وگرامر درک دارند (Language Understanding) یکی از قدرتمند ترین کار هایی که میتوان با ترنسفورمر ها انجام داد این است که از دانش آنها نسبت به زبان به عنوان زیر بنا برای سایر تسک های NLP استفاده کرد. این کار را میتوان با فاین تون کردن لایه آخر (Classifier) انجام داد. به این ترتیب میتوان با Freeze کردن وزن های درونی ترنسفورمر و ترین کردن لایه آخر آن روی دیتاستی معین، مدل را برای آن تسک بخصوص آماده کرد.

در مثالی که پیوست شده ما از ترنسفورمر ترین شده ی DistilBERT استفاده میکنیم و بدون نیاز به ترین کردن مجدد ترنسفورمر و تنها با ترین کردن مدل بر روی دیتاست نسبتا کوچکی از کامنت های IMDB ، توانایی دسته بندی این گونه کامنت ها را بدست آوریم[13]. توضیحات تکمیلی در مورد این کار در ویدیو چهارم وجود دارد.

جهت دسترسی به نسخه اصلی این notebook که توسط کمپانی HuggingFace آماده شده به آدرس زیر مراجعه کنید.

https://colab.research.google.com/github/huggingface/notebooks/blob/master/transformers_doc/tensorflow/custom_datasets.ipynb

۵- فصل پنجم: جمع بندی و نتیجه گیری

۵-۱-جمع بندی نهایی

در دنیای امروزی که محاسبات پیچیده و سنگین و پردازش بر روی داده های حجیم جزو جدایی ناپذیر رشد و توسعه بیزنس ها است، یافتن راه هایی برای کاهش هزینه ها و افزایش کارایی این اعمال بسیار مورد توجه واقع شده. چند سالی است که با قدرتمند تر شدن GPU ها و قابلیت های پردازش موازی آنها شبکه های عصبی عمیق بسیار اهمیت پیدا کرده اند و بسیاری از شبکه های عصبی به عنوان جهشی بزرگ در فناوری یاد میکنند[4].

ترنسفورمر ها جهشی بزرگ و رو به جلو در تمام این زمینه ها هستند. قابلیت پردازش میلیون ها داده و نگهداری میلیون ها پارامتر قابل ترین، قابلیت موازی سازی بسیار زیاد و عوامل دیگر باعث شده اند توجه زیادی به ترنسفورمر ها شود و بسیاری از سرویس های مهم و بزرگ امروزی مانند موتور جستجوی google به سرعت هر چه تمام تر از این تکنولوژی استقبال کنند و از ترنسفورمر ها به عنوان بخش اصلی عملیات خود استفاده کنند.

همچنین تحقیقات نشان داده بر خلاف چیزی که به نظر می‌رسد ترنسفورمر ها نه تنها در زمینه های NLP بلکه در بسیاری زمینه های دیگر مانند Computer Vision و ... نیز بسیار موثر واقع می‌شوند.

علاوه بر اینها تحقیقات اخیر روی ترنسفورمرها حاکی از آن است که قابلیت گسترش و افزایش کارایی ترنسفورمر ها تا چندین و چند برابر هم مهیاست و روز به روز نتایج جدید و شگفت انگیز تری از آنها مشاهده میکنیم.

اگر به مقالات چاپ شده در مورد ترنسفورمر ها دقت کنید، خواهید دید که ترنسفورمر ها روز به روز در حال رشد و قوی تر شدن هستند. معماری های جایگزینی که در این پروژه بررسی شدند مانند Linformer [6] و Performer[10] هر کدام به تنهایی چندین برابر ترنسفورمر ها را قوی‌تر کرده اند و علاوه بر آن خود این معماری ها هم عاری از نقص نیستند و امکان بهبود دادن به این معماری های جدیدتر هم به میزان قابل توجهی امکان پذیر است.

انجام کارهای پژوهشی در زمینه ترنسفورمر ها نیز بسیار داغ است و مملو از سوال پاسخ داده نشده می‌باشد. در نهایت میتوان گفت که آینده از آن Transformer هاست.

لازم است یادآور شویم که علاوه بر گزارشی که در حال مطالعه آن هستید، تعدادی اسلاید(Google slide) و چهار ویدیو یوتوب و تعدادی فایل Google Colab برای کد های عملی فصل چهار نیز موجود میباشد که در قسمت منابع قابل دسترس هستند.

مراجع

[1] Wikipedia. Recurrent neural network. https://en.wikipedia.org/wiki/Long_short-term_memory.

[2] Wikipedia. Long short-term memory. https://en.wikipedia.org/wiki/Recurrent_neural_network .

[3] Wikipedia. Gated recurrent unit. https://en.wikipedia.org/wiki/Gated_recurrent_unit .

[4] Vaswani, Ashish & Shazeer, Noam & Parmar, Niki & Uszkoreit, Jakob & Jones, Llion & Gomez, Aidan & Kaiser, Lukasz & Polosukhin, Illia. (2017). Attention Is All You Need. arXiv:1706.03762 .

[5] Alammar, Jay. Illustrated-transformer. http://jalammar.github.io/illustrated-transformer/ .

[6] Wang, Sinong & Z. Li, Belinda & Khabsa, Madian & Fang, Han & Ma, Hao. (2020). Linformer: Self-Attention with Linear Complexity. arXiv:2006.04768v3 .

[7] Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In International Conference on Learning Representations, 2020

[8] Wikipedia. Singular value decomposition. https://en.wikipedia.org/wiki/Singular_value_decomposition .

[9] Kriventsov, Stan.Linformer. Linformer: Self-Attention with Linear Complexity (paper review). https://medium.com/deep-learning-reviews/linformer-self-attention-with-linear-complexity-paper-review-70aac1d197d9 .

[10] Choromanski, Krzysztof & Likhosherstov, Valerii & Dohan, David & Song, Xingyou & Gane, Andreea & Sarlos,Tamas & Hawkins, Peter & Davis, Jared & Mohiuddin, Afroz & Kaiser, Lukasz & Belanger, David & Colwell, Lucy & Weller, Adrian. (2020). Rethinking Attention with Performers. arXiv:2009.14794v1.

[11] YannicKilcher. Rethinking Attention with Performers (Paper Explained). https://www.youtube.com/watch?v=xJrKIPwVwGM&list=PL3naYV_Z4HeEjbvIRa59x8khwaZ7gcytN&index=22 .

[12] Amirmahani, Amir Hassan & Dar, Gholamreza. Introduction to Hugging Face Transformer library.ipynb.

[13] Amirmahani, Amir Hassan & Dar, Gholamreza. Fine Tuning Transformer using IMDB dataset.ipynb.

[14] Vu, Kevin & Corp, Exxact. A Deep Dive Into the Transformer Architecture – The Development of Transformer Models. https://www.kdnuggets.com/2020/08/transformer-architecture-development-transformer-models.html.

[15] Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Alberti, Chris & Ontanon, Santiago & Pham, Philip & Ravula, Anirudh & Wang, Qifan & Yang, Li & Ahmed, Amr. (2020). Big Bird: Transformers for Longer Sequences. arXiv:2007.14062 .

[16] Campagnola, Chiara. From Transformers to Performers: Approximating Attention. https://towardsdatascience.com/from-transformers-to-performers-approximating-attention-69c88af0b11f.

[17] The A.I. Hacker - Michael Phi. Illustrated Guide to Transformers Neural Network: A step by step explanation. https://www.youtube.com/watch?v=4Bdc55j80l8&list=PL3naYV_Z4HeEjbvIRa59x8khwaZ7gcytN&index=2 .

[18] YannicKilcher. Linformer: Self-Attention with Linear Complexity (Paper Explained). https://www.youtube.com/watch?v=-_2AF9Lhweo&list=PL3naYV_Z4HeEjbvIRa59x8khwaZ7gcytN&index=20&ab_channel=YannicKilcher .

[19] Transformers and Language Models - YouTube Playlist. https://youtube.com/playlist?list=PL3naYV_Z4HeEjbvIRa59x8khwaZ7gcytN .

Abstract

For many years, LSTMs and GRUs have been the go-to method for almost every NLP[20] and language understanding task. Just like any method, LSTMs and GRUs have their own flaws and weaknesses.

In this work we discussed the creation of Transformers, an Attention based alternative to LSTMs and GRUs , that has shown to be extremely scalable and performant.

We also researched about state of the art variartions of Transformers like Performers and Linformers to name a few, and compared them against regular Transformers, discussed the ideas behind them and found out about their strengths and weaknesses.

Finally, we learned about implementing Transformers to help us with various real life problems and talked about the future of the field in genereal and how transformers can change everything!

[20] Natural language processing

Shahid Bahonar University of Kerman

Faculty of Engineering

Department of Computer Engineering

B.Sc. Thesis

Introduction to Transformers

Supervisor : Dr M.Eftekhari

By : Gholamreza Dar and Amirhasan Amirmahani

Date : 8/29/2021