معادل سازی جملات در زبان فارسی

مهسا سهیل شمائی و مهدی آصفی[1]

چکیده

در این طرح پژوهشی، سعی خواهد شد به نحو پیاده سازی فرآیند معادل سازی جملات فارسی به وسیله شبکه‌های عصبی و یادگیری ماشین اشاره شود. معادل سازی جملات همواره یکی از نیازمندی‌های نویسندگان و دانشجویان بوده است چرا که به وسیله آن می‌توان ساختارها را تغییر داد و جملات متفاوتی ایجاد کرد. همچنین این ابزار برای سایر کاربران نیز کاربرد بسیاری دارد که می‌توان به تغییر متن ایمیل یا پیامک به وسیله آن اشاره کرد. برخلاف اینکه نیاز به این ابزار به شدت در محیط آکادمیک کشور احساس می‌شود اما تا کنون گامی برای تحقق آن برداشته نشده است. به همین سو ما تلاش کردیم که این مشکل را حل، و به سمت تولید این ابزار و در دسترس قرار دادن آن به صورت رایگان حرکت کنیم.

کلمات کلیدي

هوش مصنوعی , یادگیری ماشین, پردازش زبان طبیعی, معادل سازی

۱- مقدمه

تولید جملات معادل توسط هوش مصنوعی یکی از شاخه های مهم پردازش زبان طبیعی است. برای دستیابی به این مهم نیاز است که چالش‌های متعددی پشت سر گذاشته شود که در ادامه به آنها اشاره خواهیم کرد. برای دستیابی به این مهم سعی کردیم که مطالعات سایر محققان در این زمینه که سعی در پیاده سازی این مدل در زبان انگلیسی را داشتند مطالعه، و از آنها الهام بگیریم. مدل‌های متفاوتی در این زمینه وجود داشتند که ما به بررسی آن پرداخته و بهترین آن را متناسب با شرایط انتخاب کردیم.

۲- جمع آوری داده و مشکلات آن

در انجام پروژه‌های پردازش زبان طبیعی روی داده‌های فارسی همیشه با مشکل کمبود داده روبرو می‌شویم. درنتیجه، کمیت و کیفیت داده‌ها در بحث معادل‌سازی جملات در زبان فارسی نیز کم است. به همین دلیل ما داده‌های خود را با استفاده از ترجمه‌ی دیتاست‌های زبان انگلیسی مرتبط با این مبحث به دست آوردیم. برای انجام این کار از دیتاست Paranmt 50m **استفاده** کردیم .این دیتاست شامل ۵۰ میلیون جفت جمله معادل به زبان انگلیسی است و حجمی معادل ۱۱ گیگابایت دارد.این دیتاست مشکلات عمده‌ای داشت که فرآیند استفاده از آن را دشوار می‌کرد. برای مثال تعدادی از جملات آن دقیقا یکسان بودند و در طرف مقابل تعدادی از جملات هیچ ارتباطی با هم نداشتند. بنابراین لازم بود پیش‌پردازش‌هایی در این زمینه روی آن‌ها انجام شود. با استفاده از در نظر گرفتن تعداد کلمات مشابه در جفت جملات درصد شباهت جملات یک جفت نسبت به یکدیگر را محاسبه کردیم. در نتیجه جملاتی که

دارای شباهت صددرصد هستند یعنی هیچ تفاوتی با هم نداشته و دقیقا با هم یکسان هستند و نمی‌توانند چیزی به مدل آموزش دهند را حذف کردیم. همینطور جفت جملاتی که دارای شباهت زیر ۱۵ ٪ هستند نیز تاثیری مثبتی در یادگیری مدل ندارند. به این علت که می‌شود گفت این جفت جملات تا حد زیادی معادل نیستند . همچنین، جملات با طول زیاد نیز می‌توانند باعث کند شدن روند یادگیری مدل شوند. در نتیجه ما جملاتی با طول بالاتر ۵۰ کلمه را از دیتاست حذف کردیم. همچنین حجم بالای این دیتاست بارگیری آن روی حافظه کامپیوتر را بسیار سخت می‌کرد. به همین دلیل باید سعی می‌کردیم آن را به صورت قسمت‌های کوچک‌تر وارد حافظه، و سپس پردازش‌های لازم را روی آن انجام می‌دادیم. ما توانستیم با پیاده سازی Google Translate با کمک زبان پایتون بخشی از داده‌ها را ترجمه کنیم. اما به دلیل حجم زیاد دیتاست و محدود بودن مقدار رم در گوگل کولب نمی‌توانستیم کل دیتاست را به صورت یکجا ترجمه کنیم. در نتیجه با تقسیم دیتاست به ۸ قسمت مساوی توانستیم کل دیتاست را به زبان فارسی ترجمه کنیم.

پس از ترجمه نیز مشکلات زیادی پیش پای ما قرار گرفت که می‌توان به عدم ترجمه شدن بخشی از کلمات به علت عدم وجود معادل برای آن‌ها اشاره کرد. همچنین بخشی از جملات با وجود متفاوت بودن در زبان مبدا پس از ترجمه دقیقا مشابه یکدیگر می‌شدند. برای حل این مسائل از معیار فاصله‌ی بین جملات به اسم فاصله لون‌اشتاین استفاده کردیم. فاصله لون‌اشتاین یا فاصله ویرایش در نظریه اطلاعات و علوم کامپیوتر مقیاسی برای محاسبه‌ی میزان تفاوت میان دو رشته است. فاصله لون‌اشتاین بین دو رشته به وسیله‌ی کمترین تعداد عملیات مورد نیاز برای تبدیل یک رشته به رشته دیگر معین می‌شود، که یک عملیات می‌تواند یک ضمیمه، یا جایگزینی یک کارکتر باشد. تعمیم فاصله لوناشتاین (فاصله دامرا–لون‌اشتاین) اجازه ترانهش دو کاراکتر را به عنوان یک عملیات می‌دهد. این معیار به افتخار ولادمیر لون‌اشتاین، که این فاصله را در سال 1965 مطرح کرد، نام گذاری شده‌است. همچنین از این موضوع در برنامه‌هایی که نیاز به یافتن مقدار شباهت، یا تفاوت دو رشته را دارند، مانند مقابله گر املائی، استفاده می‌شود. همچنین برای حل مشکل وجود کلمات انگلیسی اقدام به حذف کلیه داده‌ای که دارای کارکترهای انگلیسی هستند کردیم.

۱-۲- دیتاست نهایی

در جدول زیر می‌توانید بخشی از داده‌های ترجمه شده را مشاهده کنید. ما این دیتاست را به صورت دسترسی آزاد برای همگان روی کگل قرار داده‌ایم. همانطورکه مشاهده می‌شود داده‌ها در سه ستون دسته بندی شده اند. ستون اول نوع وظیفه‌ی تعریف شده را مشخص می‌کند که در این مورد مطابق فرمت رسمی مدل های 5T به صورت paraphrase تعریف شده است. دو ستون بعدی مربوط به جملات معادل به هم هستند که ستون اول جمله اصلی و ستون آخر جمله ی معادل با آن است. همانطور که مشاهده می‌شود برخی از داده‌ها دقیق نیستند اما به دلیل وجود مقدار زیاد داده این نویزها در مرحله آموزش مدل نادیده گرفته می‌شوند.

prefix	Input text	Output text
paraphrase	بنابراین من به شما نیاز دارم که از زندگی شخصی من دور بمانید.	من به شما نیاز دارم که از زندگی شخصی من دور بمانید.
paraphrase	احتمالا شما مادر بسیار سلطه پذیری داشتید.	احتمالاً به این دلیل که شما یک مادر بسیار مسلط داشتید.
paraphrase	وحشتناک به نظر می رسد.	وحشتناک به نظر می رسد
paraphrase	می دانید چه کسی پوست بدی داشت و چه کسی موهای بدی داشت.	می دانید، پوست بد و موی بد.
paraphrase	مطمئنی که نمیتونم از این موضوع با تو حرف بزنم	مطمئنی که من تو را از آن حرف نمی زنم
paraphrase	ما الان به اندازه کافی نزدیک هستیم که دندان های ما را حس کنند.	اکنون به اندازه کافی نزدیک هستیم تا دندان هایشان را به آنها نشان دهیم.
paraphrase	حتی اگر چیز زیادی باقی نمانده باشد، مقدار کمی را به ما بدهید.	حتی اگر چیز زیادی باقی نمانده باشد، کمی از آنچه باقی مانده است به ما بدهید.
paraphrase	در بند ۱۸ به این اصل اشاره شده است.	این اصل در بند ۱۸ ذکر شده است .
paraphrase	این سرنوشت من بود، فکر می کنم.	حدس می زنم این سرنوشت من است.

جدول ۱: تعدادی از نمونه جملات موجود در دی‌تاست

۳- مدل

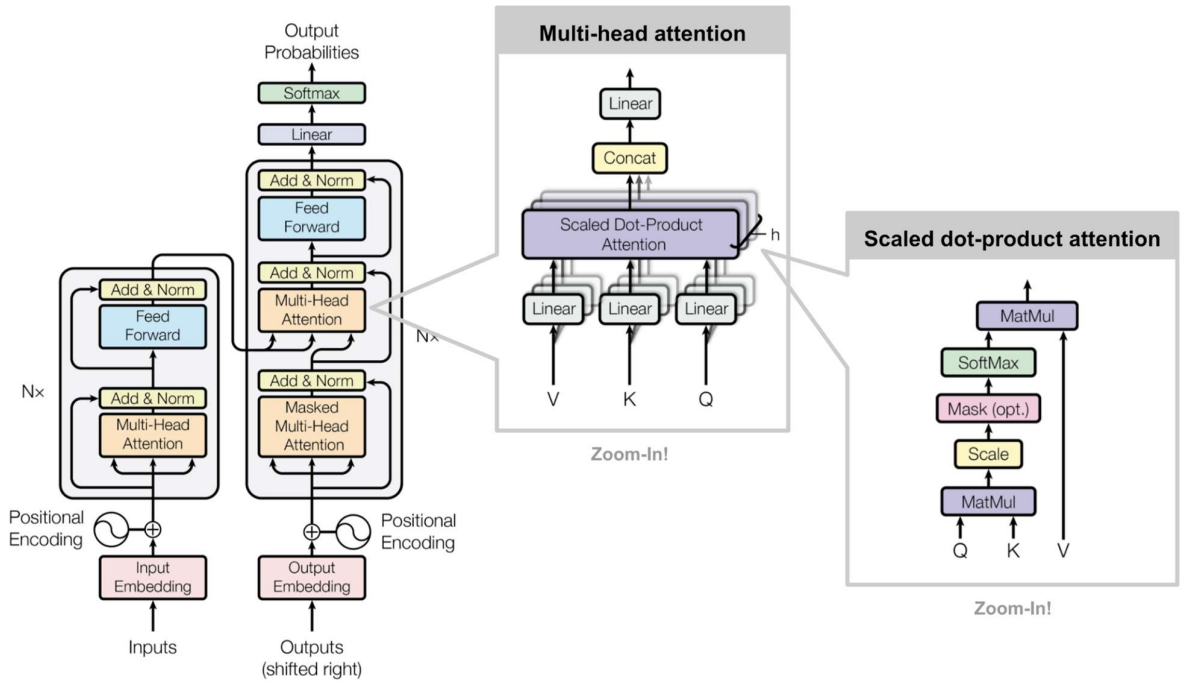
در حال حاضر بهترین مدل‌ها در زمینه پردازش زبان طبیعی مدل هایی با ساختار Transformers هستند. مدل(T5Transformer Transfer Text-to-Text) T5 نیز از همین ساختار استفاده کرده است. T5 یک مدل زبانی است که توسط Google توسعه داده شده و هدف آن بهبود عملکرد یادگیری انتقالی است. یادگیری انتقالی به این معناست که در ابتدا یک مدل را برای یک عمل که داده‌های غنی و جامعی دارد پیش آموزش می‌دهیم. سپس مدل را برای عمل مورد نظر خود با استفاده از داده‌های متناسب با عمل بهینه کنیم. در پردازش زبان طبیعی اغلب از این روش استفاده می‌شود. به دلیل اینکه با پیش آموزش مدل روی حجم زیادی از داده ، مدل درک بهتری از زبان و ساختار آن پیدا خواهد کرد و می‌تواند سریع‌تر عمل مورد نظر را یاد بگیرد. ما از مدل mT5 در این پروژه استفاده

کردیم که ساختاری مشابه با مدل 5T دارد. تنها تفاوت این دو مدل در دیتاست‌هایی است که روی آن پیش‌آموزش صورت گرفته است. مدل 5mT روی دیتاست (multilingual Colossal Clean Crawled Corpus) mC4پیش آموزش داده شده است. دیتاست 4mC شامل متن‌هایی از بستر صفحات وب هست که با استفاده از چند خزنده جمع آوری شده‌اند. این دیتاست حاوی داده‌های تمیز است. با استفاده از روش‌های زیر این دیتاست پاکسازی شده است.

- جملاتی که با علایم نگارشی مرسوم مانند نقطه، علامت سوال و علامت تعجب به پایان نرسیده است حذف شده‌اند.
- هر صفحه‌ای که شامل کلمات توهین آمیز است حذف شده است.
- هر خطی که شامل کلمه JavaScript بوده حذف شده است. بدلیل پرهیز از جمع آوری خطاهای این زبان برنامه نویسی که در صفحات وب ظاهر می‌شود.
- با حذف هر صفحه ای که شامل براکت {} می‌شود. سورس کدها پاک شده اند.
- جملات که شامل تکرار کلمات هستند با بازه ۳ کلمه تکراری در هر جمله حذف شده است.

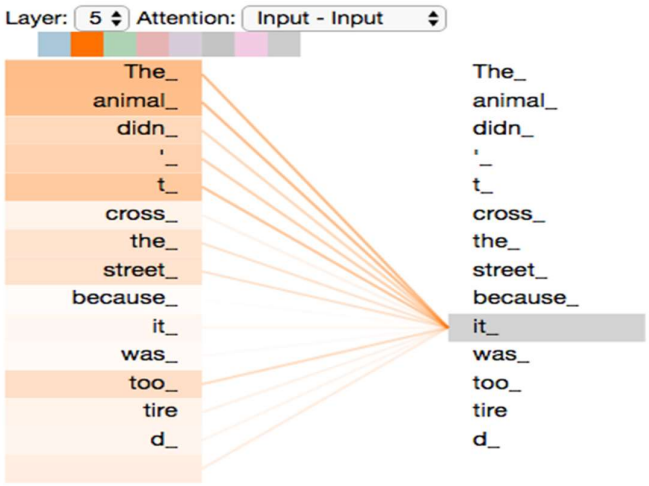
این دیتاست شامل داده های متنی به ۱۰۸ زبان مختلف است که زبان فارسی یکی از آنهاست. حجم داده فارسی موجود در این دیتاست برابر با ۲۲۰ گیگابایت است در نتیجه این مدل درک مناسبی از زبان فارسی دارد. در نتیجه انتخاب مناسبی برای این پروژه است.

۱-۳- ساختار مدل



شکل ۱: ساختار کلی Transformer ها (مدل 5mT نیز یک مدل با همین ساختار است.)

همانطور که در شکل ۱ مشاهده می‌کنید، ساختار مدل ۵mT یک ساختار استاندارد Transformer هست. این مدل متشکل از دو قسمت کدگذارEncoder و کدگشا Decoder است بعلاوه در آخر یک لایه شبکه عصبی خطی (Linear) دارد که روی آن تابع SoftMax اعمال می‌شود و خروجی نهایی تولید می‌شود. قسمت Encoder دارای دو لایه Self -Attention و Feed Forward است. Self-Attention با استفاده از مکانیزم Attention ارتباط میان دو کلمه در جمله را پیدا می‌کند. این به ما کمک می‌کند که درک درستی از جمله پیدا کنیم. به عنوان مثال جمله زیر را در نظر بگیرید: من آب درون بطری را داخل لیوان ریختم تا آن پر شود. در این جمله واضح است کلمه آن به کلمه لیوان اشاره دارد. حال جمله بعدی را در نظر بگیرید: من آب درون بطری را داخل لیوان ریختم تا آن خالی شود. همانطور که می بینید با تغییر دادن فقط یک کلمه در جمله معنای کلمه آن تغییر کرد. اینجا کلمه آن به کلمه بطری اشاره دارد.

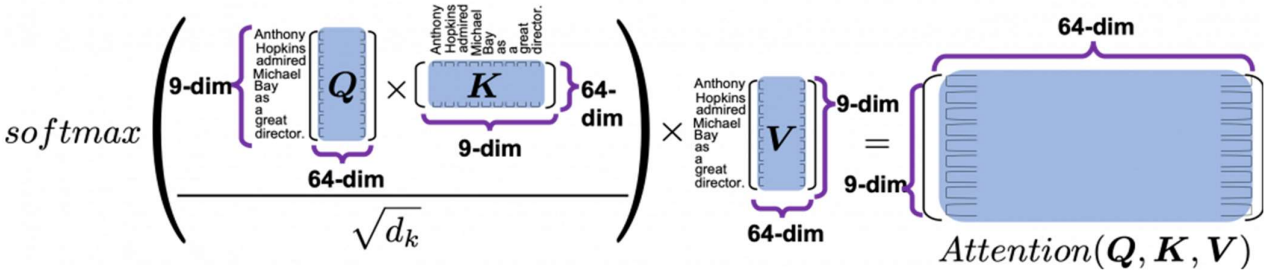


شکل ۲: نشان دهنده لایه Self-Attention (کلمه it در جمله به کدام کلمه اشاره دارد؟)

با استفاده از مکانیزم Attention می‌توان ارتباط کلمه را با سایر کلمات در جمله با استفاده از مفهوم کلی جمله محاسبه کرد. در عمل مکانیزم Attention با استفاده از سه بردار Query، Keys و Values عددی را به عنوان میزان ارتباط بین دو کلمه در جمله نشان می دهد. بردار Query در حقیقت همان بردار word embedding کلمه‌ای است که می خواهیم ارتباط آن را با کل کلمات در جمله بدست آوریم. بردار Keys از زیر هم قرار دادن بردارهای word embedding تمام کلمات یک جمله بدست می‌آید. در اینجا با توجه به این که ما به دنبال کلمه‌ای هستیم که بیشترین ارتباط را با کلمه ای اولی که انتخاب کردیم داشته باشد، در نتیجه مقدار Values با Keys یکسان است.

پس از مقدار دهی بردارهای که در بالا معرفی شد ابتدا مقادیر Query و Keys ضرب داخلی می‌شوند. نتیجه آن ماتریس شباهت (Similarity) می‌شود که بعد از اعمال کردن تابع SoftMax روی آن برداری بدست می آید که در حین فرآیند آموزش مدل بهبود پیدا می کند یعنی همان بردار وزن ها. سپس بردار وزن را در بردار Values ضرب نظیر به نظیر می‌کنیم و مقادیر بدست آمده را جمع می کنیم. مقدار بدست آمده همان مقدار خروجی لایه Attention است. اما در مدل ۵T از لایه های Multi-Head Attention استفاده شده است. در این نوع لایه ها در ابتدا با توجه به تعداد سرها (Heads)، بردار word embedding را تقسیم می‌کنیم. سپس هر کدام از قسمت‌ها را به یکی از سرهای

این لایه به عنوان ورودی می‌دهیم. در واقع به تعداد سرها در این لایه ما لایه Attention داریم. در آخر مقادیر تمام سرها با هم جمع شده و به عنوان خروجی به لایه بعدی که همان لایه Feed Forward است داده می شود.

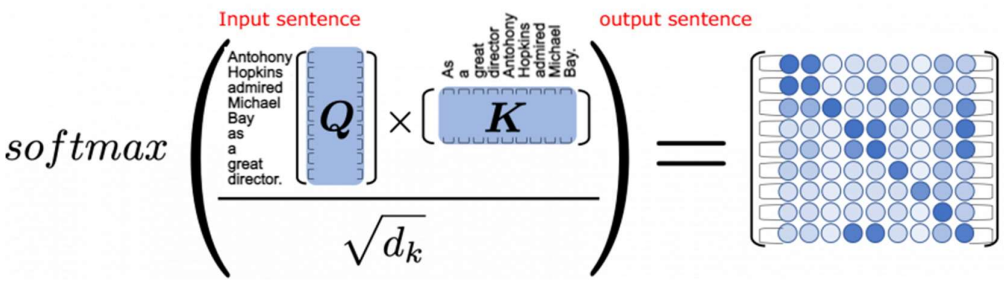


شکل۳: Queryو Value و Key در لایه Self-attention (مقادیر هر ماتریس شامل خروجی word embeding برای هر کلمه است.)

شبکهFeed Forward در قسمت Encoder یک شبکه شامل وزن‌هایی است که در طی فرآیند یادگیری مقدار مناسب را بدست می‌آورند. پارامترهای لایه‌های Feed Forward در کل دو سوم از تمام پارامترهای قابل آموزش در یک مدل Transformers را تشکیل می‌دهند. در حقیقت این لایه‌ها نقش حافظه Key-Value را دارند. در لایه Attention کل جمله به صورت دنباله ای از کلمات وارد لایه میشوند و وابستگی بین کلمات و موقعیت آن ها موجود است اما در این شبکه هر کلمه از جمله به صورت جدا وارد یک شبکه تمام متصل(Fully Connected) می شود و به صورت موازی بدون وابستگی به م.قعیت کلمه در جمله \ردازش میشود. وزن های این شبکه طی فرایند آموزش بهبود میابند . به علاوه این شبکه با استفاده از وزن هایش باعث می شود که مقادیر Key-Value حفظ شده و به لایه بعدی منتقل شود زیرا در صورت موجود نبودن این لایه این مقادیر در شبکه حفظ نمیشوند. فرآیند یادگیری در این شبکه مانند یک شبکه عصبی خطی است و با استفاده روش انتشار رو به عقب صورت میگیرد. این روش با بدست آوردن مشتق خطا در لایه پایانی وزن ها را بهبود می‌دهد. به علاوه ابعاد ورودی و خروجی در لایه های Feed Forward با هم یکسان است. در زیر تابع این شبکه را مشاهده می کنید که شامل دو لایه کاملا متصل است و یگ تابع ReLu برای غیر خطی سازی نتیجه در بین این دو زیر لایه قرار گرفته است.

$$FFN(x) = max(0, xW_1 + b_1)W_2 + b_2$$

قسمت Decoder علاوه بر دو لایه‌ای که در قسمت Encoder توضیح داده شد یک لایه دیگر به نام Attention Decoder-Encoder دارد. این لایه همانند Self-Attention عمل می‌کند با این تفاوت که ارتباط بین کلمات در جمله ورودی و جمله خروجی را محاسبه می‌کند. در اصل تفاوت یک لایه Self-Attention با این لایه فقط در مقادیر بردار Keys است. در این لایه Query با استفاده از یک جمله از مثال‌های آموزشی ورودی انتخاب می‌شود و Keys از مثال های آموزشی خروجی است.لازم به ذکر است که قسمت Decode جمله خروجی را به عنوان ورودی دریافت می‌کند و جمله ورودی اصلی را از قسمت Encoder می‌گیرد.



شکل ۴: Query و Key در لایه Attention Decoder-Encoder (مقادیر هر ماتریس شامل خروجی word embeding برای هر کلمه است.)

بعد از لایه‌های Decoder و Encoder به لایه Linear می‌رسیم. همانطور که از اسمش مشخص است این یک شبکه عصبی خطی است که پردازش روی هر جز از دنباله را به صورت تک و بدون در نظر گرفتن سایر اعضای دنباله انجام می‌دهد. در اینجا این لایه روی هر کلمه از جمله بدون در نظر گرفتن کلمه بعدی در جمله یا کلمه مرتبط به این کلمه در جمله خروجی انجام می‌دهد. در اصل این لایه خروجی لایه Decoder که یک بردار هست را به یک کلمه تبدیل میکند. فرض کنید مدل ما ۱۰۰۰۰ کلمه فارسی را که در طی فرایند آموزش یاد گرفته می شناسد(تعداد لغاتی که میخواهیم شناخته شود قبل از شروع آموزش مشخص می شود). در نتیجه خروجی این لایه یه بردار با اندازه ۱۰۰۰۰ است که هر اندیس متعلق به یک کلمه است. این لایه امتیاز هر کلمه را با توجه به بردار ورودی مشخص می کند.

در آخر خروجی لایه قبل وارد یک تابع SoftMax می شود فرمول این تابع به صورت زیر است:

$$s\left(x_i\right) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}$$

این تابع باعث می‌شود که اعداد خروجی از لایه قبل شفاف تر شوند زیرا با استفاده از این لایه امتیاز ها تبدیل به درصد می شوند (یعنی جمع مقادیر ۱۰۰۰۰ خانه بردار خروجی باید برابر یک باشد) در نتیجه کلمه ای که بیشترین احتمال را دارد به عنوان خروجی مشخص می شود.

۲-۳- مراحل آموزش مدل

آموزش این مدل در دومرحله صورت گرفته. است:

- یک: آموزش بدون نظارت روی داده‌های ۴mC
- دو: آموزش نظارت شده روی داده‌های جمع‌آوری شده برای معادل‌سازی جملات

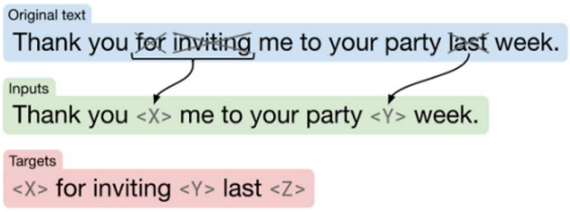
۱-۲-۳- آموزش بدون نظارت

معرفی کنندگان مدل ۵mT برای آموزش اولیه مدل خود سه روش آموزش بدون نظارت را آزمایش کردند:

۱. Language Modeling : این روش شامل حدس لغت بعدی در جمله با توجه به کلمات قبلی است.

۲. Deshuffling: در این روش جایگاه تمام لغات در جمله به صورت تصادفی تغییر می‌کند و مدل سعی در پیدا کردن شکل درست جمله دارد.
۳. Corrupting Spans: این روش دنباله از کلمات در جمله را پنهان می کند و مدل کلمات مخفی را پیش بینی می کند.

در این مرحله روش Spans Corrupting یا همان Objective Denoising روی داده های دیتاست ۴mC اعمال میشود.
روش Objective Denoizing به این صورت عمل می‌کند که متنی را از دیتاست انتخاب می‌کند، سپس قسمتی از جمله که شامل تعداد دلخواهی از کلمات است را می پوشاند، در ادامه سعی می‌کند که با استفاده از مدل، کلمات پوشیده شده را حدس بزند. به این ترتیب عمل یادگیری فقط با استفاده از یک متن برچسب گذاری نشده انجام می‌شود. این قسمت از آموزش توسط افرادی که مدل را معرفی و پیاده سازی کردند ارائه شده است. به زبانی دیگر ما یک مدل از پیش آموزش دیده شده داریم.



شکل ۵: نحوه پی‌ش آموزش مدل ۵mT.

عمل پیش آموزش روی مدل انجام می شود تا مدل با ساختار زبان آشنا بشود و کلمات را بشناسد. همانطور که در شکل ۵ میبینید بعد از جمع آوری دیتا با استفاده از یک جمله بدون لیبل میتوان یک خروجی و ورودی برای آموزش مدل ایجاد کرد. به این صورت که چند کلمه به صورت تصادفی از جمله برداشته می شود و به جای آن ها نشانه هایی قرار داده میشود در اینجا این نشانه ها <X> و <Y> است. این جمله جدید به عنوان ورودی به مدل داده می شود. کلمات برداشته شده هم به صورتی که در شکل ۵ با قرمز مشخص شده اند به عنوان خروجی یا هدف به مدل داده می شود. مدل با استفاده از این جمله و کلمات خروجی که باتوجه به ورودی پیش بینی می شود می تواند تابع خطا را محاسبه کند و وزن ها را بهبود دهد. با تکرار این کار روی دیتاست های حجیم مدل به مرور با ساختار جملات و نقش کلمات در جملات یک زبان آشنا می شود.

۲-۲-۳- آموزش با نظارت

در این مرحله ما با استفاده از مدل ۵mT و دیتاستی که جمع آوری کرده‌ایم، مدل را برای عمل معادل سازی جملات آموزش می‌دهیم. برای پیاده سازی این پروژه ما از زبان پایتون استفاده کردیم که کتابخانه های غنی و متعددی در زمینه پردازش عمیق و پردازش زبان‌های طبیعی دارد. در دیتاست ما هر مثال آموزشی شامل دو جمله می‌شود. جمله اول به عنوان جمله اصلی و ورودی مدل، و جمله دوم به عنوان جمله معادل سازی شده با استفاده از جمله اصلی که همان خروجی یا همان هدف مدل است. درنتیجه، روش آموزش استفاه شده روش یادگیری با نظارت

است. در آموزش با نظارت بعد از بدست آوردن خروجی مدل مقدار خطا محاسبه می‌شود و از آن برای بهبود مقادیر وزن در شبکه استفاده می‌شود.

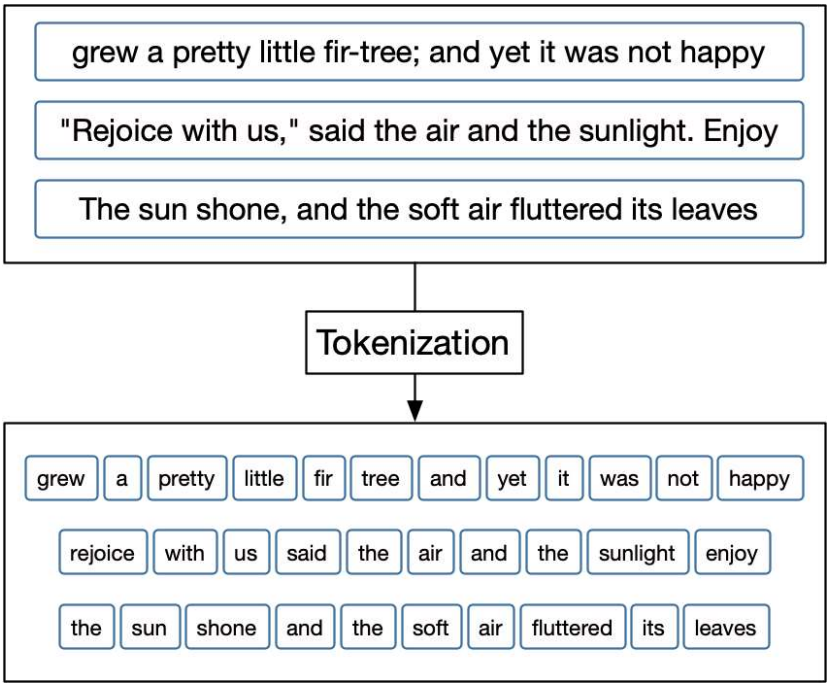
برای استفاده از مدل از پیش آموزش دیده mT۵ نیاز بود که در ابتدا آن را روی سیستم خود پیاده سازی کنیم. محیط Google Colab برای پیاده سازی این پروژه مناسب است اما متاسفانه محدودیت‌هایی برای Ram و GPU دارد که روی عملکرد مدل ما تاثیر گذاشته است. به دلیل سنگین بودن مدل کامل mT۵ ما از مدل small-mT5 استفاده کردیم. امروزه تقریبا تمام مدل های بروز در تمام زمینه‌های هوش مصنوعی بروی پلتفرم Hugging Face پیاده سازی شده‌اند. در نتیجه ما با استفاده از اضافه کردن کتابخانه این پلتفرم به محیط برنامه نویسی خود توانستیم با قطعه کدی مدل mT۵ را روی Ram سرور خود پیاده سازی کنیم. سپس برای بهبود دادن مدل خود با توجه به دیتاستی که آماده کردیم نیازمند این هستیم که توابع و کلاس هایی را متناسب با مدل آماده کنیم که بتوانیم عمل آموزش را انجام دهیم.

یکی از اولین کلاس‌ها که می‌توانید در کدهای پروژه مشاهده کنید کلاس Dataset است. این کلاس داده‌های ما را که به صورت یک فایل csv است دریافت می‌کند و تغییرات لازم را طوری انجام می‌دهد که بتوان داده‌ها را به عنوان ورودی در قدم‌ها و دوره‌های مختلف آموزش و ارزیابی به مدل وارد کرد. این کلاس با ارث بری از کلاسی به همین نام در کتابخانه Pytorch پیاده سازی شده و باید با توجه به استانداردهای مدل نوشته می‌شود. عمل tokenizing در این قسمت انجام می‌شود. این عمل در قسمت توکن سازی توضیح داده شده است.

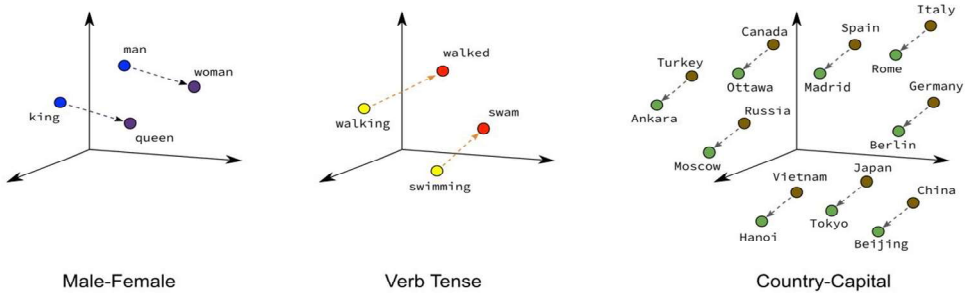
قسمت اصلی دیگری که در واقع تمام فرآیند یادگیری را انجام می‌دهد کلاس T5FineTuner است. این کلاس با استفاده از ارث‌بری از کلاسی در کتابخانه Pytorch Lightning ایجاد می شود. این کتابخانه به تازگی منتشر شده و به کمک برنامه نویسان آمده است. با استفاده از این کتابخانه می‌توان مراحل آموزش یک مدل را با خط کدهای کمتری پیاده سازی کرد. این مراحل شامل حرکت رو به جلو در لایه‌ها، انجام قدم‌های یادگیری در هر دوره از آموزش، انجام قدم‌های ارزیابی با استفاده از داده‌های Validation و بهینه‌سازی ابر پارامترها می‌شود. توابع دیگر در این کلاس موجود است که از مهمترین آن‌ها می‌توان به Data Loader ها اشاره کرد. این توابع مثال‌های آموزشی و آزمایشی را به صورت دسته‌های چندتایی به عنوان ورودی به مدل می دهند. اینکار به دلیل حجم محدود رم صورت می‌گیرد و باعث میشود که حافظه رم کاملا سرریز نشود.

۳-۳- توکن سازی (tokenization)و embedding

Tokenization اولین مرحله در هر وظیفه‌ی NLP است و تأثیر مهمی بر روند آموزش مدل دارد. توکنایزر داده‌های بدون ساختار و متن زبان طبیعی را به تکه‌هایی از اطلاعات که می‌توانند به عنوان عناصر مجزا در نظر گرفته شوند، تجزیه می‌کند.



شکل ۶: چگونگی عملکرد Tokenizer.



شکل ۷: چگونگی عملکرد Word Embedding.

توکنایز کردن دارای انواع مختلفی است که ما در این پروژه مطابق با استاندارد مدل‌های ۵T از تکنیک sub-word برای توکن‌سازی استفاده کردیم. کتابخانه‌ای که برای اینکار انتخاب کردیم کتابخانه sentencePiece می‌باشد که توسط گوگل طراحی شده است. sentencePiece یک توکنایزر و دی- توکنایزر (تبدیل توکن به متن) بدون نظارت است که عمدتاً برای سیستم‌های تولید متن مبتنی بر شبکه عصبی است که در آن اندازه واژگان قبل از آموزش مدل عصبی تعیین شده است. همچنین میدانیم که کامپیوترها تنها قادر به درک اعداد هستند. پس برای فهم جملات و کلمات به زبان انسانی نیاز به تبدیل تک تک کلمات به بردارهایی معادل هستند. به این عملیات word embedding گفته می‌شود. embedding یک نمایش آموخته شده برای متن است که در آن کلماتی که معنی یکسانی دارند بازنمایی مشابهی دارند. این رویکرد در نمایش کلمات و اسناد را می‌توان یکی از پیشرفت‌های کلیدی یادگیری عمیق در مشکلات چالش برانگیز پردازش زبان طبیعی در نظر گرفت. word embedding در واقع دسته‌ای از تکنیک‌ها هستند که در آن کلمات جداگانه به‌عنوان بردارهایی با ارزش حقیقی در یک فضای برداری از پیش تعریف شده نمایش داده می‌شوند. هر کلمه به یک بردار نگاشت می‌شود و مقادیر بردار توسط یک شبکه عصبی آموخته می‌شوند،

و از این رو این تکنیک اغلب در حوزه یادگیری عمیق قرار می‌گیرد. می‌توان گفت word embedding مهم ترین نقش در دقت مدل در ساخت جملات جدید دارد. هر چه این قسمت رابطه بین کلمات در یک زبان را بهتر متوجه شده باشد می‌تواند جملاتی را تولید کند که از نظر معنایی و ساختاری به زبان انسان نزدیک‌تر هستند. شکل زیر نشان می‌دهد که چگونه با استفاده از فاصله لغات در فضا می توان به ارتباط آن ها از هم پی برد.

قسمت word embedding درون لایه های Encoder و Decoder قرار دارد. یکی از قسمت هایی که در آموزش بدون نظارت بهبود پیدا کرده است همین قسمت است. در نتیجه ما با استفاده از word embedding و tokenization آماده در مدل ۵mT می‌توانیم زبان فارسی را به مدل آموزش دهیم و آن را برای عمل بهینه سازی آماده کنیم.

۴-۳- ابرپارامترها

به طور معمول بهینه سازی ابرپارامتر ها در دقت نتایج نهایی بسیار تاثیر گذار است. بهینه سازی مناسب ابرپارامترها می‌تواند باعث افزایش دقت خروجی شود. اما در معادل سازی جملات هدف ما رسیدن به حداکثر دقت نیست. ما می‌خواهیم جملاتی معادل جمله ورودی تولید کنیم، نه دقیقا مشابه. در نتیجه بهتر است برای دقت مدل حد آستانه‌ای در نظر گرفته شود. درنتیجه مدل قادر می‌شود که برای یک جمله، چند جمله متفاوت با دقت‌های مختلف ارائه دهد. در ادامه به شرح چند ابرپارامتر مهم در فاز آموزش می‌پردازیم.

ما از الگوریتم AdamOptimizer برای آموزش مدل خود استفاده کردیم این الگوریتم برای بهبود دادن مقادیر وزن مدل در هنگام آموزش مدل استفاده می‌شود. در این الگوریتم مقدار اپسیلون برابر با $\varepsilon = 1 * 10^{-8}$ است. با توجه به محدود بودن منابع پردازشی در دسترس مقدار batch برای فاز آموزش و ارزشیابی مقدار ۵ در نظر گرفته شده است. مقدار ابرپارامتر learning rate با نتیجه گیری از کار های مشابه مقدار $3 * 10^{-4}$ را دارد. همچنین به علت محدود بودن زمان در سرور های Kaggle و Colab ما مقدار Epoch که همان تعداد تکرار شدن مدل برای آموزش روی تمام دیتاست هست را برابر ۲ قرار دادیم .بعلاوه مدل ۵mT شامل در اندازه‌های مختلف با توجه به تعداد پارامترها و لایه‌ها ارائه شده است. به دلیل محدودیت پردازشی ما از مدل ۵small-mT استفاده کرده‌ایم. این مدل شامل ۶۰ میلیون پارامتر، ۶ بلاک ترنسفورمر و در هر بلاک هر کدام از لایه های multi-head attention داری ۸ سر هستند.

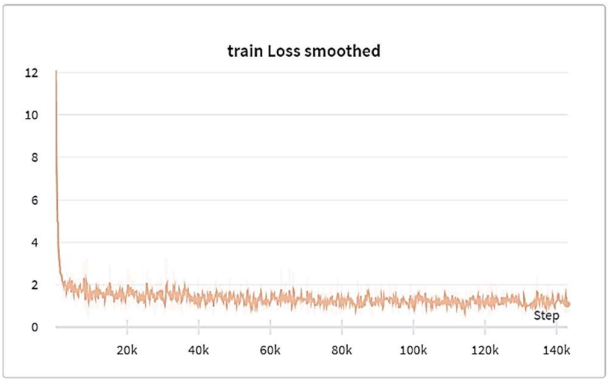
Model size variants

Model	Parameters	# layers	d_{model}	d_{ff}	d_{k}
Small	60M	6	512	2048	6
Base	220M	12	768	3072	6
Large	770M	24	1024	4096	6
3B	3B	24	1024	16384	1

شکل ۷: ابعاد و تعداد پارامترهای انواع دی‌گر مدل

۴- نتایج

برای آموزش چنین مدل‌های سنگینی همواره نیاز به وجود GPU (کارت گرافیک) قدرتمند است. این کارت‌های گرافیک امکان موازی‌سازی عملیات‌های لایه‌ها را مهیا می‌سازند و تاثیر چشم‌گیری در سرعت پردازش خواهند داشت. برای آموزش این مدل ما از کارت گرافیک NVIDIA TESLA K ۸۰ استفاده کردیم. این کارت گرافیک دارای ۲۴ گیگ حافظه GDDR۵ می‌باشد. در زیر نمودار خطای مدل پس از ۱۲ ساعت آموزش را مشاهده میکنید.



شکل ۸: نمودار تغی‌ی‌رات خطا در حی‌ن آموزش

همانطور که مشاهده می‌شود خطای مدل در ابتدا بسیار بالاست و علت آن را می‌توان به عدم آشنایی مدل به عمل معرفی شده ریشه‌یابی کرد. اما این خطا پس از چند step با شیب زیادی کاهش میابد که گواه این است که مدل به درک درستی از مسئله نزدیک شده است و اکنون می‌تواند جملات را به گونه‌ای تغییر دهد که خطای پایینی حاصل شود به این معنا که جمله خروجی مدل که معادل سازی شده با استفاده از ورودی است شباهت زیادی به جمله هدف یا همان خروجی در مثال های آموزشی دارد .همانطور که در تصویر بالا مشاهده می‌کنید ۱۴۰ هزار قدم از یادگیری طی شده است. هر قدم شامل آموزش مدل توسط یک دسته ۵ تایی از مثال‌های آموزشی می‌شود .بدلیل حجم بالای دیتای جمع آوری شده و در دسترس نبودن سیستم پر قدرت که توانایی پردازش روی تمام دیتاست را داشته باشد ما دیتاست را به ۸ قسمت تقسیم کردیم و مدل را تنها روی یکی از این قسمت ها آموزش داده ایم .در یکی از ۸ قسمت داده‌ی ما تقریبا ۶۱۰ هزار نمونه آموزشی موجود است . درنتیجه با ضرب ۵ در تعداد قدم متوجه می‌شویم که به دلیل محدودیت زمانی kaggle در استفاده از GPU مدل ما تنها روی ۷۰۰ هزار داده آموزش دیده و نتواسته ۲ بار کامل روی ۸/۱ داده آموزش ببیند.

۵- آزمایش

بعد از پایان یافتن آموزش مدل، فایلی را شامل مقادیر وزن‌هایی که در این ۱۴۰ هزار قدم آموزش دیده‌اند و بهبود پیدا کرده‌اند داریم که می‌توان با استفاده از آن، توانایی مدل را بسنجیم. هر چند که مدل ما به اندازه کافی آموزش ندیده است، اما به جهت پیش آموزشی که دیده بود می‌تواند به درستی زبان فارسی را درک نموده و جملات درستی از لحاظ معنایی و قواعدی تولید کند. در زیر چند نمونه از مثال‌های آزمایش شده را قرار دادیم. شما می‌توانید با مراجعه به این نوت بوک در وبسایت kaggle مدل را با مثال‌های آموزشی خود نیز امتحان کنید.

```
paraphraser("امروز هوا آفتابی و بسیار گرم است.")
```

```
'امروز هوا آفتابی و گرم است.'
```

```
paraphraser("حتمالا شما مادر بسیار سلطه پذیری داشتید.")
```

```
'شما باید مادر بسیار سلطه پذیری داشته باشید'
```

شکل ۸ **:**همانطور که در مثال دوم ملاحظه می کنید با وجود غلط املایی در جمله ورودی مدل توانسته جمله را معادل سازی کند و همچنین غلط املایی را رفع نماید.

۶- توسعه های آتی

این مدل با وجود تلاش‌های ما دارای نقصان‌هایی است که قابل بهبود می‌باشند. مهم‌ترین گام برای بهبود این مدل آماده‌سازی دیتاست دقیق از جملات معادل فارسی است که این مهم بدون صرف وقت و هزینه زیاد میسر نخواهد بود. در گام بعدی باید سیستم‌های گرافیکی قوی‌تری اختیار کرد که پروسه آموزش مدل را سریع‌تر و با دقت بیشتری پیش برد. پس از انجام گام های یادشده می‌توان این مدل را به مرحله استفاده تجاری و درآمد زایی رساند.

مراجع

[١] Bird, Jordan J., Anikó Ekárt, and Diego R. Faria. "Chatbot Interaction with Artificial Intelligence: human data augmentation with T5 and language transformer ensemble for text classification." *Journal of Ambient Intelligence and Humanized Computing* (2021): 1-16.

[٢] Hudson, G. Thomas, and Noura Al Moubayed. "Ask me in your own words: paraphrasing for multitask question answering." *PeerJ Computer Science* 7 (2021): e759.

[٣] Chada, Rakesh. "Simultaneous paraphrasing and translation by fine-tuning transformer models." *arXiv preprint arXiv:2005.05570* (2020).

[٤] Witteveen, Sam, and Martin Andrews. "Paraphrasing with large language models." *arXiv preprint arXiv:1911.09661* (2019).

[٥] Egonmwan, Elozino, and Yllias Chali. "Transformer and seq2seq model for paraphrase generation." *Proceedings of the 3rd Workshop on Neural Generation and Translation*. 2019.

[٦] Xue, Linting, et al. "mT5: A massively multilingual pre-trained text-to-text transformer." *arXiv preprint arXiv:2010.11934* (2020).

[٧] Chi, Zewen, et al. "mT6: Multilingual pretrained text-to-text transformer with translation pairs." *arXiv preprint arXiv:2104.08692* (2021).

[٨] Wieting, John, and Kevin Gimpel. "ParaNMT-50M: Pushing the limits of paraphrastic sentence embeddings with millions of machine translations." *arXiv preprint arXiv:1711.05732* (2017).

[٩] Wieting, John, Jonathan Mallinson, and Kevin Gimpel. "Learning paraphrastic sentence embeddings from back-translated bitext." *arXiv preprint arXiv:1706.01847* (2017).

[١٠] Wieting, John, et al. "Towards universal paraphrastic sentence embeddings." *arXiv preprint arXiv:1511.08198* (2015).

[١١] Hu, J. Edward, et al. "Parabank: Monolingual bitext generation and sentential paraphrasing via lexically-constrained neural machine translation." *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. No. 01. 2019.

[١٢] Grover, Khushnuma, et al. "Deep learning based question generation using t5 transformer." *International Advanced Computing Conference*. Springer, Singapore, 2020

[١٣] The Illustrated Transformer – Jay Alammar – Visualizing machine learning one concept at a time. (jalammar.github.io)