
بسم الله الرحمن الرحيم

دانشگاه صنعتی اصفهان

دانشکده مهندسی برق و کامپیوتر

گزارش پروژه‌ی کارشناسی

عنوان پروژه:

سیستم پرسش و پاسخ با استفاده از یادگیری ماشین

استاد راهنما:

دکتر مهران صفایانی

توسط:

مرضیه نوری

بهمن ۱۴۰۰

فهرست مطالب

چکیده

امروزه کم‌تر پیش می‌آید در روز حداقل یک بار برای یافتن پاسخ سوالی که داریم، به موتورهای جستجویی نظیر گوگل متوسل نشویم. آمار[1] نشان می‌دهد روزانه ۵/۶ میلیارد جستجو روی گوگل انجام می‌شود و این تنها مربوط به یکی از چندین موتور جستجوی مطرح است. با افزایش سرعت تولید داده‌ها، از جمله داده‌های متنی، نیاز به سیستم‌هایی که علاوه بر استخراج اسناد مرتبط با سوال ما، پاسخ مستقیم سوالمان را نیز به ما برگردانند، بیشتر شده است. در این پروژه، برای آنکه قدمی در راستای حل این چالش برداریم، مدلی طراحی کرده‌ایم که مبتنی بر مبدل ParsBERT[2] بوده و روی مجموعه‌داده‌ی PersianQA[3] که یک مجموعه‌ی پرسش و پاسخ فارسی است، تنظیم دقیق[4] شده است. در این پروژه نهایتا با تنظیم ابرپارامترها و آزمایش یازده حالت، به بالاترین امتیاز F1[5] ٪۵۹,۱۲ و امتیاز تطابق دقیق[6] (EM) ٪۴۲,۷۹ رسیده‌ایم.

[1] https://blog.hubspot.com/marketing/google-search-statistics
[2] transformer
[3] Persian Question Answering
[4] fine-tune
[5] F1 score
[6] exact match (EM)

فصل اول

مقدمه

امروزه با توجه به رشد روزافزون روند تولید داده‌ها، اهمیت آنالیز و بهره جستن از آن‌ها به کمک یادگیری ماشین، بر کسی پوشیده نیست؛ یکی از زیرشاخه‌های مهم این حوزه، پردازش زبان‌های طبیعی[1] است که خود شاخه‌ای وسیع به شمار می‌آید. هدف در پردازش زبان‌های طبیعی آنست که سیستم‌های کامپیوتری بتوانند زبان‌های طبیعی مانند انگلیسی و فارسی را درک کنند و در اموری مانند طراحی سیستم‌های پرسش و پاسخ[2]، ترجمه‌ی ماشینی[3]، تحلیل احساس[4]، تشخیص گفتار[5]، خلاصه‌سازی اسناد[6] و ... به کمک ما بیایند.

[1] Natural Language Processing (NLP)
[2] Question Answering Systems
[3] Machine Translation
[4] Sentiment Analysis
[5] Speech Recognition
[6] Document Summarization

همانطور که اشاره شد، یکی از کاربردهای پردازش زبان‌های طبیعی، بحث طراحی سیستم‌های پرسش و پاسخ است. در دنیایی که روزانه توده‌ی عظیمی از اطلاعات تولید می‌شود، به نظر می‌رسد آنچه که در حالت عادی، موتورهای جستجو در دسترس ما قرار می‌دهند، یعنی اسنادی مرتبط با سوالات ما، به اندازه‌ی گذشته کارآمد نیست؛ چرا که هدف اغلب افراد از جستجو بر بستر وب، یافتن پاسخی مشخص برای یک پرسش است و ترجیح بر آنست که مستقیما بتوانند پاسخ سوال خود را دریافت کنند؛ تا آنکه سندی مرتبط با موضوع مورد پرسش خود ببینند که احتمالا پاسخ سوالشان را در خود دارد. خصوصا با گسترش استفاده از تلفن همراه و جستجو توسط آن، افراد عموما تمایلی به خواندن متن‌های بلندبالا که فقط بخشی از آن برایشان مفید خواهد بود، ندارند.

بر این اساس در این پروژه، می‌خواهیم به کمک یادگیری ماشین، سیستم پرسش و پاسخی به زبان فارسی طراحی کنیم که با داشتن یک متن و یک پرسش، بتواند پاسخ را از متن استخراج کند. این سیستم‌ها عموما با عنوان سیستم‌های درک مطلب[1] شناخته می‌شوند. سیستم طراحی‌شده، مبتنی بر مبدل[2] بوده و با استفاده از مدل ParsBERT، و به کمک مجموعه‌داده‌ی PersianQA برای این کار تنظیم دقیق[3] شده است.

[1] reading comprehension systems
[2] transformer
[3] fine-tune

فصل دوم

معرفی داده‌های پروژه

در این فصل به معرفی مجموعه‌داده‌های مرتبط با پرسش و پاسخ خواهیم پرداخت و سپس ملاک‌های ارزیابی سیستم‌هایی که با این مجموعه‌دادگان ساخته می‌شوند را معرفی خواهیم کرد.

۲-۱ معرفی مجموعه‌داده‌های مرتبط با پرسش و پاسخ

در این پروژه از مجموعه‌داده‌ی PersianQA[1] استفاده کرده‌ایم. این مجموعه‌داده، از مجموعه‌ی SQuAD[2] الهام گرفته است. در این بخش به معرفی این دو مجموعه می‌پردازیم.

۲-۱-۱ SQuAD

SQuAD یک مجموعه‌داده‌ی درک مطلب به زبان انگلیسی است. پاراگراف‌های این مجموعه‌داده، برگرفته از مقالات

[1] Persian Question Answering
[2] Stanford Question Answering Dataset

ویکی‌پدیا بوده و پرسش‌ها و پاسخ‌های آن، جمع‌سپاری[1]شده‌اند. این مجموعه در دو نسخه ارائه شده است:

- SQuAD 1.1 شامل بیش از صد هزار پرسش است که از بیش از پانصد مقاله‌ی ویکی‌پدیا استخراج شده‌اند. تمام این سوالات را می‌توان با توجه به متن پاسخ داد. پاسخ به هر سوال، گستره‌ای[2]از متن را شامل می‌شود.[۱۳]
- SQuAD 2.0 علاوه بر صد هزار پرسش نسخه‌ی پیشین، شامل بالغ بر پنجاه هزار سوال بدون جواب نیز هست. در واقع پاسخ به این سوالات، با استفاده از متن، ممکن نیست. از این سوالات، با عنوان سوالات بدون جواب یاد می‌کنیم.[۱۲]

نکته‌ی مهم در رابطه با این مجموعه‌داده آنست که اگر پرسشی قابل پاسخ‌دهی باشد، پاسخ آن، قطعا گستره‌ای از متن است. این بدان معناست که سیستم‌هایی که از این مجموعه‌داده استفاده می‌کنند، متن پاسخ را تولید نمی‌کنند؛ بلکه آن را از مقاله‌ای که در اختیار دارند استخراج می‌سازند؛ یعنی بازه‌ای به‌هم‌پیوسته از پاراگراف را به عنوان پاسخ، انتخاب می‌کنند. این کار شبیه زمانیست که بخشی از یک متن را برجسته[3]می‌کنیم.

هر نمونه از SQuAD از سه بخش اصلی پرسش، متن و پاسخ تشکیل شده است. در زیر، نمونه‌ای از یک پرسش با پاسخ از این مجموعه آورده شده است:

anwer	context	question	title
{'answer_start': [727], 'text': ['animal electricity']}	William Champion's brother, John, patented a process in 1758 for calcining zinc sulfide into an oxide usable in the retort process. Prior to this, only calamine could be used to produce zinc. In 1798, Johann Christian Ruberg improved on the smelting process by building the first horizontal retort smelter. Jean-Jacques Daniel Dony built a different kind of horizontal zinc smelter in Belgium, which processed even more zinc. Italian doctor Luigi Galvani discovered in 1780 that connecting the spinal cord of a freshly dissected frog to an iron rail attached by a brass hook caused the frog's leg to twitch. He incorrectly thought he had discovered an ability of nerves and muscles to create electricity and called the effect "animal electricity". The galvanic cell and the process	What did Galvani name the effect he created of causing the frogs legs to twitch?	Zinc

[1] crowdsourced
[2] span
[3] highlight

	of galvanization were both named for Luigi Galvani and these discoveries paved the way for electrical batteries, galvanization and cathodic protection.		

همانطور که در نمونه‌ی بالا می‌بینیم، هر پرسش قابل پاسخی، علاوه بر متن جواب، شاخص[1]شروع جواب از متن را نیز در بردارد که برای مثال آورده‌شده، این عدد برابر با ۷۲۷ است. این شاخص نشان می‌دهد پاسخ پرسش ما از چندمین حرف مقاله شروع می‌شود. واضح است که شاخص پایان جواب از مجموع شاخص شروع جواب و طول جواب به دست می‌آید.

نمونه‌ای از یک پرسش بدون جواب از این مجموعه را در زیر می‌بینیم:

anwer	context	question	title
text': [], '} {[] :'answer_start	In addition to Old Persian and Avestan, which are the only directly attested Old Iranian languages, all Middle Iranian languages must have had a predecessor "Old Iranian" form of that language, and thus can all be said to have had an (at least hypothetical) "Old" form. Such hypothetical Old Iranian languages include Carduchi (the hypothetical predecessor to Kurdish) and Old Parthian. Additionally, the existence of unattested languages can sometimes be inferred from the impact they had on neighbouring languages. Such transfer is known to have occurred for Old Persian, which has (what is called) a "Median" substrate in some of its vocabulary. Also, foreign references to languages can also provide a hint to the existence of otherwise unattested languages, for example through toponyms/ethnonyms or in the recording of vocabulary, as Herodotus did for what he called ""Scythian	What cannot be inferred from the impact on a neighboring ?language	Iranian_languages

[1] index

همانطور که مشاهده می‌شود، برای پاسخ به این سوال که «چه چیزی را نمی توان از تأثیر بر زبان همسایه استنباط کرد؟» در متن داده‌شده، منبع کافی وجود ندارد و این سوال بدون جواب طبقه‌بندی می‌شود.

SQuAD علاوه بر مجموعه‌ی آموزش، شامل دو مجموعه‌ی دیگر مربوط به اعتبارسنجی[1] و آزمایش[2] نیز هست. مجموعه‌ی آموزش و اعتبارسنجی به شکل عمومی منتشر شده‌اند اما مجموعه‌ی آزمایش به صورت عمومی در دسترس نیست. در مجموعه‌ی آزمایش و اعتبارسنجی، هر پرسش، دارای سه پاسخ است که لزوما عین یکدیگر نیستند اما هر سه پاسخ درستی ارزیابی می‌شوند. دلیل این امر آنست که هر یک از این پرسش‌ها توسط شخص متفاوتی پاسخ داده شده‌اند. این امر با هدف مقاوم[3] کردن ارزیابی صورت گرفته است.

PersianQA ۲-۱-۲

PersainQA یک مجموعه‌داده‌ی درک مطلب به زبان فارسی است. هر نمونه از این مجموعه‌داده نیز مشابه SQuAD از سه جزء اساسی تشکیل شده است که شامل پرسش، متن و پاسخ می‌باشد. متن‌های این مجموعه‌داده، از روی مقالات ویکی‌پدیای فارسی و از دسته‌بندی‌های موضوعی مختلفی همچون تاریخی، جغرافیایی، مذهبی، علمی و ... استخراج شده‌اند. پرسش و پاسخ‌های مربوط به این مقالات، توسط افراد مختلفی که فارسی، زبان مادری آن‌هاست، صورت گرفته است. لازم به ذکر است بعضی از پرسش‌ها به زبان محاوره‌ای آورده شده‌اند. [۲]

این مجوعه‌داده شامل بیش از ۹۰۰۰ پرسش است که مانند نسخه‌ی دوم SQuAD، دارای سوالات بدون جواب نیز می‌باشد. علاوه بر این ۹۰۰۰ پرسش، حدود ۹۰۰ پرسش دیگر نیز به عنوان دادگان آزمایش وجود دارد. به طور متوسط، هر پاراگراف، هفت پرسش قابل پاسخ و سه پرسش غیرقابل‌پاسخ دارد. هر یک از پرسش‌های موجود در نمونه‌های آزمایش، دارای دو پاسخ هستند که همچون SQuAD لزوما برابر یکدیگر نیستند.

در زیر نمونه‌ای از یک پرسش دارای جواب از این مجموعه آورده شده است:

پاسخ	پرسش	متن	موضوع
{'text': ['صابون، به تعریف علم شیمی، نمک	صابون چیه؟	نخستین اشاره به ساختن یک ماده شبیه صابون مربوط به حدود ۲۸۰۰ سال پیش از میلاد در بابل باستان است. برخی منابع مصری مربوط به حدود ۱۵۰۰ پیش از میلاد نیز به ساختن چنین ماده‌ای اشاره دارند. در	صابون

[1] validation
[2] test
[3] robust

یک اسید چرب است. '], 'answer_start': [585]}		حدود ۶۰۰ سال پیش از میلاد، ملوانان فینیقی‌ها، فن صابون‌سازی (یا صابون پزی) را به سواحل مدیترانه بردند. در قرن اول میلادی، بهترین صابون از چربی بز و خاکسترهای به‌دست‌آمده از سوزاندن چوب «درخت آلش» به‌دست می‌آمد. تا پایان قرن هجدهم، صابون را از چربی حیوانی و خاکستر چوب تهیه می‌کردند. در همان هنگام، معلوم شد که می‌توان به‌جای خاکستر چوب، از سود سوزآور، که قلیای حاصل از نمک معمولی است، استفاده کرد. صابون، به تعریف علم شیمی، نمک یک اسید چرب است. صابون عمدتاً برای شست‌وشو، حمام‌کردن و پاکیزگی استفاده می‌شود، ولی در ریسیدن پارچه هم از صابون استفاده می‌گردد و جزء مهمی از روان‌سازها است. صابون توالت سخت با بوی دلپذیر در خاورمیانه در دوره طلایی اسلامی تولید شد، زمانی که صابون سازی به یک صنعت تبدیل شد. دستور العمل‌های تهیه صابون توسط محمد بن زکریا رازی شرح داده شده‌است، که همچنین دستور تهیه گلیسیرین از روغن زیتون را داده است.	

مشابه آنچه در SQuAD دیدیم، هر پرسشِ دارای پاسخ، علاوه بر متن جواب، شاخص شروع جواب از متن را نیز در بردارد که برای مثال آورده‌شده این عدد برابر با ۵۸۵ است.

نمونه‌ای از یک پرسش بدون جواب از این مجموعه در زیر آورده شده است:

پاسخ	پرسش	متن	موضوع
{'text': [], 'answer_start': []}	اوج گیری تصوف چه اثری روی ابوبکر بن سعد داشت؟	ابومحمّد مُشرف‌الدین مُصلِح بن عبدالله بن مشرّف (۶۰۶ – ۶۹۰ هجری قمری) متخلص به سعدی، شاعر و نویسنده پارسی‌گوی ایرانی است. اهل ادب به او لقب «استادِ سخن»، «پادشاهِ سخن»، «شیخِ اجلّ» و حتی به‌طور مطلق، «استاد» داده‌اند. او در نظامیه بغداد، که مهم‌ترین مرکز علم و دانش جهان اسلام در آن زمان به حساب می‌آمد، تحصیل و پس از آن به‌عنوان خطیب به مناطق مختلفی از جمله شام و حجاز سفر کرد. سعدی سپس به زادگاه خود شیراز، برگشت و تا پایان عمر در آن‌جا اقامت گزید. آرامگاه سعدی در شیراز واقع شده‌است که به سعدیه معروف است. بیشتر عمر او مصادف با حکومت اتابکان فارس در شیراز	سعدی

		و هم‌زمان با حمله مغول به ایران و سقوط بسیاری از حکومت‌های وقت نظیر خوارزمشاهیان و عباسیان بود. البته سرزمین فارس، به واسطه تدابیر ابوبکر بن سعد، ششمین و معروف‌ترین اتابکان سَلغُری شیراز، از حمله مغول در امان ماند. همچنین قرن ششم و هفتم هجری مصادف با اوج‌گیری تصوف در ایران بود و تأثیر این جریان فکری و فرهنگی در آثار سعدی قابل ملاحظه است.	

۲-۲ ملاک‌های ارزیابی

عملکرد سیستمی که با استفاده از PersianQA خواهیم ساخت، با دو ملاک امتیاز تطابق دقیق[1] و امتیاز F1[2] ارزیابی می‌شود.

- تطابق دقیق یک معیار باینری (درست/نادرست) است که نشان می‌دهد آیا خروجی سیستم دقیقا با پاسخ درست مطابقت دارد یا خیر. به عنوان مثال، اگر سیستم ما، «انیشتین» را به عنوان پاسخ یک سوال برگرداند، در حالی که پاسخ صحیح «آلبرت انیشتین» باشد، در آن صورت سیستم نمره‌ی تطابق دقیق صفر برای آن مثال، دریافت خواهد کرد. تطابق دقیق یک معیار سخت‌گیرانه است.
- F1 یک ملاک کمتر سختگیرانه است. برای این معیار، دو پارامتر مطرح است: دقت[3] و یادآوری[4].

در مثال «انیشتین»، سیستم دارای دقت (precision) ٪۱۰۰ است؛ چرا که پاسخ آن یعنی انیشتین، زیرمجموعه‌ای از پاسخ درست یعنی آلبرت انیشتین می‌باشد. پارامتر یادآوری (recall) برای این مثال برابر با ٪۵۰ است؛ چرا که فقط شامل یکی از دو کلمه‌ایست که در پاسخ اصلی ذکر شده است. امتیاز F1 از طریق رابطه‌ی (۲-۱) به دست می‌آید:

$$F1 = 2 \times \frac{precision \times recall}{precision + recall} \qquad (۲-۱)$$

بنابراین در مثال بالا، این عدد برابر خواهد بود با:

$$F1 = 2 \times \frac{100 \times 50}{100 + 50} = 66.67\%$$

[1] Exact Match (EM)
[2] F1 score
[3] precision
[4] recall

هنگام ارزیابی روی مجموعه‌های اعتبارسنجی یا آزمایش، حداکثر امتیازات F1 وتطابق دقیق را در دو پاسخ ارائه‌شده توسط انسان برای آن سؤال در نظر می‌گیریم؛ به عنوان مثال، اگر سوالی که پاسخ آن آلبرت انیشتین بوده در مجموعه‌ی اعتبارسنجی حضور داشته باشد و پاسخ اول «آلبرت انیشتین» و پاسخ دوم «انیشتین» باشد، سیستم برای این مثال، تطابق دقیق ۱۰۰ درصد و امتیاز F1 ۱۰۰ درصد را دریافت می‌کند. در نهایت، نمرات تطابق دقیق و F1 در کل مجموعه داده‌ی ارزیابی برای به دست آوردن نمرات گزارش شده‌ی نهایی به طور میانگین محاسبه می‌شوند.

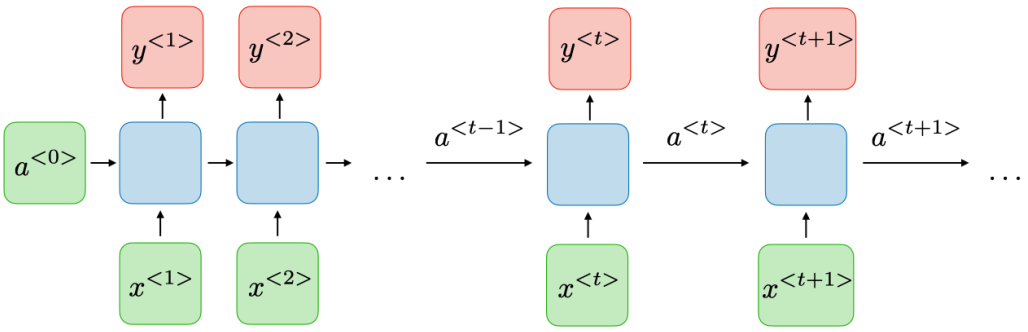
فصل سوم

روش‌های متداول پردازش زبان طبیعی

در این فصل، با روش‌های مرسومی که در پردازش زبان طبیعی به کار می‌روند، آشنا می‌شویم. نظر به اینکه پروژه‌ی پیاده‌سازی‌شده در این پروژه، مبتنی بر مبدل‌هاست، از ذکر جزئیات بیش از حد در این فصل پرهیز کرده‌ایم.

۳-۱ شبکه‌های عصبی بازگشتی

شبکه‌های عصبی بازگشتی که با نام RNN نیز شناخته می‌شوند، دسته‌ای از شبکه‌های عصبی هستند که در هر مرحله، علاوه بر ورودی جدید، از خروجی مرحله‌ی قبل نیز تاثیر می‌گیرند. در واقع خروجی تولید شده در هر گام از زمان، تنها متکی به ورودی جدید نیست و از خروجی گام‌های پیشین نیز متاثر است. شکل ۳-۱ معماری یک مدل ساده از شبکه‌ی عصبی بازگشتی را نشان می‌دهد.



شکل (۳-۱) معماری شبکه‌های عصبی بازگشتی [۱]

همانطور که می‌بینیم، در گام tام، ورودی جدید $x^{\langle t \rangle}$ به همراه $a^{\langle t-1 \rangle}$ که خروجی گام قبل است، وارد شبکه می‌شود و خروجی $y^{\langle t \rangle}$ را تولید می‌کند.

ذات ترتیبی این شبکه‌ها، آن‌ها را به شبکه‌های مناسبی برای داده‌های ترتیبی، مانند متن و گفتار تبدیل کرده است.

در شبکه‌ی آورده‌شده، به ازای هر ورودی، یک خروجی دریافت می‌کنیم. به این معماری، شبکه‌ی بازگشتی چند به چند[1] با تعداد ورودی و خروجی برابر گفته می‌شود. شبکه‌ی چند به چند یعنی ما از چند ورودی، به چند خروجی می‌رسیم. از این معماری می‌توان برای کارهایی هم‌چون تشخیص موجودیت‌های نام‌دار[2]استفاده کرد؛ اما باید گفت این مدل، تنها ساختاری نیست که می‌توان شبکه‌های عصبی بازگشتی را به کار برد. به فراخور هدفی که دنبال می‌کنیم می‌توانیم از دیگر معماری‌های بازگشتی نیز استفاده کنیم. از این معماری‌ها می‌توان به معماری یک به چند[3](مورد استفاده در تولید موسیقی[4])، معماری چند به یک[5](مورد استفاده در طبقه‌بندی احساس[6]) و معماری چند به چند با تعداد ورودی و خروجی نابرابر (مورد استفاده در ترجمه‌ی ماشینی) اشاره کرد.

از مزایای این شبکه‌ها می‌توان به موارد زیر اشاره کرد:

- بر خلاف شبکه‌های عصبی استاندارد، در شبکه‌های عصبی بازگشتی، پردازش ورودی با هر طولی ممکن است و نیاز نیست ورودی حتما دارای طول مشخصی باشد.

[1] many to many
[2] Named Entity Recognition (NER)
[3] one to many
[4] music generation
[5] many to one
[6] sentiment classification

- با توجه به تاثیری که این شبکه از گذشته می‌گیرد، اطلاعات در طول زمان منتقل می‌شوند.
- اندازه‌ی مدل با افزایش اندازه‌ی ورودی تغییر نمی‌کند و فقط به تعداد گام‌های زمانی بیشتری برای پردازش نیاز است.

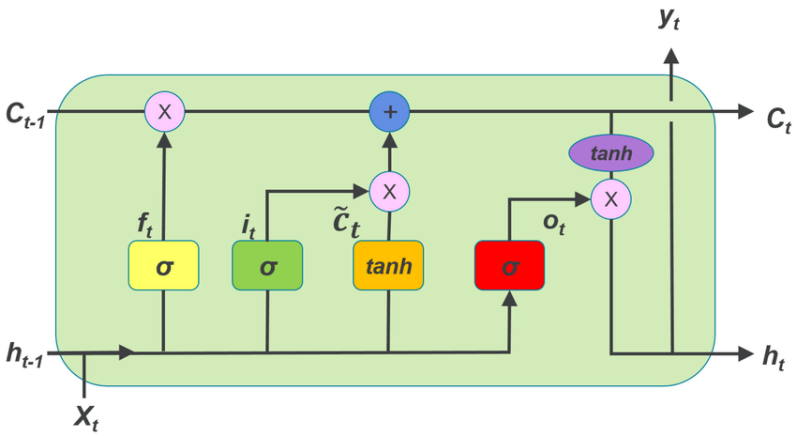
در کنار مزایایی که برای شبکه‌های عصبی بازگشتی، برشمردیم، تعدادی ایراد جدی نیز متوجه آن‌ها است:

- به این دلیل که پردازش ورودی باید به صورت متوالی انجام گیرد، این شبکه‌ها به کندی آموزش می‌بینند و پردازش موازی در آن‌ها ممکن نیست.
- این نوع شبکه‌ها با اینکه از گذشته تاثیر می‌گیرند، اما در طول زمان، وابستگی‌های بلندمدت را فراموش می‌کنند.
- این معماری، با مشکل محو و گرادیان روبه‌روست.

برای رفع دو مشکل محو گرادیان و فراموشی وابستگی‌ها، نوعی از شبکه‌های عصبی بازگشتی با نام Long Short Term Memory که به اختصار از آن‌ها با عنوان LSTM یاد می‌شود، معرفی شد که در بخش بعد به آن می‌پردازیم.

۲-۳ Long Short Term Memory

شبکه‌ی LSTM با هدف حل مشکل محو گرادیان و هم‌چنین حفظ طولانی‌تر وابستگی‌ها نسبت به RNN ساده، ایجاد شد.[۵] در این مدل، سه دروازه و یک سلول حافظه (c) معرفی شدند. این دروازه‌ها عبارتند از دروازه‌ی ورودی[1](i)، دروازه‌ی خروجی[2](o) و دروازه‌ی فراموشی[3](f). شکل ۳-۲ معماری داخلی این شبکه را به صورت شماتیک نشان می‌دهد.



شکل (۳-۲) ساختار یک بلوک LSTM [۸]

[1] input gate
[2] output gate
[3] forget gate

دروازه‌ی i مشخص می‌کند چه مقدار از اطلاعات فعلی (گام t) یعنی $\tilde{C}_t$ در حافظه ذخیره شده و وارد مرحله‌ی بعد شود. برای به دست آوردن $\tilde{C}_t$ باید خروجی گام قبلی یعنی h_{t-1} و ورودی جدید یعنی x_t در کنار هم قرار بگیرند. سپس در ماتریس وزن W_C که در طول فرآیند آموزش مقادیر آن به دست آمده است، ضرب شود. بعد از آن با بایاس[1] b_C که آن هم در طول یادگیری، به دست آمده است جمع گردد و نهایتا از یک تابع فعالسازی tanh عبور کند. برای محاسبه‌ی دروازه‌ی i در گام فعلی، به شکل مشابه، باید خروجی گام قبلی یعنی h_{t-1} و ورودی جدید یعنی x_t در کنار هم قرار گرفته و در ماتریس وزن W_i که همراه با مدل آموزش دیده است، ضرب شود. بعد با بایاس b_i که آن هم در طول یادگیری، به دست آمده است جمع شده و در آخر از یک تابع فعالسازی sigmoid عبور کند. صورت ریاضی این دو عملیات به ترتیب در فرمول‌های (۳-۱) و (۳-۲) آمده است.

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \quad (۳-۱)$$

$$\tilde{C}_t = tanh(W_C\,[h_{t-1}, x_t] + b_C) \quad (۳-۲)$$

دروازه‌ی f تعیین می‌کند چه میزان از اطلاعات گذشته از سلول حافظه پاک شود. برای به دست آوردن خروجی این دروازه در گام t، باید خروجی گام قبلی یعنی h_{t-1} و ورودی جدید یعنی x_t الحاق شوند و پس از آن در ماتریس وزن W_f که همراه با مدل آموزش دیده است، ضرب شود. بعد با بایاس b_f که آن هم در طول یادگیری، به دست آمده جمع گردد و از یک تابع فعالسازی tanh عبور کند. صورت ریاضی این عملیات در فرمول (۳-۳) آمده است.

$$f_t = \sigma(w_f[[h_{t-1}, x_t] + b_f) \quad (۳-۳)$$

نهایتا C_t با رابطه‌ی (۳-۴) محاسبه می‌شود. دقت داریم که در این فرمول علامت * به معنای ضرب عنصر به عنصر[2] است.

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \quad (۳-۴)$$

دروازه‌ی o مشخص می‌کند چه اطلاعاتی از حافظه خوانده شده و به خروجی انتقال یابد. برای محاسبه‌ی این دروازه، عملیاتی مشابه دو دروازه‌ی قبلی صورت می‌گیرد. یعنی باید خروجی گام قبلی یعنی h_{t-1} و ورودی جدید یعنی x_t در کنار هم قرار بگیرند. سپس در ماتریس وزن W_o که آموزش دیده است، ضرب شود، با بایاس b_o که آن هم در طول

[1] bias
[2] element-wise

یادگیری، به دست آمده است جمع گردد و نهایتا از یک تابع فعالسازی sigmoid عبور کند. رابطه‌ی (۳-۵) بیانگر این عملیات است.

$$o_t = \sigma(w_o[[h_{t-1}, x_t] + b_o) \quad (۳-۵)$$

حال به سراغ محاسبه‌ی خروجی می‌رویم. پس از آنکه سلول حافظه در زمان t را از تابع فعالسازی tanh عبور دادیم، با ضرب عنصر به عنصر دروازه‌ی o در آن، خروجی در این گام یعنی h_t به دست می‌آید. فرمول به دست آوردن خروجی در رابطه‌ی (۳-۶) آمده است.

$$h_t = o_t * \tanh(C_t) \quad (۳-۶)$$

شبکه‌های LSTM گرچه مشکل محو گرادیان را تا حد خوبی حل می‌کنند و اطلاعات گذشته را مدت زمان بیشتری در حافظه‌ی خود نگه می‌دارند، اما همچنان مشکلاتی هست که توسط این شبکه‌ها نیز حل نمی‌شود. مثلا با به کار بردن این معماری نیز نمی‌توان پردازش داده‌ها را به صورت موازی پیش برد و پردازش حتما باید به صورت ترتیبی انجام شود. بعلاوه دیدیم که معماری LSTM نسبت به RNN، پیچیده‌تر است. این پیچیدگی بدان معناست که تعداد پارامترهای بیشتری نسبت به RNN دارد و آموزش این نوع از مدل‌ها، حتی از شبکه‌های RNN نیز کندتر انجام می‌شود. هم‌چنین گرچه مدت‌زمانی که LSTM می‌تواند اطلاعات گذشته را حفظ کند نسبت به RNN پیشرفت داشته است، با اینهمه نمی‌توان گفت LSTM از پس حفظ وابستگی‌های بلندمدت نیز به خوبی برمی‌آید.

چالش‌هایی که برشمردیم، باعث شدند تا مدل‌های دیگری پدید آیند که مبتنی بر مبدل‌ها هستند. با این مدل‌ها در فصل آینده آشنا خواهیم شد.

فصل چهارم

روش‌های مبتنی بر مبدل

چندین سال است که از مفهوم یادگیری انتقالی در حوزه‌ی بینایی ماشین استفاده می‌شود. یادگیری انتقالی به این معناست که شبکه‌ای عصبی را روی دادگانی حجیم، به صورت بدون نظارت برای انجام کار مشخصی آموزش می‌دهیم و سپس از این مدل به‌عنوان لایه‌های پایه‌ای برای تنظیم دقیق روی داده‌های نظارت‌شده‌ی خود، استفاده می‌کنیم. مبدل، در واقع تلاشی برای پیاده‌سازی این مفهوم در زمینه‌ی پردازش زبان طبیعی است. در این فصل ابتدا به توضیح این مدل پرداخته و پس از آن دو مدل مبتنی بر مبدل شامل BERT و ParsBERT را معرفی خواهیم کرد.

۴-۱ مبدل

مدل مبدل، با تکیه بر مکانیسم توجه[1] پردازش موازی داده‌های ترتیبی را مانند متن، فراهم آورده است. این موضوع افزایش سرعت چشمگیری در فرآیند آموزش این مدل‌ها را باعث شده است. غیر از این، امکان استفاده از مفهوم یادگیری انتقالی نیز با استفاده از مدل‌های مبتنی بر مبدل، میسر گردیده است. [۱۴] در ادامه به توضیح این مدل می‌پردازیم.

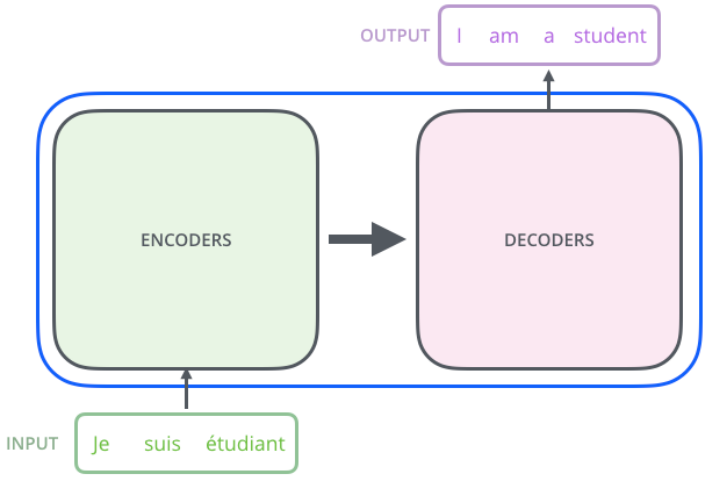
[1] attention mechanism

۴-۱-۱ معماری مبدل

در این بخش با معماری مدل بیشتر آشنا می‌شویم.

۴-۱-۱-۱ معماری رمزگذار-رمزگشا

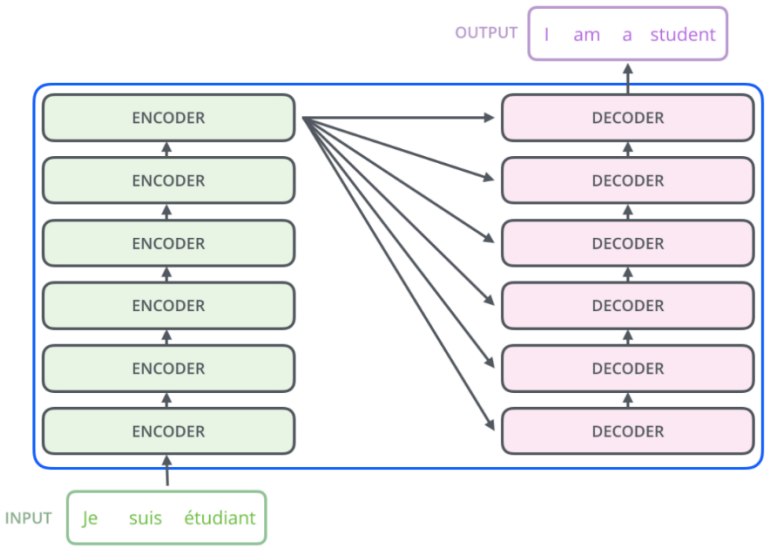
نسخه‌ی اولیه‌ی این مدل بر روی کار ترجمه‌ی ماشینی انجام گرفته است. لذا در این بخش، با در نظر داشتن این کار، به تشریح معماری مدل می‌پردازیم. یک مدل مناسب برای ترجمه‌ی ماشینی، معماری رمزگذار[1]- رمزگشا[2]است. این معماری برای کار ترجمه‌ی ماشینی در تصویر (۴-۱) آمده است. در این مثال، زبان مبدا، فرانسه، زبان مقصد، انگلیسی و جمله‌ی مورد ترجمه "Je suis étudiante" به معنای «من دانشجو هستم» در نظر گرفته شده است.



شکل (۴-۱) معماری رمزگذار-رمزگشا در ترجمه‌ی ماشینی [۶]

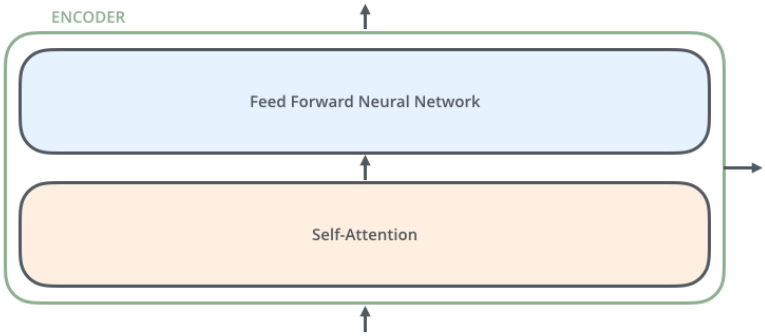
بخش رمزگذار، از شش لایه رمزگذار که روی هم پشته[3]شده‌اند، تشکیل شده است. در مورد بخش رمزگشا هم همین شرایط برقرار است. در هر بخش، خروجی لایه‌ی قبلی، ورودی لایه‌ی بعدیست. اتصال بین رمزگذارها و رمزگشاها به شکل کلی، در تصویر (۴-۲) آمده است.

[1] encoder
[2] decoder
[3] stack



شکل (۴-۲) اتصال دو بخش رمزگذار و رمزگشا در مبدل [۶]

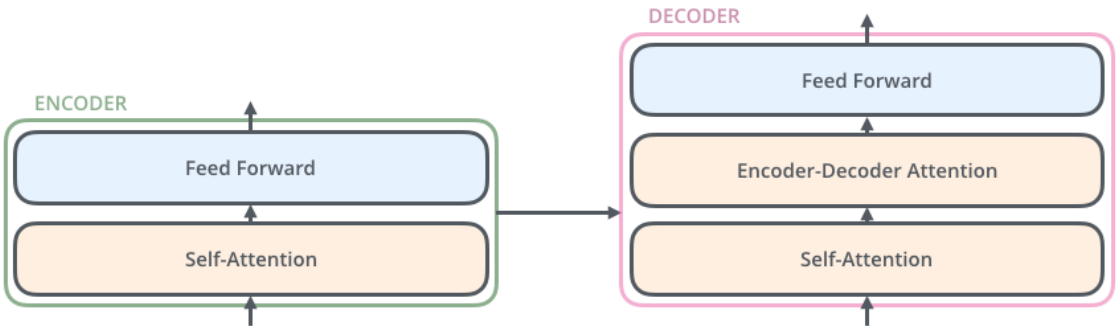
هر رمزگذار از دو زیرلایه‌ی خودتوجهی[1] و شبکه‌ی عصبی پیش‌خور[2] تشکیل شده است. همچنین از یک اتصال باقی‌مانده[3] در اطراف هر یک از زیرلایه‌ها استفاده می‌شود و یک لایه نرمالسازی پس از آن قرار می‌گیرد. شکل (۴-۳) دو زیرلایه‌ی اصلی را بدون اتصال باقی‌مانده و لایه‌ی نرمالسازی، برای هر رمزگذار نشان می‌دهد.



شکل (۴-۳) زیرلایه‌های اصلی رمزگذار [۶]

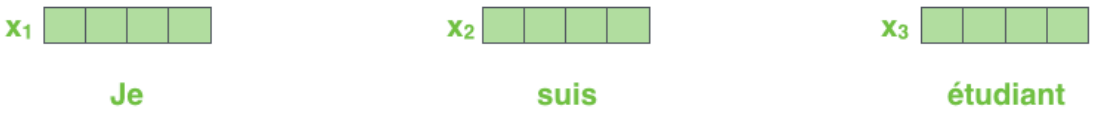
[1] self attention
[2] Feed Forward Neural Network (FFNN)
[3] residual connection

هر رمزگشا، علاوه بر زیرلایه‌های ذکر شده برای رمزگذار، یک زیرلایه‌ی دیگر نیز به نام توجه رمزگذار-رمزگشا[1] دارد. این لایه مکانیسم توجه چندسر[2] را روی خروجی رمزگذار اعمال می‌کند. مشابه رمزگذار، در بخش رمزگشا نیز یک اتصال باقی‌مانده در اطراف هر زیرلایه وجود دارد و به دنبال آن یک لایه نرمالسازی انجام می‌شود. شکل (۴-۴) دو بخش رمزگشا و رمزگذار را با زیرلایه‌های اصلی آن‌ها نشان می‌دهد.



شکل (۴-۴) زیرلایه‌های اصلی رمزگذار و رمزگشا [۶]

گفتیم که ورودی هر یک از رمزگذارها، خروجی رمزگذار قبلی است. به این ترتیب کلمات ورودی پس از آنکه توسط یک الگوریتم تعبیه‌سازی[3] به بردارهایی نظیر شدند، وارد رمزگذار اول می‌شوند. هر یک از کلمات ورودی به شکل برداری با اندازه‌ی ۵۱۲ نمایش داده می‌شوند و تعداد این کلمات بسته به طول متن می‌تواند متفاوت باشد. به شکل معمول این تعداد را برابر با طول بزرگترین جمله‌ی ورودی در نظر می‌گیرند. در ادامه برای سادگی کار، اندازه‌ی هر بردار را روی شکل چهار فرض می‌کنیم. شکل (۴-۵) نشان‌دهنده‌ی همین موضوع است. x_i به بردار نظیر iامین کلمه اطلاق می‌شود.

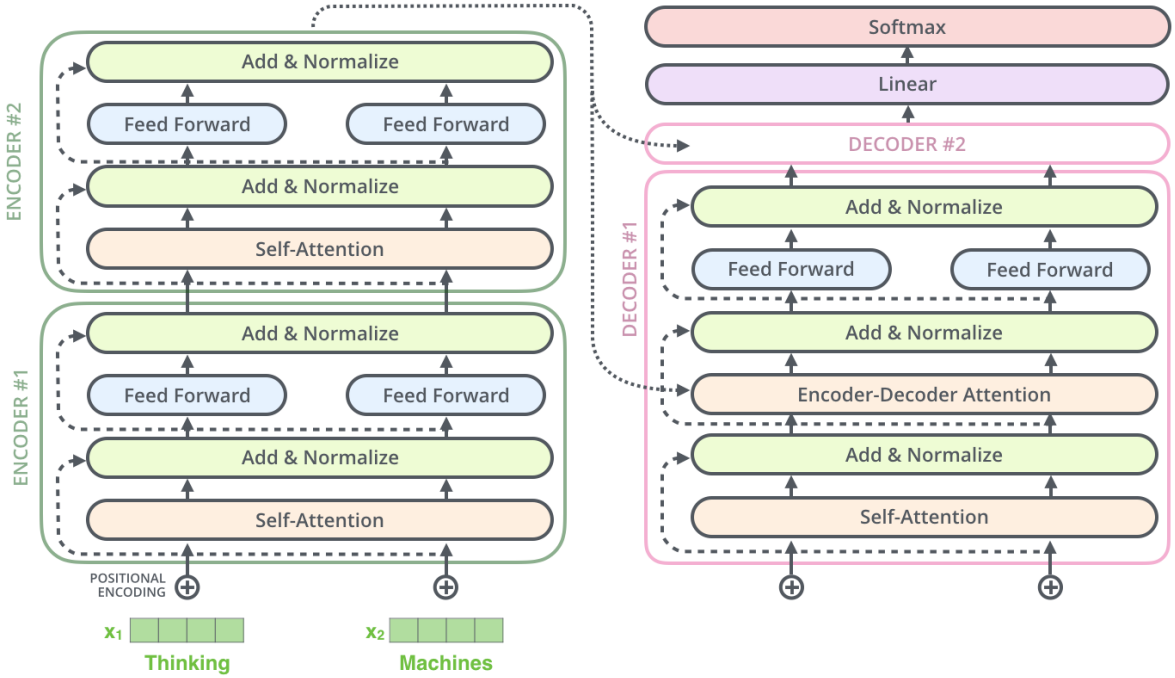


شکل (۴-۵) نحوه‌ی نمایش هر کلمه به شکل بردار با اندازه‌ی فرضی [۶]

خروجی آخرین رمزگشا در هر مرحله یک بردار خواهد بود اما آنچه ما می‌خواهیم به دست آوریم، یک کلمه در زبان مقصد است. لذا برای آنکه از این بردارها، کلمات را استخراج کنیم، ابتدا به یک لایه شبکه عصبی خطی نیاز داریم که بردار ما را به برداری با اندازه‌ی بزرگتر نظیر کند. برای مثال اگر تعداد لغاتی که از پیش آموخته شده است، هزار باشد،

[1] encoder-decoder attention
[2] multi-head attention
[3] embedding

اندازه‌ی برداری که پس از اعمال لایه‌ی خطی خواهیم داشت، باید هزار باشد. در این صورت هر درایه از بردار را می‌توان متناظر با یک کلمه از کل لغات در نظر گرفت. نهایتا با اعمال softmax این امتیازات، به احتمال تبدیل می‌شوند و بالاترین احتمال، به عنوان کلمه‌ی خروجی ما در نظر گرفته می‌شود. در نتیجه معماری کلی این مدل با فرض آنکه تعداد رمزگشاها و رمزگذارها، ۲ باشد، در شکل (۴-۶) آمده است:



شکل (۴-۶) معماری کلی مدل با فرض وجود دو رمزگذار و رمزگشا [۶]

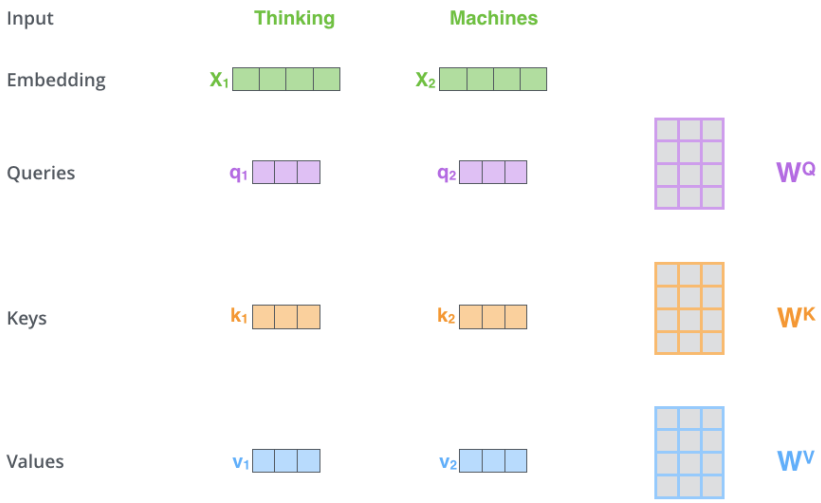
۴-۱-۱-۲ مکانیسم توجه

همانطور که گفته شد، با استفاده از مفهوم توجه، مدل مبدل به نتایج چشمگیری در زمینه‌ی پردازش زبان طبیعی دست یافت. در این بخش می‌خواهیم در مورد این مکانیسم و چگونگی اعمال آن در مبدل‌ها صحبت کنیم.

به شکل ساده مکانیسم توجه بیان می‌کند در تولید خروجی، به چه بخش‌هایی از ورودی باید بیشتر توجه کنیم. این مکانیسم باعث می‌شود مدل‌ها در کارهای مربوط به پردازش زبان طبیعی مانند سیستم‌های پرسش و پاسخ، بهتر عمل کنند. در مورد این مثال، توجه به این شکل معنی می‌شود که در پاسخ به یک پرسش از روی متن، روی چه بخش‌هایی از متن باید تمرکز بیشتری کرد. در مورد کار ترجمه‌ی ماشینی هم می‌توان گفت مکانیسم توجه به مدل کمک می‌کند تا بتواند تصمیم بگیرد در ترجمه‌ی هر کلمه، به چه کلماتی از ورودی باید بیشتر توجه کند.

در این مکانیسم ابتدا برای هر کلمه‌ی ورودی که با یک بردار نمایش داده می‌شود (x_i) ، سه بردار با نام‌های Query، Key و Value ساخته می‌شود. بردار Query مربوط به کلمه‌ی iام را با q_i، بردار Key مربوط به کلمه‌ی iام را با k_i و بردار Value مربوط به کلمه‌ی iام را با v_i نشان می‌دهیم. سایز این سه بردار معمولا از بردار اصلی کوچک‌تر است. در مدل اصلی اندازه‌ی بردار هر کلمه ۵۱۲ و اندازه‌ی مربوط به هر یک بردارهای ساخته‌شده برای آن، برابر با ۶۴ می‌باشد. این بردارها از ضرب بردار اصلی در سه ماتریس W^Q، W^K و W^V که به همراه مدل، آموزش دیده‌اند، به دست می‌آید.

با در نظر گرفتن عبارت "Thinking Machines" به عنوان دنباله‌ی ورودی، سه بردار ذکر شده برای هر یک از کلمات Thinking و Machines ساخته می‌شود. در شکل (۴-۷) ابعاد این بردارها و ماتریس‌های W^Q، W^K و W^V به صورت نمادین آمده است.



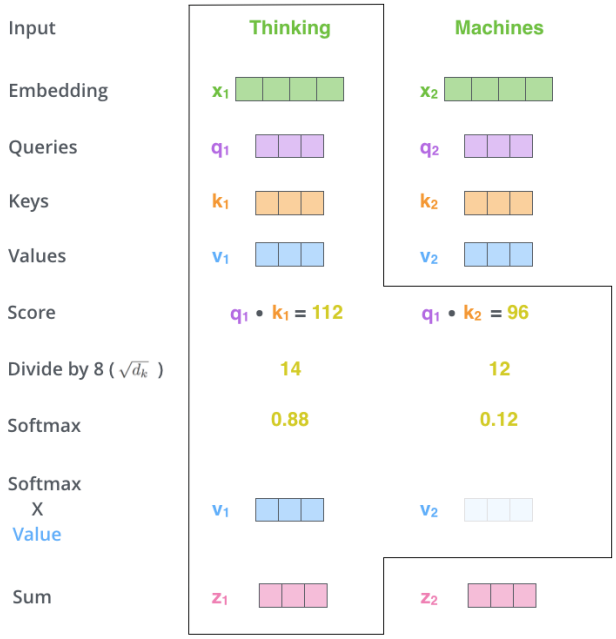
شکل (۴-۷) ابعاد نمادین ماتریس‌ها و بردارهای ساخته‌شده [۶]

در قدم بعدی امتیاز هر یک از کلمات در قیاس با کلمه‌ای که در حال پردازش آن هستیم محاسبه می‌شود تا بدانیم کدام بخش از ورودی شایسته‌ی توجه بیشتری است. در این گام اگر در حال پردازش اولین کلمه هستیم، ضرب داخلی بردار Query متناظر با آن را در تک‌تک بردارهای Key متناظر با تمام کلمات ورودی محاسبه می‌کنیم.

پس از آن هر یک از این امتیازات بر هشت تقسیم می‌شوند. عدد هشت از این رو انتخاب می‌شود که جذر اندازه‌ی بردارهای ساخته‌شده، یعنی ۶۴ می‌باشد. این کار برای پایدارسازی گرادیان‌ها صورت می‌گیرد. در نهایت با انجام عملیات softmax روی این اعداد، امتیازات نرمال شده و مجموع آن‌ها به یک می‌رسد. گام بعدی آنست که اعداد حاصل از عملیات softmax را در بردار Value نظیر هر کلمه‌ی ورودی ضرب کنیم. نهایتا بردارهای به دست‌آمده را با یکدیگر جمع می‌کنیم تا خروجی این زیرلایه برای کلمه‌ی اول به دست آید. با این کار در واقع ما یک جمع وزن‌دار ترتیب داده‌ایم که کلماتی که امتیاز

بیشتری داشته‌اند، تاثیر بیشتری نیز داشته باشند و به لایه‌ی بالاتر انتقال پیدا کنند. بردار ساخته شده در این مرحله برای کلمه‌ی iام را z_i می‌نامیم.

شکل (۴-۸) برای درک بهتر این عملیات و با اعداد فرضی آورده شده است.

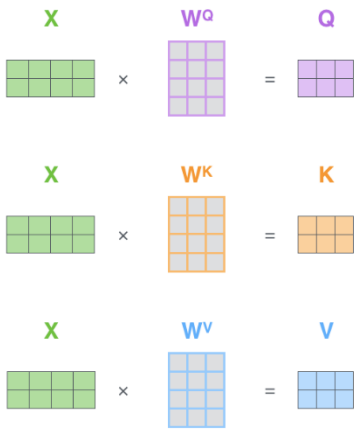


شکل (۴-۸) عملیات محاسبه‌ی بردار خروجی برای هر کلمه [۶]

این کار برای تمام کلمات ورودی به شکل گفته‌شده، انجام می‌گیرد. گرچه برای ساده شدن انتقال مطلب، در اینجا محاسبات را کلمه به کلمه در نظر گرفتیم، اما در واقع، این عملیات به صورت ماتریسی صورت می‌گیرد تا سرعت انجام محاسبات بیشتر شود. در این صورت ما به جای آنکه برای کلمه‌ی iام بردارهای x_i، q_i، k_i و v_i را داشته باشیم، برای تمام کلمات ورودی، ماتریس X، Q، K و V را خواهیم داشت. ابعاد این ماتریس‌ها با در نظر گرفتن فرضیات قبلی در شکل (۴-۹) آمده است. بنابراین نهایتا خروجی تمام کلمات در ماتریسی به نام Z ذخیره خواهد شد.

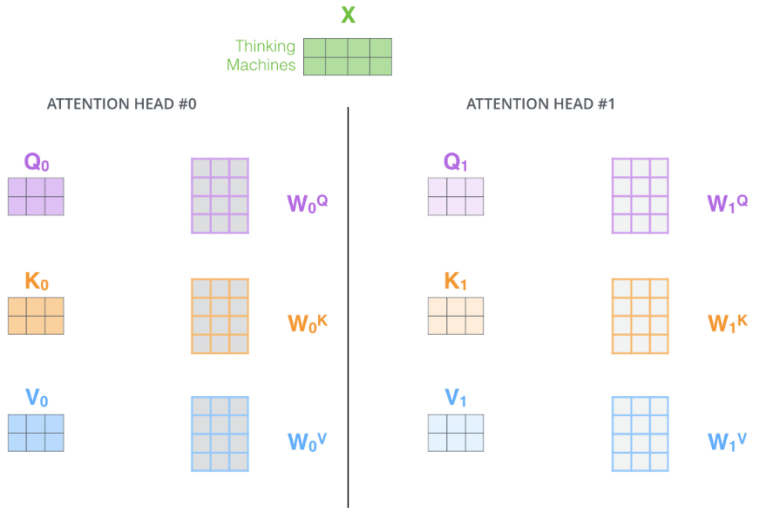
۴-۱-۱-۳ مکانیسم توجه چندسر

دیدیم که در محاسبه‌ی امتیازات برای کلمه‌ی در حال پردازش، خود آن کلمه را نیز دخیل دانستیم و امتیاز آن را نیز محاسبه کردیم. گاهی پیش می‌آید که امتیازات باقی کلمات، مغلوب کلمه‌ی اصلی واقع شوند و مکانیسم توجه به خوبی عمل نکند. برای رفع این مشکل، رویکردی با نام توجه چند سر در این مدل معرفی می‌شود.



شکل (۴-۹) فرم ماتریسی محاسبات [۶]

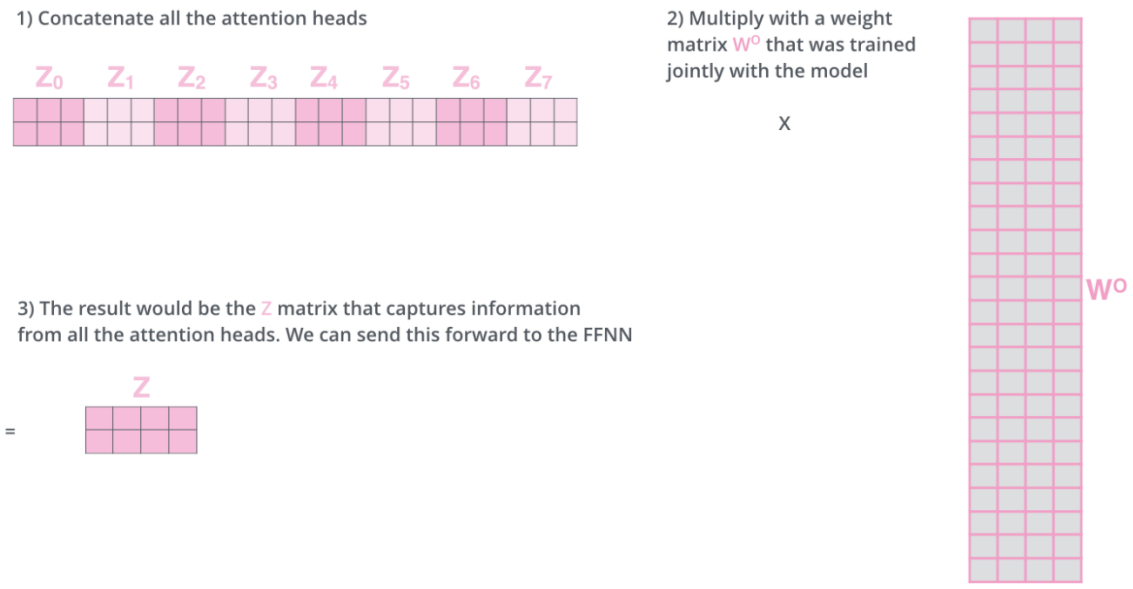
این مکانیسم به مدل این امکان را می‌دهد که بتواند روی بخش‌های مختلف ورودی تمرکز کند. در این مکانیسم در واقع ما چندین مجموعه Query، Key و Value خواهیم داشت که هم‌چون قبل، از ضرب ماتریس کلمات در ماتریس‌های آموزش‌دیده‌ی W^Q، W^K و W^V به دست می‌آیند. برای هر سر از توجه، این ماتریس‌ها متفاوتند. شکل (۴-۱۰) این تفاوت را نمایش می‌دهد. در مدل مبدل اصلی، تعداد این سرها، هشت است.



شکل (۴-۱۰) تفاوت ماتریس‌ها در سرهای مختلف توجه [۶]

در این صورت، با محاسبه‌ی امتیازات به گونه‌ای که گفته شد، هشت ماتریس Z متفاوت به دست می‌آید. این هشت ماتریس را در کنار هم الحاق می‌کنیم تا ماتریس بزرگتری به دست آید. ماتریس به دست آمده را در ماتریس وزن W^0 که همراه با مدل، آموزش دیده است، ضرب می‌کنیم تا خروجی نهایی زیرلایه‌ی خودتوجهی به دست آید.

این عملیات در شکل (۴-۱۱) آمده است.



شکل (۴-۱۱) عملیات تولید خروجی نهایی زیرلایه‌ی خودتوجهی [۶]

۴-۱-۱-۳ کاربرد مکانیسم توجه در بخش‌های مختلف مبدل

در زیرلایه‌ی توجه رمزگذار-رمزگشا در سمت رمزگشا، ماتریس Q از لایه‌ی قبلی رمزگشا تامین می‌شود. ماتریس‌های K و V از خروجی آخرین رمزگذار می‌آیند. این مورد به رمزگشا اجازه می‌دهد که نسبت به تمام کلمات دنباله‌ی ورودی آگاهی داشته باشد و بداند در هر گام، کدام‌یک از آن‌ها به توجه بیشتری نیاز دارند.

در زیرلایه‌ی خودتوجهی در سمت رمزگذار، هر سه ماتریس Q، K و V از خروجی لایه قبلی در رمزگذار می‌آیند. به این دلیل به آن، مکانیسم خودتوجهی گفته می‌شود. در واقع هر موقعیت[1] در رمزگذار می‌تواند به تمام موقعیت‌های لایه قبلی رمزگذار توجه کند.

در زیرلایه‌ی خودتوجهی در سمت رمزگشا، به طور مشابه، به هر موقعیت در رمزگشا اجازه داده می‌شود به تمام موقعیت‌های رمزگشا از ابتدا تا آن موقعیت، توجه کند.

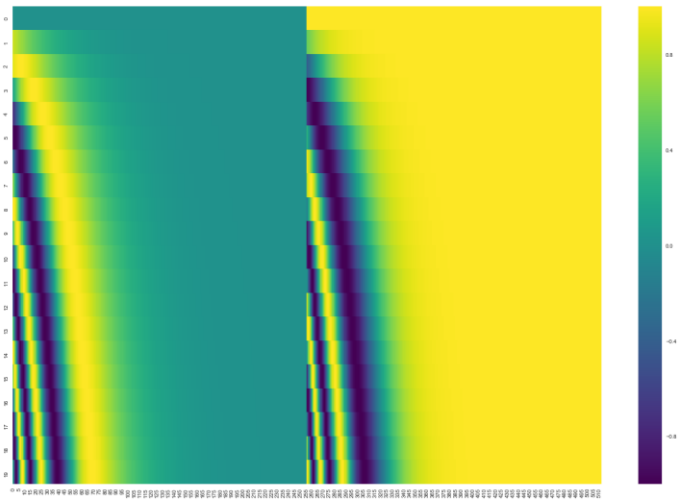
[1] position

۴-۱-۲ رمزگذاری موقعیتی

می‌دانیم که در داده‌های زبانی، ترتیب اهمیت بالایی دارد؛ بعنوان مثال دو عبارت «کار برای زندگی» و «زندگی برای کار»، با آنکه متشکل از تعدادی کلمه‌ی کاملا مشترک هستند، به دلیل تفاوت در موقعیت کلمات، معانی کاملا متفاوتی را منتقل می‌کنند.

تا اینجا در مدل مبدل، دیدیم که کلمات به شکل موازی پردازش می‌شوند. سوالی که پیش می‌آید اینست که فاکتور تاثیرگذاری مثل ترتیب چگونه در این مدل‌ها لحاظ می‌گردد؟ باید گفت این مدل از رمزگذاری موقعیتی[1] استفاده می‌کند. به این معنا که جایگاه هر کلمه در مرحله‌ی تعبیه‌سازی به همراه خود کلمه در بردار نهایی آن واژه، تعبیه می‌شود و هر بردار علاوه بر محتوای کلمه، اطلاعاتی از جایگاه کلمه در جمله را نیز دارد. بدین ترتیب بردار دیگری که دارای الگویی خاص است و مربوط به جایگاه کلمات می‌شود، به بردار مربوط به کلمه اضافه می‌شود. این بردار از فرمولی که در مقاله‌ی اصلی آمده است، محاسبه می‌شود که به علت پیچیدگی آن، از آوردن این رابطه در گزارش خودداری کرده‌ایم.

شکل (۴-۱۲) مربوط به این الگو برای دنباله‌ای از بیست کلمه با سایز ۵۱۲ می‌باشد. همانطور که مشخص است، نیمه‌ی سمت راست تصویر و نیمه‌ی سمت چپ آن به شکل مشهودی، متفاوتند. این بدان دلیل است که یک نیمه از رابطه‌ی کسینوسی و نیمه‌ی دیگر از رابطه‌ی سینوسی به دست آمده و نهایتا کنار یکدیگر الحاق شده‌اند. برای مثال اگر بخواهیم بردار نهایی کلمه‌ی اول را برای ورود به رمزگذار اول به دست آوریم، آن را با برداری که ۲۵۶ عنصر اول آن صفر و ۲۵۶ عنصر بعدی آن یک است جمع می‌کنیم.



شکل (۴-۱۲) بردارهای رمزگذاری موقعیتی برای بیست کلمه با اندازه‌ی ۵۱۲ [۶]

[1] positional encoding

۴-۲ مدل BERT

مدل BERT[1] یک مدل زبانیست که در سال ۲۰۱۸ و توسط گوگل معرفی شد. در زمان انتشار این مدل، نتایج مربوط به یازده کار حوزه‌ی پردازش زبان طبیعی، از جمله پرسش و پاسخ با داده‌های SQuAD، بهبود قابل قبولی پیدا کردند. [۳]

این مدل یک مدل دوطرفه‌ی چندلایه از رمزگذارهای مبدل‌هاست که در بخش قبل با آن آشنا شدیم. پیش از انتشار مدل BERT، مدل‌هایی مانند ELMo[2] و OpenAI GPT کاربرد داشتند که هر یک، با مشکلاتی مواجه بوند.

از مدل زبانی ELMo می‌توان برای تعبیه‌سازی کلمات به روشی جدید استفاده کرد که بر خلاف الگوریتم‌های تعبیه‌سازی پیشین همچون GloVe [۹] یا word2vec [۷]، کلمه را با توجه به کاربرد آن در جمله تعبیه می‌کند. برای مثال برای کلمه‌ی stick چندین تعبیه‌سازی مختلف وجود دارد که به فراخور بافت متنی که این کلمه در آن ظاهر می‌شود، انتخاب می‌شود که کدام تعبیه‌سازی مناسب است. این مدل مبتنی بر LSTM ساخته شده است و ایراداتی که برای این نوع شبکه‌ها برشمردیم، متوجه این مدل نیز هست. علاوه بر این، با وجود آنکه LSTM به شکل دوطرفه در معماری آن به کار رفته است، اما به این دلیل که صرفا نتایج حاصل از مسیر رفت و برگشت را به یکدیگر الحاق می‌شوند، نمی‌توان گفت مدل از هر دو جهت، درک خوبی به دست می‌آورد.[۱۰]

مدل OpenAI GPT با اینکه مبتنی بر مبدل‌ها ساخته شده است و مشکلات شبکه‌های عصبی بازگشتی را ندارد، اما به دلیل آنکه تنها در یک جهت عمل می‌کند، نمی‌تواند از هر دو سمت کلمات، آگاهی کسب کند. این مدل از چند لایه رمزگشای مبدل‌ها که به صورت پشته‌شده قرار گرفته‌اند، تشکیل شده است.[۱۱]

مدل BERT، هیچ یک از مشکلات دو مدل یادشده را ندارد و همانطور که گفته شد، هم مبتنی بر مبدل‌هاست و هم از یک معماری دوطرفه برخوردار است. در ادامه با معماری این مدل آشنا می‌شویم.

۴-۲-۱ معماری BERT

این مدل در دو نسخه با اندازه‌های متفاوت ارائه شده است. برای هر یک از آن‌ها، تعداد لایه‌های رمزگذار، اندازه‌ی حالت پنهان، تعداد سرهای توجه و تعداد کل پارامترها در جدول زیر آمده است:

	تعداد لایه‌های رمزگذار	اندازه‌ی حالت پنهان	تعداد سرهای توجه	تعداد کل پارامترها
BERT BASE	12	768	12	110 M
BERT LARGE	24	1024	16	340 M

جدول (۴-۱) تعداد اجزای مربوط به معماری دو نسخه‌ی BERT

[1] Bidirectional Encoder Representations from Transformers
[2] Embeddings From Language Models

۴-۲-۲ نمایش ورودی در مدل BERT

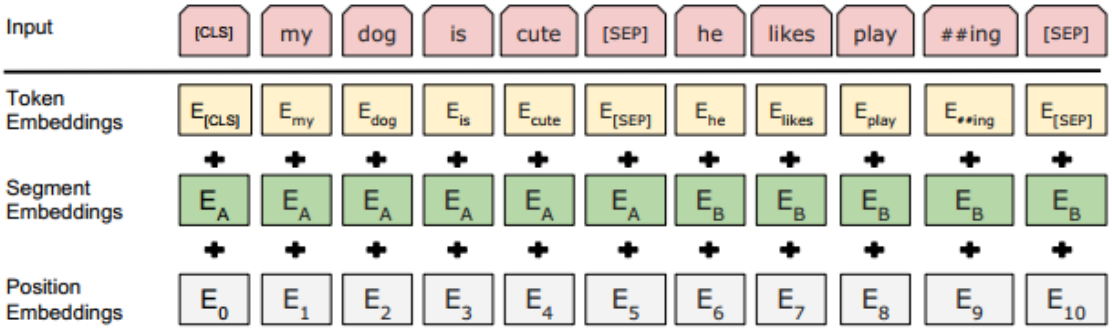
مدل BERT به گونه‌ای طراحی شده است که ورودی آن هم می‌تواند یک جمله‌ی واحد باشد و هم دو جمله‌ی مجزا از هم. باید توجه داشت منظور از جمله در اینجا، لزوما یک جمله‌ی زبانی نیست؛ دنباله‌ی به‌هم‌پیوسته‌ای از کلمات را در این مدل می‌توان یک جمله در نظر گرفت. ترجمه‌ی ماشینی می‌تواند مثالی از یک کار پایین‌دستی[1]باشد که ورودی آن تنها یک جمله است. در حالیکه سیستم‌های پرسش و پاسخ، برای استخراج جواب، به دو جمله‌ی پرسش و متن به عنوان ورودی نیاز دارند.

در مدل BERT، برای تعبیه‌سازی کلمات، از الگوریتم WordPiece استفاده می‌شود. این الگوریتم، تعداد سی هزار نشانه[2] را در خود دارد. در این نوع از تعبیه‌سازی، کلمات ممکن است خود به قسمت‌های کوچک‌تر تنظیم شوند. برای مثال اگر بخواهیم واژه‌ی snowing را تعبیه کنیم، توسط این الگوریتم کلمه به دو بخش snow و ##ing شکسته می‌شود و هر یک به شکل مستقل تعبیه می‌گردند.

برای آماده کردن دنباله‌ی ورودی برای ورود به مدل، ابتدای کار یک توکن خاص به نام [CLS] که مخفف Classification است به ابتدای رشته‌ی ورودی اضافه می‌شود. در انتهای جمله نیز توکن خاص [SEP] ظاهر می‌شود که مخفف Seperate است؛ این توکن برای جدا کردن جملات نامتوالی به کار می‌رود. اگر نمونه‌های ما تک‌جمله‌ای باشند، نشانه‌ی آخر آن‌ها و اگر دو جمله‌ای باشند، نشانه‌ی میان دو جمله و نشانه‌ی انتهایی آن‌ها، [SEP] خواهد بود. جملات، جز آنکه با نشانه‌ی از هم جدا می‌شوند، از طریق روش دیگری نیز این کار برایشان صورت می‌گیرد. به این صورت که بردار هر کلمه، با برداری که تعیین می‌کند این کلمه متعلق به جمله‌ی اول است یا جمله‌ی دوم، جمع می‌شود. به این کار تعبیه‌سازی بخش[3]گفته می‌شود.

در نتیجه برای هر کلمه‌ی ورودی، بردار مربوط به کلمه در WordPiece، بردار تعبیه‌سازی موقعیتی کلمه که در بخش مبدل به آن پرداختیم و بردار تعبیه‌سازی بخش که مربوط به جدا کردن دو جمله از یکدیگر است، با یکدیگر جمع می‌شوند تا بردار نهایی ورودی هر کلمه را بسازند. شکل (۴-۱۳) برای نشان دادن این عملیات آمده است.

[1] downstream task
[2] token
[3] segment embedding



شکل (۴-۱۳) عملیات آماده‌سازی نشانه‌های ورودی [۳]

۴-۲-۳ رویکردهای استفاده از مدل BERT

در سیستم‌هایی که از BERT استفاده می‌کنند، می‌توان از دو رویکرد استفاده کرد. این رویکردها عبارتند از: رویکرد مبتنی بر ویژگی[1] و رویکرد تنظیم دقیق.

در رویکرد مبتنی بر ویژگی، خروجی ما کلمات تعبیه‌شده هستند اما تفاوت این تعبیه‌سازی‌ها با الگوریتم‌های معروفی مانند GloVe یا word2vec در اینست که در BERT، کلمات با در نظر داشتن معانی آن‌ها در بافت جمله درون‌سازی می‌شوند و کلمات در بافت‌های متنی متفاوت می‌توانند نمایش متفاوتی داشته باشند. این تعبیه‌ها را می‌توان در یک مدل از پیش‌ساخته‌شده بعنوان ورودی استفاده کرد.

در رویکرد تنظیم دقیق، مدلی که با دادگان بدون برچسب ساخته‌ایم را برای تنظیم دقیق یک کار نظارت‌شده‌ی پایین‌دستی به کار می‌بریم. مزیت این استراتژی آن است که تعداد کمی از پارامترها را باید از ابتدا یاد گرفت و اکثرا از پارامترهای از پیش‌آموزش‌دیده استفاده می‌شود. این مورد باعث می‌شود بتوان در هزینه و انرژی صرفه‌جویی چشمگیری کرد. به طوریکه در طول ۳-۴ اپاک هم می‌توان مدل را برای کار مورد نظر آماده کرد.

با وجود اینکه از این مدل می‌توان با هر دو استراتژی یادشده استفاده کرد، اما باید گفت ارزش افزوده‌ای که BERT ایجاد کرده است، بیشتر در خدمت رویکرد تنظیم دقیق است. لذا در بخش بعد به طور دقیق‌تر به رویکرد تنظیم دقیق می‌پردازیم.

۴-۲-۴ استفاده از مدل BERT با رویکرد تنظیم دقیق

این رویکرد از دو مرحله‌ی اصلی ساخته شده است که عبارتند از: پیش‌آموزش[2] و تنظیم دقیق.

[1] feature-based approach
[2] pre-training

در طول مرحله‌ی پیش‌آموزش، مدل بر روی داده‌های بدون برچسب، آموزش داده می‌شود. برای تنظیم دقیق، ابتدا مدل BERT با پارامترهای از پیش آموزش‌داده‌شده مقداردهی اولیه می‌شود و همه‌ی پارامترها با استفاده از داده‌های برچسب‌گذاری شده از وظایف پایین‌دستی تنظیم می‌شوند. هر کار پایین‌دستی گرچه با پارامترهای یکسان مقداردهی اولیه می‌شود، اما دارای مدل‌های تنظیم‌شده‌ی جداگانه‌ای است.

ویژگی متمایزکننده‌ی BERT معماری یکپارچه آن در مورد کارهای مختلف است. چندانکه بین معماری مدل از پیش‌آموزش‌دیده و مدل نهایی برای یک کار پایین‌دستی، حداقل تفاوت وجود دارد.

۴-۲-۴-۱ مرحله‌ی پیش‌آموزش

همانطور که پیش‌تر گفته شد، مدل BERT یک مدل زبانی دوطرفه است. ضمنا می‌دانیم تعریف مدل زبانی به صورت ساده، پیش‌بینی کلمه‌ی بعدی در دنباله‌ای از داده‌ها، در نظر گرفته می‌شود. با در نظر داشتن این دو نکته، سوالی که مطرح می‌شود آنست که اگر بخواهیم هر کلمه‌ای، از دادگان پیش و پس از خود استفاده کند و دوطرفه باشد، آن کلمه به شکل غیرمستقیم خود را خواهد دید و به شکلی بدیهی پیش‌بینی درستی خواهد داشت که مورد استفاده‌ی ما نیست. پس چطور می‌توان معماری را به شکل دوطرفه طراحی کرد؟

راه حلی که طراحان مدل BERT برای این مشکل ارائه داده‌اند آنست که مدلی طراحی کنند که درصد مشخصی از کلمات ورودی را به صورت تصادفی پنهان کند و سپس به حدس آن کلمات بپردازد. از این تکنیک با عنوان Masked Language Model یا به اختصار MLM یاد می‌شود و به این ترتیب آموزش دوطرفه ممکن می‌شود.

تعداد خوبی از کارهای پایین‌دستی مانند سیستم‌های پرسش و پاسخ و یا استنتاج زبان طبیعی، به درک رابطه‌ی میان دو جمله نیاز دارند اما به نظر می‌رسد MLM به تنهایی نمی‌تواند این درک را ممکن سازد. در نتیجه کار دیگری به نام Next Sentence Prediction یا پیش‌بینی جمله‌ی بعد که به اختصار به آن NSP گفته می‌شود، به عنوان دومین گام که در مرحله‌ی پیش‌آموزش انجام می‌شود، به میان آمده است تا این مشکل را حل کند.

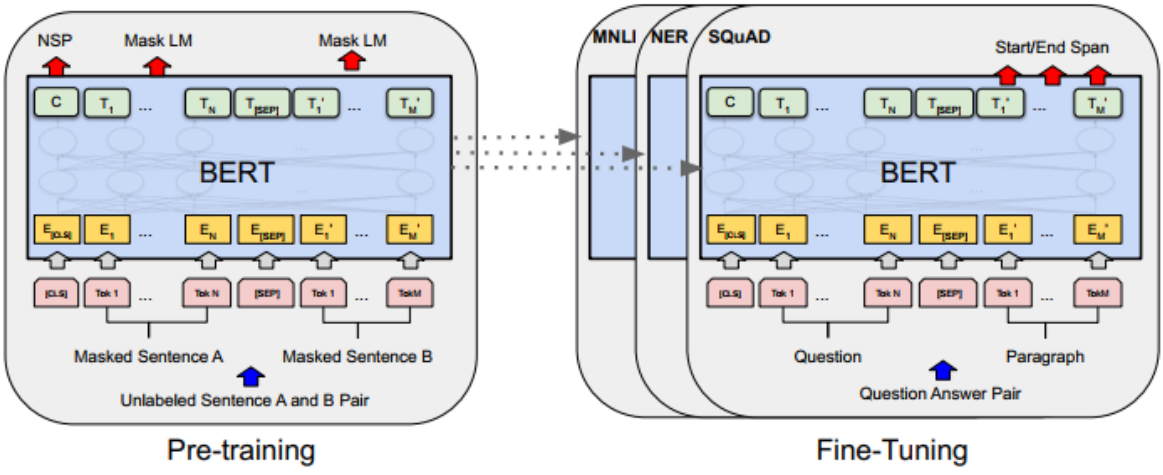
در این تکنیک، دو جمله به مدل داده می‌شود و مدل باید یاد بگیرد آیا جمله‌ی دوم، دنباله‌ی جمله‌ی اول بوده است یا خیر. به این صورت که در نیمی از دادگان پیش‌آموزش، دو جمله‌ای که مدل دریافت می‌کند دنباله‌ی هم هستند و در نیمی دیگر کاملا بی‌ربط بوده و دنباله‌ی یکدیگر نیستند.

گفتنیست داده‌های مرحله‌ی پیش‌آموزش از ویکی‌پدیای انگلیسی و همچنین مجموعه‌داده‌ی BookCorpus به دست آمده است.

۴-۲-۴-۲ مرحله‌ی تنظیم دقیق

چه داده‌ی ورودی ما یک جمله‌ای باشد و چه دو جمله‌ای، مکانیسم توجه این امکان را فراهم می‌کند تا مدل بتواند به درک خوبی از مفهوم زبان برسد. در صورتی که ورودی ما از دو جمله تشکیل شده باشد، با استفاده از همین مکانیسم می‌توان به ارتباط میان آن دو نیز توجه کرد.

برخلاف مرحله‌ی پیش‌آموزش که به زمان زیادی برای آموزش نیاز داشتیم، در این مرحله، تنها با چند اپاک می‌توان مدل را برای یک کار پایین‌دستی آماده کرد. شکل (۴-۱۴) این دو مرحله را در کنار هم نشان می‌دهد. همانطور که می‌بینیم در سمت چپ که مربوط به مرحله‌ی پیش‌آموزش است، دو کار مدل زبانی پنهان (MLM) و پیش‌بینی جمله‌ی بعدی (NSP) آمده‌اند. جملات A و B که درصدی از نشانه‌هایشان پنهان شده است، وارد فاز پیش‌آموزش می‌شوند و هر یک از کلماتشان، وارد بدنه‌ی اصلی مدل می‌گردند. پس از آنکه از ۱۲ یا ۲۴ لایه رمزگذار، بسته به اندازه‌ی مدل، عبور کردند، خروجی نهایی مربوط به هر کلمه حاصل می‌شود. دقت داریم که در ابتدای فرآیند، نشانه‌های خاص [CLS] و [SEP] اضافه شده‌اند. سپس تمام کلماتی که پنهان شده بوند حدس زده می‌شوند و خروجی C، اطلاعاتی درباره‌ی تمام ورودی را در اختیار ما می‌گذارد که با یک کار کلاس‌بندی دودویی، خروجی NSP از آن استخراج می‌شود که نشان می‌دهد آیا جمله‌ی B در ادامه‌ی جمله‌ی A بوده است یا خیر. در مرحله‌ی تنظیم دقیق، اگر کار را پرسش و پاسخ در نظر بگیریم، جمله‌ی اول پرسش و جمله‌ی دوم متن خواهد بود. به یاد داریم که منظور از جمله، جمله‌ی زبانی نیست که دارای تنها یک فعل باشد و هر متنی که قبل از نشانه‌ی [SEP] بیاید را می‌توان یک جمله در نظر گرفت. برای این کار پایین‌دستی پس از انجام فرآیند تنظیم دقیق، بازه‌ی شروع و پایان جواب به دست می‌آید.



شکل (۴-۱۴) مراحل پیش‌آموزش و تنظیم عمیق مدل BERT [۳]

۴-۳ مدل ParsBERT

گفتیم که مدل BERT یک مدل زبانی است که برای زبان انگلیسی طراحی گردیده است. اما اگر بخواهیم از این مدل برای کارهایی که با دادگان فارسی سروکار دارند، استفاده کنیم، چندان موثر نخواهد بود؛ چرا که به شکل طبیعی ساختار زبان فارسی و انگلیسی با یکدیگر تفاوت‌های اساسی دارند. در این بخش به معرفی مدل زبانی ParsBERT خواهیم پرداخت که با کمک مدل BERT انگلیسی، برای زبان فارسی طراحی شده است.[۴]

ParsBERT یک مدل زبانی مختص زبان فارسی است که در سال ۲۰۲۱ ارائه شد. معماری این مدل با الهام از معماری BERT انگلیسی و مشابه آن می‌باشد. به همین دلیل، در این فصل تنها به فرآیند آماده‌سازی داده‌های فارسی برای مدل، بسنده کرده‌ایم و باقی، شبیه به مدل BERT عمل می‌کند.

۴-۳-۱ جمع‌آوری داده

منابعی که داده‌های مرحله‌ی پیش‌آموزش مدل ParsBERT از آن‌ها استخراج شده است، در جدول (۴-۲) فهرست گردیده‌اند.

منبع	تعداد جمله
Persian Wikipedia	1878008
BigBang Page	3017
Chetor	166312
EliGasht	214328
DigiKala	177357
Ted Talks	46833
Books	25335
Miras-text	35758281

جدول (۴-۲) منابع دادگان مرحله‌ی پیش‌آموزش برای مدل ParsBERT

۴-۳-۲ پیش‌پردازش داده

پس از جمع‌آوری مجموعه‌داده‌ی پیش‌آموزش، چند مرحله پیش‌پردازش، از جمله پاکسازی[1]، جایگزینی و نرمال‌سازی، برای تبدیل مجموعه داده‌ها به یک قالب مناسب ضروری است. ابتدا باید تمام کاراکترهای ناخواسته و بی‌اهمیت موجود در مجموعه را پاک کنیم. این کار از طریق یک سری فرآیند انجام می شود که عبارتند از:

۱) اصلاح تمام کاراکترهای Unicode.
۲) حذف تمام اطلاعات بی اهمیت به عنوان مثال URL ها.
۳) حذف تمام برچسب های غیر ضروری مانند تگ های HTML، CSS و JS.

[1] sanitizing

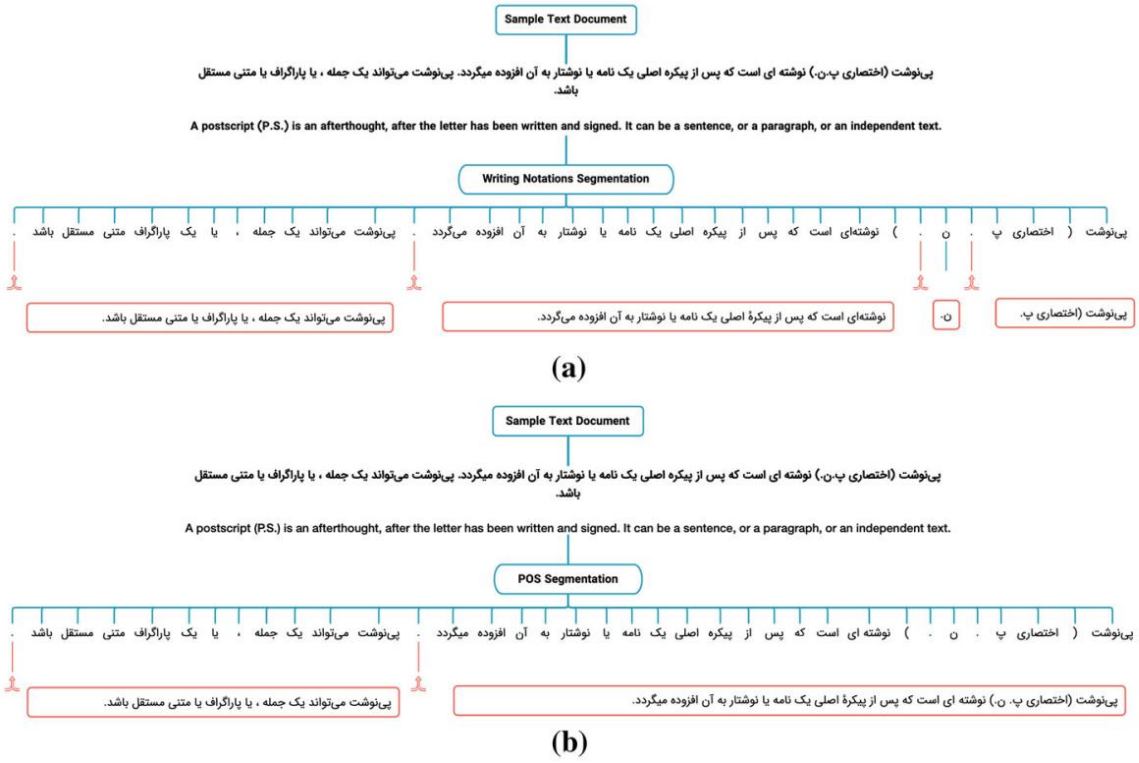
۴) نرماسازی متن فارسی با استفاده از کتابخانه‌ی HAZM که یک کتابخانه‌ی محبوب برای پیش پردازش متن فارسی است. نرمالسازی متن، اصلاح حروف فارسی، حذف فاصله‌های اضافی، تبدیل اعداد انگلیسی و عربی به فارسی، اصلاح علائم نگارشی و اصلاح نیم‌فاصله را شامل می‌شود.
۵) حذف کاراکترهای اسکی ناخواسته در زبان فارسی.
۶) جایگزینی کاراکترهای عربی با معادل آن‌ها در فارسی.
۷) حذف یا جایگزینی کاراکترهای Unicode بی‌معنی.

۴-۳-۳ تقسیم اسناد به جملات صحیح فارسی

پس از اینکه مجموعه پیش‌پردازش شد، باید به جملات واقعی هر سند، تبدیل شود. یک جمله‌ی صحیح در فارسی بر اساس نمادهای نقطه، علامت تعجب، علامت سوال و دونقطه تشخیص داده می شود. با این حال، تقسیم محتوا صرفا بر اساس این نمادها، باعث بروز مشکلاتی می‌شود. شکل (۴-۱۵)، نمونه‌ای از چنین مسئله‌ای را نشان داده شده است. مشاهده می شود که نتیجه، شامل جملات کوتاه بی‌معنی است. زیرا در فارسی اختصاراتی وجود دارد که با علامت نقطه (.) از هم جدا شده‌اند. به عنوان یک جایگزین، استفاده از اطلاعات اجزای کلام[1] می‌تواند راه حل مناسبی برای رسیدگی به این خطاها و تولید خروجی های دلخواه باشد. در زبان فارسی تمام جمله‌ها به فعل ختم می‌شوند. بنابراین، هر گاه قسمتی با نشانه‌ای که به عنوان «فعل» تگ شده است، پایان یابد و یکی از نمادها به دنبال آن قرار گیرد، به عنوان یک بخش معنی‌دار صحیح علامت‌گذاری شده و به عنوان یک جمله از بقیه قسمت‌ها جدا می‌شود.

پس از آنکه مجموعه‌داده‌ی پیش‌آموزش آماده شد، نوبت به دو مرحله‌ی پیش‌آموزش و تنظیم دقیق می‌رسد. این دو مرحله شبیه آنچه که در BERT دیدیم، اتفاق میفتد؛ بنابراین از تکرار آن‌ها پرهیز می‌کنیم. گفتنیست در مقاله‌ی اصلی، نتایج تنظیم دقیق این مدل برای سه کار تحلیل احساس، طبقه‌بندی متون و تشخیص موجودیت‌های نام‌دار آمده است. در فصل آینده در مورد تنظیم دقیق این مدل برای پرسش و پاسخ صحبت خواهیم کرد

[1] Part Of Speech (POS)



شکل (۴–۱۵) مثالی از تقسیم یک متن به جملات با استفاده از دو شیوه (a):توجه به نشانه‌ها (b): توجه به اجزای کلام [۴]

فصل پنجم
حل مسئله

در این فصل به تنظیم دقیق مدل ParsBERT برای کار پرسش و پاسخ فارسی روی دادگان PersianQA، پیاده‌سازی مدل با استفاده از کتابخانه‌های اکوسیستم Hugging Face و نتایج نهایی مدل خواهیم پرداخت.

۵-۱ تنظیم دقیق ParsBERT برای کار پرسش و پاسخ

مدل BERT برای کار پرسش و پاسخ روی مجموعه‌داده‌ی SQuAD در هر دو نسخه، تنظیم دقیق شده است. به روشی مشابه می‌توان ParsBERT را روی PersianQA تنظیم دقیق کرد. هدف مدل ساخته‌شده با استفاده از هر یک از این دو مدل، آنست که بتواند بازه‌ی پاسخ را از روی متن داده شده پیدا کند.

در طول روند تنظیم دقیق، دو بردار شروع و پایان جواب معرفی می‌شوند که همراه مدل آموزش می‌بینند. اندازه‌ی هر یک از این بردارها به اندازه‌ی لایه‌ی پنهان است که در مدل BERT پایه و ParsBERT، ۷۶۸ در نظر گرفته شده است. احتمال آنکه کلمه‌ی iام که در دنباله‌ی ورودی است، شاخص شروع پاسخ ما باشد، یعنی P_i با فرمول (۵-۱) محاسبه می‌شود:

$$P_i = \frac{e^{S.T_i}}{\sum_j e^{S.T_j}} \qquad (۱-۵)$$

در این فرمول، T_i برابر است با بردار لایه‌ی نهایی پنهان برای iامین نشانه که برای BERT پایه و $ParsBERT$ اندازه‌ای برابر با ۷۶۸ دارد. S نیز همان بردار شروع پاسخ است که چند خط قبل از آن صحبت شد. به این ترتیب برای هر نشانه‌ی موجود در متن یک احتمال برای اینکه آن کلمه شروع پاسخ سوال ما باشد، به دست می‌آید.

فرمولی مشابه با رابطه‌ی بالا برای احتمال آنکه توکن i، توکن پایان پاسخ ما باشد نیز ارائه می‌شود که یک شرط اضافه‌تر دارد. و آن اینست که شاخص پایان جواب باید از شاخص شروع جواب بیشتر باشد. به این معنی که پایان جواب نمی‌تواند پیش از شروع جواب در متن ظاهر شده باشد. رابطه‌ی (۵-۲) این فرمول را نشان می‌دهد.

$$P_i = \frac{e^{E.T_i}}{\sum_j e^{E.T_j}} \qquad (۵-۲)$$

در این فرمول، E همان بردار پایان جواب است که در فرآیند تنظیم دقیق معرفی و آموخته می‌شود.

نهایتا اگر i را شاخص شروع پاسخ و j را شاخص پایان پاسخ در نظر بگیریم، i و jی که $S \cdot T_i + E.T_j$ را بیشینه می‌کنند به شرطی که $j \geq i$، حدس نهایی ما خواهد بود.

با در نظر گرفتن سوالات بدون جواب، فرض می‌کنیم که شاخص شروع و پایان جواب این نوع سوالات، مربوط به نشانه‌ی [CLS] باشد؛ بدین ترتیب همانطورکه در بخش قبل برای تک‌تک نشانه‌ها احتمالی برای شروع و پایان پاسخ محاسبه کردیم، این بار باید توکن [CLS] را نیز در محاسباتمان دخیل کنیم. امتیاز مربوط به حالت بدون پاسخ (s_{null}) با فرمول (۵-۳) محاسبه می‌شود که در آن C مربوط می‌شود به بردار نهایی متناظر با [CLS].

$$s_{null} = S.C + E \cdot C \qquad (۵-۳)$$

برای محاسبه‌ی امتیاز برای سوالات دارای جواب، رابطه‌ی (۵-۴) را به دست آوردیم.

$$s_{i,j} = \max_{j \geq i} S \cdot T_i + E.T_j \qquad (۵-۴)$$

در این صورت زمانی نتیجه می‌گیریم که پرسش داده شده با توجه به متن قابل پاسخ نیست که رابطه‌ی (۵-۵) برقرار باشد.

$$s_{i,j} > s_{null} + \tau \qquad (۵-۵)$$

که در آن τ آستانه‌ایست که برای آن که سوالی را بدون جواب تلقی کنیم در نظر گرفته می‌شود.

۵-۲ پیاده‌سازی

برای پیاده‌سازی این پروژه، از اکوسیستم Hugging Face استفاده شده است. Hugging Face در اصل یک شرکت هوش مصنوعی است که اکوسیستمی فراهم آورده که با تجمیع و توسعه‌ی کاربردی‌ترین ابزارهای پردازش زبان طبیعی، حل مسائل مربوط به این حوزه را تسهیل بخشیده است. کتابخانه‌های این اکوسیستم، از جمله transformers، یکی از کتابخانه‌های محبوب زبان پایتون به حساب می‌رود.

۵-۲-۱ آماده‌سازی داده‌ها

برای طراحی سیستم پرسش و پاسخ، در ابتدا نیاز است داده‌ا را آماده کنیم. برای آماده‌سازی داده‌های PersianQA از کتابخانه‌ی datasets، استفاده کرده‌ایم. تابع load_dataset به ما کمک می‌کند مجموعه‌داده‌ی پردازش زبان طبیعی مورد نظر را بارگیری و از آن استفاده کنیم. این مجموعه‌داده می‌تواند در هر جایی ذخیره شده باشد. روی Hugging Face Hub، روی دیسک محلی یا بر روی Github. در این پروژه PersianQA را از Hugging Face Hub بارگیری نموده‌ایم.

با استفاده از تابع load_metric می‌توان ملاک‌های ارزیابی مورد نظر را بارگیری کرد. در صورتی که ملاک‌های ارزیابی توسط کتابخانه‌ی datasets به صورت مستقیم، پشتیبانی نشود نیز می‌توان ملاک‌های ارزیابی جدیدی تعریف و از آن‌ها استفاده کرد. معیارهایی که پیش‌تر معرفی کردیم، برای این پروژه توسط کتابخانه‌ی datasets پشتیبانی می‌شود.

پیش از آنکه داده‌ها را وارد مدل کنیم، نیاز است آن‌ها را به نشانه تبدیل کنیم. این عملیات توسط کلاس AutoTokenizer و با تابع from_pretrained قابل انجام است. با این تابع، می‌توانیم Tokenizer از پیش‌آموزش‌دیده‌ی مد نظر را بارگیری و از آن استفاده کنیم. مدل ParsBERT که می‌خواهیم از آن استفاده کنیم، با آدرس HooshvareLab/bert-fa-zwnj-base از کتابخانه‌ی transformers قابل دسترسی است. Tokenizer آن نیز بر مبنای الگوریتم WordPiece می‌باشد که پیش‌تر به آن پرداختیم.

۵-۲-۲ تنظیم دقیق مدل

برای ساخت مدلی که لایه‌ی خروجی آن، متناسب با یک کار پایین‌دستی خاص است، می‌توان از Automodelهایی که توسط کتابخانه‌ی transformers معرفی می‌شوند استفاده کرد. هر کار پایین‌دستی که بخواهیم انجام بدهیم، AutoModel مخصوص خود را دارد که لایه‌ی آخر آن، به فراخور کار مورد نظر طراحی شده است. در مورد کار پرسش و پاسخ، از AutoModelForQuestionAnswering استفاده کرده‌ایم. با استفاده از متد from_pretrained از این کلاس، می‌توانیم مدلی که می‌خواهیم از آن یک مدل پرسش و پاسخ بسازیم را انتخاب کنیم و وزن‌های آن را بارگیری نماییم. این کلاس، با جایگزین کردن لایه‌ی خروجی مدل از پیش‌آموزش‌دیده با لایه‌ی جدید متناسب با کار پرسش و پاسخ، مقدمات تنظیم

دقیق را برای ما فراهم می‌آورد. در این مرحله به تعداد ایپاکی که می‌خواهیم، مدل را آموزش می‌دهیم تا وزن‌های جدید، به دست آیند.

کار آموزش مدل از طریق کلاس Trainer از کتابخانه‌ی transformers، قابل انجام است. برای آنکه بتوانیم از این کلاس استفاده کنیم، ابتدا باید آرگومان‌های مربوط به آموزش را تنظیم کرده و به عنوان پارامتر ورودی به تابع train از این کلاس بدهیم. پس از آن با فراخوانی این تابع، فرآیند آموزش شروع می‌شود. در طول آموزش، پس از پایان هر ایپاک، می‌توان تابع هزینه‌ی مربوط به داده‌های آموزش و داده‌های اعتبارسنجی یا آزمایش را دید.

۵-۲-۳ ارزیابی مدل

خروجی مدل برای هر نمونه، مقدار تابع هزینه و بردار نهایی متناظر با احتمال آغاز و پایان جواب هر نشانه را بر می‌گرداند. برای انتخاب بهترین شاخص شروع و پایان، می‌توان نشانه‌ای که بهترین امتیاز را برای شاخص شروع بودن دارد، و نشانه‌ای که بهترین امتیاز را برای شاخص پایان بودن دارد، برداریم و به عنوان جواب در نظر بگیریم. اما همانطور که اشاره کردیم، باید این شرط را در نظر بگیریم که شاخص پایان کوچک‌تر از شاخص شروع نباشد؛ به همین دلیل ما به تعداد n بهترین کاندید جواب را انتخاب می‌کنیم. میزان n متغیر است. در این پروژه این عدد برابر ۲۰ در نظر گرفته شده است. از بهترین جواب شروع به بررسی می‌کنیم و بهترین جوابی که شرط را ارضا کرد را به عنوان جواب نهایی در نظر می‌گیریم. برای این کار نیازمند یک حلقه‌ی تودرتو هستیم.

پس از پایان کار مدل، می‌توان آن را در اکوسیستم Hugging Face قرار داد. این اکوسیستم هم‌چنین یک رابط کاربری نیز فراهم کرده است که به راحتی می‌توان به تست مدل پرداخت. تصویر زیر مربوط به مدلیست که تنظیم کرده‌ایم. این مدل، برای این سوال که «من به چه کاری علاقمندم؟» با توجه به متن «من به کار پردازش زبان‌های طبیعی علاقه دارم.»، جواب «کار پردازش زبان‌های طبیعی» را برگردانده است.

شکل (۵-۱) تست نمونه‌ای از پرسش و متن در مدل تنظیم‌شده با استفاده از Hugging Face API

۵-۳ نتایج ارزیابی

نتایج مربوط به تنظیم ابرپارامترهای نرخ یادگیری، اندازه‌ی دسته و تعداد ایپاک این سیستم در یازده حالت و دو ملاک ارزشیابی ما یعنی تطابق دقیق و امتیاز F1 در جدول (۵-۱) آمده است.

شماره حالت	تعداد ایپاک	نرخ یادگیری	اندازه‌ی دسته	امتیاز تطابق دقیق	امتیاز F1
#1	2	1e-5	1	39.56	53.36
#2	3	1e-5	1	40.64	55.47
#3	4	1e-5	1	41.29	56.12
#4	2	2e-5	1	42.79	56.65
#5	3	2e-5	1	43.11	58.66
#6	4	2e-5	1	41.61	57.13
#7	2	2e-5	2	42.90	57.86
#8	3	2e-5	2	42.90	58.70
#9	**4**	**2e-5**	**2**	**42.79**	**59.12**
#10	5	2e-5	2	42.15	58.03
#11	3	2e-5	3	40.64	55.30

جدول (۵-۱) نتایج حاصل از تنظیم دقیق ParsBERT با مجموعه‌داده‌ی PersianQA

نتایج تابع هزینه‌ی آموزش و اعتبارسنجی برای هر حالت، در ادامه آمده است.

نرخ یادگیری = 1e-5، اندازه‌ی دسته = 1، تعداد ایپاک = 2		
Epoch	Training Loss	Validation Loss
1	3.035800	2.606466
2	2.274800	2.577513

جدول (۵-۲) نتایج حالت اول آزمایش

نرخ یادگیری = 1e-5، اندازه‌ی دسته = 1، تعداد ایپاک = 3		
Epoch	Training Loss	Validation Loss
1	3.056700	2.791818
2	2.251500	2.666221
3	1.862000	3.047574

جدول (۵-۳) نتایج حالت دوم آزمایش

نرخ یادگیری = 1e-5، اندازه‌ی دسته = 1، تعداد ایپاک = 4		
Epoch	Training Loss	Validation Loss
1	3.051900	2.693343
2	2.259600	2.569704
3	1.841600	2.997958
4	1.182800	3.624714

جدول (۵-۴) نتایج حالت سوم آزمایش

نرخ یادگیری = 2e-5، اندازه‌ی دسته = 1، تعداد ایپاک = 2		
Epoch	Training Loss	Validation Loss
1	3.069700	2.635133
2	2.134000	2.600149

جدول (۵-۵) نتایج حالت چهارم آزمایش

نرخ یادگیری = 2e-5، اندازه‌ی دسته = 1، تعداد ایپاک = 3		
Epoch	Training Loss	Validation Loss
1	3.050500	2.811161
2	2.215500	2.683151
3	1.470300	3.428255

جدول (۵-۶) نتایج حالت پنجم آزمایش

نرخ یادگیری = 2e-5، اندازه‌ی دسته = 1، تعداد ایپاک = 4		
Epoch	Training Loss	Validation Loss
1	3.086400	2.852875
2	2.248100	2.842860
3	1.566200	3.428941
4	0.855400	4.465066

جدول (۵-۷) نتایج حالت ششم آزمایش

نرخ یادگیری = 2e-5، اندازه‌ی دسته = 2، تعداد ایپاک = 2		
Epoch	Training Loss	Validation Loss
1	2.806900	2.307975
2	1.892500	2.277155

جدول (۵-۸) نتایج حالت هفتم آزمایش

نرخ یادگیری = 2e-5، اندازه‌ی دسته = 2، تعداد ایپاک = 3		
Epoch	Training Loss	Validation Loss
1	2.785900	2.301585
2	1.907500	2.281873
3	1.200700	2.742789

جدول (۵-۹) نتایج حالت هشتم آزمایش

نرخ یادگیری = 2e-5، اندازه‌ی دسته = 2، تعداد ایپاک = 4		
Epoch	Training Loss	Validation Loss
1	2.769700	2.342861
2	1.930000	2.334564
3	1.241200	2.754138
4	0.777500	3.464884

جدول (۵-۱۰) نتایج حالت نهم آزمایش

نرخ یادگیری = 2e-5، اندازه‌ی دسته = 2، تعداد ایپاک = 5		
Epoch	Training Loss	Validation Loss
1	2.789600	2.297174
2	1.985400	2.319313
3	1.288500	2.708374
4	0.777500	3.464884
5	0.802800	3.723090

جدول (۵-۱۱) نتایج حالت دهم آزمایش

نرخ یادگیری = 2e-5، اندازه‌ی دسته = 3، تعداد ایپاک = 3		
Epoch	Training Loss	Validation Loss
1	2.758700	2.236697
2	1.926900	2.259194
3	1.247600	2.479626

جدول (۵-۱۲) نتایج حالت یازدهم آزمایش

همانطور که مشاهده می‌شود، بهترین نتیجه با ابرپارامترهای: (نرخ یادگیری = 2e-5، اندازه‌ی دسته = 2، تعداد ایپاک = 4) در حالت نهم، حاصل شد.

این آزمایش‌ها بر روی سیستمی با کارت گرافیک NVIDIA GEFORCE GTX 1060M انجام گرفتند. بر روی این سخت‌افزار، هر ایپاک در حدود سی دقیقه زمان برد. باید گفت افزایش اندازه‌ی دسته به بیش از سه، ممکن نبود و سیستم را با خطای حافظه‌ی GPU مواجه می‌کرد.

نتیجه گیری

در این پروژه سعی کردیم به مسئله‌ی درک مطلب بپردازیم. ابتدا با لزوم توجه به این قضیه آشنا شدیم. پس از معرفی مجموعه‌داده‌های مورد استفاده، گفتیم که چرا بهتر است به سراغ معماری‌های غیربازگشتی برویم. در ادامه به مبدل‌ها پرداختیم و بعد یکی از مهم‌ترین مدل‌های مبتنی بر مبدل را مطالعه کردیم. دیدیم که چطور یادگیری انتقالی در بحث پردازش زبان طبیعی به کار می‌رود. در آخر مدل مبتنی بر مبدل ParsBERT را برای کار مورد نظر خود در این پروژه، یعنی پرسش و پاسخ فارسی، تنظیم دقیق کردیم و نتایج حاصل از آن را بررسی نمودیم.

در ادامه‌ی کار این پروژه، می‌توان به جای مدل BERT به سراغ مدل XLNet رفت. XLNet مدلی شبیه به BERT است. با اینهمه این مدل در بیست کار مربوط به پردازش زبان طبیعی از جمله پرسش و پاسخ، بهتر از BERT عمل می‌کند. اگر چه به جهت سنگینی محاسبات، پیچیده‌تر از BERT بوده و زمان بیشتری برای آموزش نیاز دارد. XLNet از قابلیت پردازش متون با طول متغیر برخوردار است؛ هم‌چنین از روشی استفاده می‌کند که در آن، در مرحله‌ی پیش‌آموزش، نیاز به پنهان کردن کلمات ورودی نباشد. این موضوع عملکرد بهتر این مدل را سبب شده است. امید است با پیاده‌سازی سیستم پرسش و پاسخ مبتنی بر XLNet، نتایج ارزیابی، بهبود یابند.

مراجع

[1] Amidi, A., And Amidi, S. *Recurrent nueral networks cheatsheet* Available at: https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks

[2] Ayoubi, S., And Davoodeh, M. *PersianQA: a dataset for Persian Question Answering* Available at: https://github.com/sajjjadayobi/PersianQA

[3] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

[4] Farahani, M., Gharachorloo, M., Farahani, M., & Manthouri, M. (2021). Parsbert: Transformer-based model for persian language understanding. *Neural Processing Letters*, *53*(6), 3831-3847.

[5] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735-1780.

[6] Jay Alammar, (2018, June 27). The Illustrated Transformer [Blog post]. Retrieved from https://jalammar.github.io/illustrated-transformer/

[7] Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

[8] Olah, C. Understanding lstm networks. Available at https://colah.github. io/posts/2015-08-Understanding-LSTMs/

[9] Pennington, J., Socher, R., & Manning, C. D. (2014, October). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)* (pp. 1532-1543).

[10] Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (1802). Deep contextualized word representations. arXiv 2018. *arXiv preprint arXiv:1802.05365*, *12*.

[11] Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving language understanding with unsupervised learning.

[12] Rajpurkar, P., Jia, R., & Liang, P. (2018). Know what you don't know: Unanswerable questions for SQuAD. *arXiv preprint arXiv:1806.03822*.

[13] Rajpurkar, P., Zhang, J., Lopyrev, K., & Liang, P. (2016). SQuAD: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

[14] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*.