دانشگاه کاشان

دانشکده مهندسی برق و کامپیوتر

ارائه‌ی یک مدل برای درک زبان طبیعی با استفاده از شبکه‌های عصبی عمیق

پایان‌نامه کارشناسی ارشد مهندسی نرم‌افزار

مهرداد رفیعی‌پور

استاد راهنما

دکتر جواد سلیمی سرتختی

استاد مشاور

دکتر فرشته دهقانی

۱۴۰۱

تقـدیم به

مادرم به مهربانی فرشته

چکیده

با افزایش محبوبیت تلفن‌های هوشمند، استفاده از ابزارهای مبتنی بر سیستم گفت‌وگو نیز به طرز چشم‌گیری افزایش داشته است. درک زبان طبیعی، بخشی حیاتی از یک سیستم گفت‌وگو است؛ چراکه نقص در آن باعث ایجاد گلوگاه در چرخه‌ی عملکرد سیستم گفت‌وگو می‌شود. تشخیص هدف کاربر و پرکردن جای خالی، دو وظیفه‌ی اصلی درک زبان طبیعی هستند. شبکه‌های عصبی بازگشتی به طور گسترده برای بهبود این وظایف مورد بررسی قرار گرفته‌اند، اما دارای ضعف‌های شناخته شده‌ای مانند گرادیان محو شونده و زمان آموزش بالا هستند. به تازگی، ترنسفورمر برای رفع ایرادات مذکور معرفی شده است. از طرف دیگر، تعداد کمی از کارهای پیشین، خروجی مدل‌های زبانی را برای کار مورد نظر تعبیه می‌کنند. در این پایان‌نامه، مدل CTran معرفی می‌شود. CTran یک مدل رمزنگار-رمزگشای مبتنی بر شبکه‌ی عصبی کانولوشنی و ترنسفورمر است که برای دو چالش تشخیص هدف و پرکردن جای خالی طراحی شده است. در رمزنگار CTran، از بِرت به عنوان فراهم کننده‌ی تعبیه‌ی اولیه واژه ها استفاده شده است. سپس، یک لایه‌ی کانولوشنی با اندازه‌های هسته‌ی متفاوت استفاده شده، خروجی آن ترانهاده و سپس الحاق شده است. در بخش آخر رمزنگار، خروجی به یک پشته‌ی رمزنگار ترنسفورمر تغذیه شده تا تعبیه‌ی جمله‌ی ورودی تکمیل شود. در CTran به منظور تولید خروجی برای هر وظیفه، دو رمزگشای جداگانه معرفی شده، که هردو یک رمزنگار را به طور مشترک استفاده می‌کنند. برای رمزگشای تشخیص هدف، مکانیزم توجه به خود و به دنبال آن از یک لایه‌ی خطی تماماً متصل به کار گرفته شده است. برای رمزگشای پرکردن جای خالی، رمزگشای ترنسفورمر تراز شده معرفی گردیده است. برای تراز کردن رمزگشای ترنسفورمر، از ماتریس قطری استفاده شده است. ماتریس قطری، موقعیت‌های متناظر با برچسب‌های هدف در رمزنگار را، در دسترس قرار داده و سایر موقعیت‌ها را مخفی می‌کند. در انتهای کار، به منظور سنجش صحت عملکرد مدل پیشنهادی، مدل بر روی دو مجموعه داده‌ی ATIS و SNIPS آزمایش گردید. نتایج آزمایش‌ها نشان می‌دهد که مدل پیشنهادی در تشخیص جای خالی، بر روی هر دو مجموعه داده، عملکرد بهتری از مدل‌های پیشین دارد. علاوه بر این، عملکرد دو استراتژی مدل زبانی به عنوان رمزنگار و مدل زبانی به عنوان تعبیه واژه‌ها، سنجیده شد. نتایج نشان می‌دهد که استراتژی استفاده از مدل زبانی تنها به عنوان تعبیه‌ی واژه‌ها، عملکرد بهتری دارد.

کلمات کلیدی

۱-شبکه‌ی عصبی عمیق، ۲-پردازش زبان طبیعی، ۳-درک زبان طبیعی، ۴-تشخیص هدف در متن، ۵-پر کردن جای خالی در متن

فهرست مطالب

عنـــوان **صفحه**

فهرست تصاویر

عنـــوان **صفحه**

فهرست جداول

عنـــوان **صفحه**

فصل اول
پیش‌درآمد

۱-۱ مقدمه

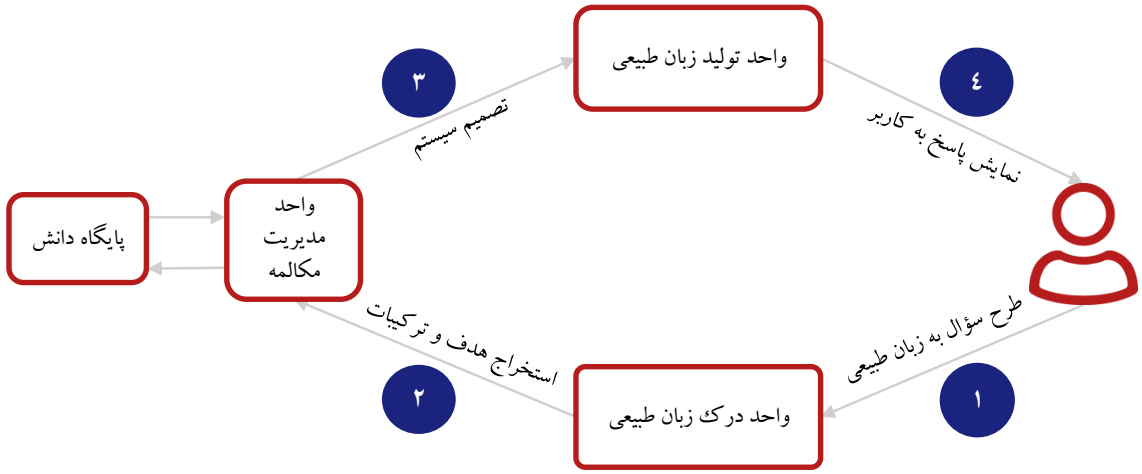
در فصل پیش رو، مقدمه‌ای بر مسئله‌ی درک زبان طبیعی بیان می‌شود. در ادامه، از کاربرد آن در سیستم‌های گوناگون و سیستم‌های امروزی سخن گفته شده و اهمیت آن تبیین گردیده است. در این راستا، ابتدا تعریف مسئله‌ی درک زبان طبیعی و دو وظیفه‌ای که پایه‌های مسئله را تشکیل داده‌اند گفته شده است و در ادامه، اهمیت مسئله و کاربرد آن را در دنیای واقعی تشریح شده است.

۱-۲ تعریف مسئله: طراحی یک مدل برای درک زبان طبیعی

سیستم گفت‌وگو[۱] یک برنامه‌ی رایانه‌ای است که برای تعامل با انسان از طریق متن یا صوت طراحی شده است. سیستم گفت‌وگو از طریق شبیه‌سازی مکالمه با انسان، امکان خودکارسازی امور را افزایش می‌دهد. در نتیجه، استفاده از آن باعث سادگی تعامل انسان با رایانه و افزایش دسترسی به اطلاعات و امکانات سیستم رایانه‌ای می‌شود. در یک سیستم گفت‌وگو، کاربر سؤالی را به زبان طبیعی مطرح می‌کند که پاسخ آن درون پایگاه دانش قرار دارد. وظیفه‌ی سیستم گفت‌وگو این است که هدف و اطلاعات مهم معنایی را از سؤال استخراج کرده و ضمن پیدا کردن پاسخ مناسب از پایگاه دانش، جوابی فراخور را به کاربر نمایش دهد. بدین منظور، مطابق شکل ۱-۱، در ابتدا سوال کاربر وارد واحد درک زبان طبیعی[۲] می‌شود. این واحد هدف و ترکیبات معنایی موجود در سوال را استخراج کرده و در اختیار واحد مدیریت مکالمه[۳] قرار می‌دهد. واحد مدیریت مکالمه، وظیفه‌ی نگهداری و ردیابی مکالمه‌ی کاربر و همچنین تعامل با پایگاه دانش را برعهده دارد. این واحد با توجه به سیاست‌های[۴] موجود، تصمیمی را اتخاذ کرده و آن را در اختیار واحد تولید زبان[۵] قرار می‌دهد، تا این واحد اقدام به تولید پاسخ کند. توانایی کاربر به سؤال کردن محدود نمی‌شود؛ بلکه می‌تواند درخواست انجام عمل نیز بدهد. در این صورت، به جای تولید پاسخ به زبان طبیعی، عمل مناسب توسط سیاست‌های موجود در پایگاه دانش مشخص شده و به واحد مدیریت مکالمه ابلاغ می‌شود.

همانطور که در شکل ۱-۱ مشهود است، درک زبان طبیعی نخستین بخش در چرخه‌ی عملکرد سیستم گفت‌وگو است. این امر بیانگر نقش حیاتی این سیستم است؛ به نحوی که نقص در عملکرد این بخش منجر به ایجاد گلوگاه[۶] برای سایر اجزاء سیستم می‌شود. درک زبان طبیعی شامل دو وظیفه‌ی "تشخیص هدف"[۷] و "پرکردن جای خالی"[۸] است. غایت در وظیفه‌ی تشخیص هدف، پیش‌بینی منظور کاربر از سوال مطرح شده است. همچنین، پرکردن جای خالی

[1] Dialogue System
[2] Natural Language Understanding
[3] Dialogue Management Unit
[4] Policy
[5] Language Generation Unit
[6] Bottleneck
[7] Intent-Detection
[8] Slot-Filling



شکل ۱-۱ - فرایند عملکرد سیستم‌های گفت‌وگو

به معنای استخراج اطلاعات معنایی[1] از درون جمله است. اطلاعات معنایی، واژه‌هایی از جمله هستند که دارای نام، کد، زمان و اطلاعاتی بوده که به تکمیل وظیفه کمک می‌کنند. پر کردن جای خالی را می‌توان به عنوان یک مسئله‌ی برچسب‌زنی توالی[2]، و تشخیص هدف را به عنوان یک مسئله‌ی کلاس‌بندی[3] تعریف کرد.

برای تعریف رسمی مسئله، تعداد واژه‌ها در زبان کاربر با V، تعداد برچسب‌های یکتا با T، تعداد اهداف یکتا در مجموعه داده با I و تعداد واژه‌های سؤال با n، نمایش داده می‌شود. در این صورت، کاربر سوال خود را در قالب $Q = \{q_1, q_2, q_3, \cdots, q_n\}, q_i \in \mathbb{W}^V$ مطرح می‌کند. واحد درک زبان طبیعی به ازاء هر Q، یک هدف $i \in \mathbb{W}^I$، و ترتیب S که به صورت $S = \{s_1, s_2, s_3, \cdots, s_n\}, s_i \in \mathbb{W}^T$ تعریف می‌شود را، تولید خواهد کرد. جدول ۱-۱ نمونه‌ای از سوال کاربر، برچسب مورد انتظار و هدف مورد نظر را از مجموعه داده‌ی ATIS نمایش می‌دهد. در این مثال، برچسب‌های O به معنای عدم وجود ترکیبات مهم معنایی در واژه است. ترکیباتی که حاوی اطلاعات مهم باشند، با برچسبی متناسب با معنای آن واژه، برچسب‌زنی می‌شوند. این برچسب‌ها از نظر معنایی هم‌راستا با هدف کاربر هستند.

سؤال کاربر	what is the abbrevation d10
برچسب‌های صحیح	O O O O B-aircraft_code
هدف کاربر	atis_abbreviation

جدول ۱-۱ - نمونه‌ی سؤال کاربر، برچسب‌های صحیح و هدف مورد نظر کاربر از مجموعه داده‌ی ATIS.

[1] Semantic Information
[2] Sequence Labeling
[3] Classification

به منظور انجام دو وظیفه‌ی یاد شده، لازم است که مدل بر روی مجموعه داده‌ی مشخصی آموزش داده شود. هدف اصلی این پایان‌نامه، طراحی مدلی است که بتواند حل مسئله‌ی مذکور را یاد بگیرد؛ یعنی در مواجه با سؤال دیده نشده، هدف صحیح و اطلاعات معنایی موجود در جمله را استخراج کرده و در اختیار سایر بخش‌های سیستم گفت‌وگو قرار دهد. طراحی یک مدل درک زبان طبیعی که در شرایط مختلف خوب عمل کند، یک چالش کلیدی است؛ چراکه زبان طبیعی پیچیده و غیرقابل پیش‌بینی است. این ویژگی زبان طبیعی باعث شده است که رایانه برای درک جمله و هدف کاربر با دشواری روبرو شود. همچنین مدل درک زبان باید همواره به‌روزرسانی و بازآموزی شود تا با تغییرات زبان و درخواست‌های جدید کاربران سازگار بماند. برای ساخت یک مدل درک زبان طبیعی، باید حجم زیادی داده جمع‌آوری و علامت‌گذاری شود. این علامت‌گذاری‌ها شامل هدف کاربر از نوشتن جمله و موجودیت‌های درون جمله می‌شوند.

۱-۳ اهمیت مسئله

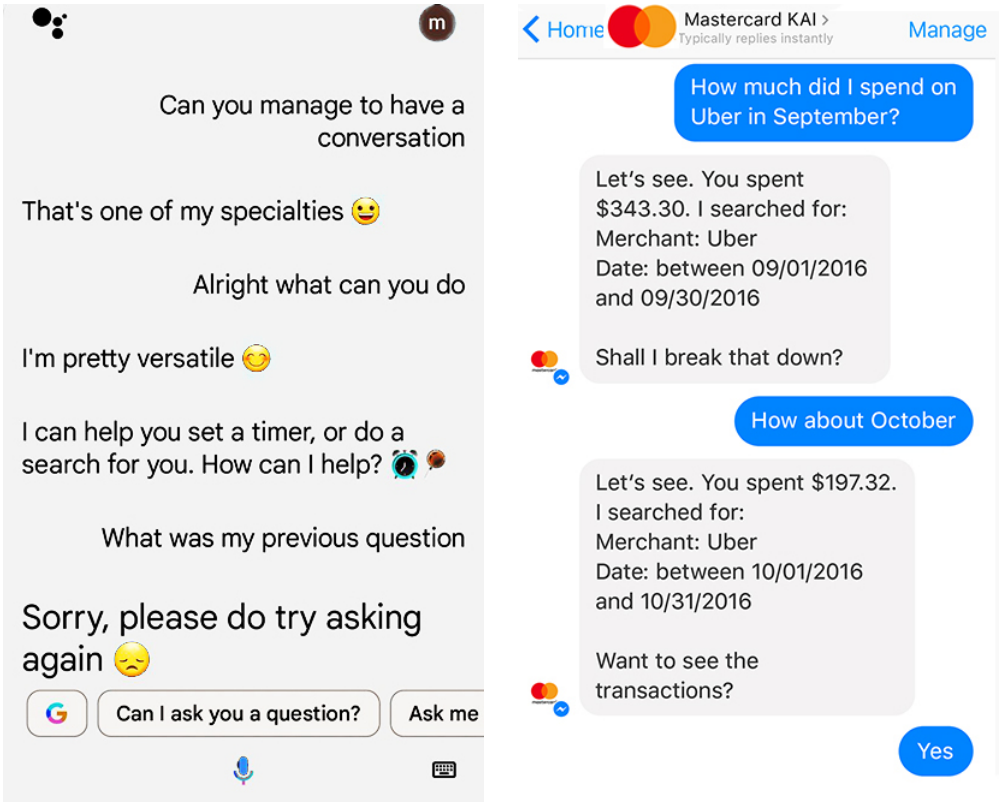
با رشد سریع تلفن‌های هوشمند، استفاده از ابزارهای مبتنی بر سیستم گفت‌وگو نیز به طرز چشم‌گیری افزایش داشته است. استفاده از سیستم گفت‌وگو می‌تواند با کاهش ترافیک خطوط ارتباطی شرکت‌ها، هزینه‌ی عملیاتی[1] آن‌ها کاهش دهد. از طرف دیگر سرعت پاسخگویی یک سیستم گفت‌وگو، باعث راحتی کار کاربر برای دریافت خدمت و افزایش رضایتمندی او می‌شود. در ادامه به معرفی برخی از ابزارهای دیجیتال که از سیستم گفت‌وگو بهره برده‌اند پرداخته می‌شود.

❖ **خدمات مشتری به صورت خودکار:** می‌توان از یک سیستم گفت‌وگو برای ارائه خدمات به صورت خودکار به مشتریان در وب‌سایت‌ها استفاده کرد. چنین سیستمی قادر است به سوالات متداول پاسخ‌های خودکار ارائه دهد، مشتریان را به بخش مربوطه هدایت کند و اطلاعاتی سودمند برای کسانی که کمک بیشتری می‌خواهند ارائه دهد.

❖ **دستیار شخصی دیجیتال:** دستیارهای شخصی دیجیتال، توانایی انجام کارهای روزمره مانند پخش موسیقی، تنظیم یادآور و کنترل ابزارهای هوشمند را دارند. بهره‌وری از این امکانات، منجر به افزایش کیفیت زندگی کاربران می‌شود. از نمونه‌های امروزی این دستیارها می‌توان به دستیار گوگل در شکل ۱-۲، الکسا و سیری اشاره کرد.

❖ **دستیار شخصی خرید:** این نوع از دستیارها در وبسایت‌های فروشگاهی به منظور فراهم کردن تجربه‌ی خرید بهتر برای کاربران مورد بهره‌وری قرار می‌گیرند. کاربر می‌تواند به جای ساعت‌ها گشتن در وبسایت برای یافتن محصول، با ارائه‌ی درخواست خود به دستیار خرید، سریع‌تر به محصول مورد نظرش برسد. چنین سیستمی قادر به ارائه پیشنهادهای ویژه به کاربر به منظور افزایش احتمال خرید او نیز هست. شکل ۱-۳ چنین سیستمی را در فروشگاه مایکروسافت و

[1] Operational Cost



(ب) دستیار شخصی گوگل (الف) دستیار دیجیتال مسترکارت

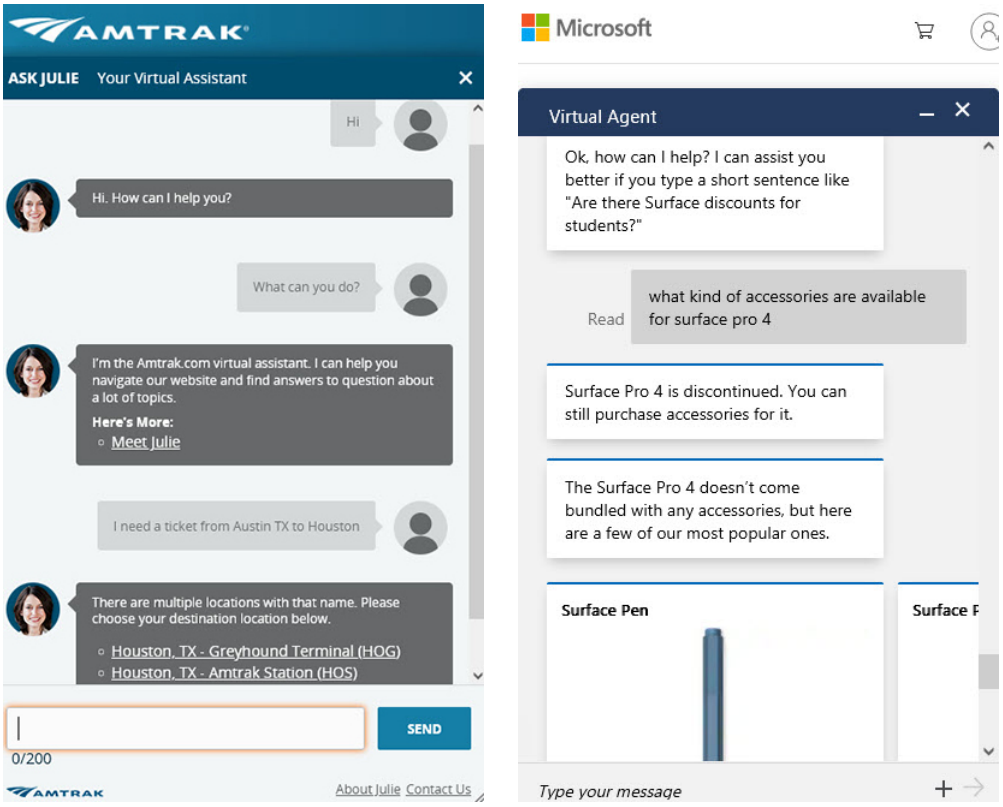
شکل ۱-۲ – دستیار دیجیتال استفاده شده در وبسایت‌ها برای بهبود تجربه‌ی کاربری مشتریان

شرکت راه‌آهن امترک نمایش می‌دهد.

❖ **تدریس خصوصی برخط:** این ابزار شامل یک سیستم گفت‌وگو برای تفسیر سوالات دانش‌آموزان و ارائه پاسخ‌های مناسب می‌باشد. قادر به ارائه راهنمایی و پشتیبانی شخصی به دانش آموزان است و می‌تواند به آن‌ها به منظور تسلط بر مطالعات خود کمک کند.

❖ **تعامل در رسانه‌های اجتماعی:** برای ارائه خدمات تعامل در شبکه‌های اجتماعی می‌توان از یک سیستم گفت‌وگو استفاده کرد. این سیستم قادر است سوالات مشتریان را در کانال‌های شبکه‌های اجتماعی تفسیر کرده و به شیوه‌ای مناسب به آن‌ها پاسخ دهد. همچنین، این سیستم می‌تواند واژه‌های کلیدی و عباراتی را که نشان‌دهنده احساسات مثبت یا منفی هستند، شناسایی کرده و به آن پاسخ دهد. از موارد پیاده‌سازی شده می‌توان به ربات شرکت تلوزیونی ام‌تی‌وی، و سامسونگ استرالیا نام برد [۱].

از طرف دیگر، مدلی که توانایی حل دو مسئله‌ی تشخیص هدف کاربر و استخراج روابط معنایی را بر داشته باشد، در بسیاری از وظایف دیگر پردازش زبان طبیعی نیز کاربرد دارد. همانطور که گفته شد، تشخیص هدف کاربر یک مسئله‌ی کلاس‌بندی است. از سایر وظایف کلاس‌بندی در زبان طبیعی که می‌توانند از معماری ارائه شده بهره ببرند،



(ب) پشتیبان دیجیتال شرکت راه‌آهن امترک (الف) دستیار دیجیتال فروشگاه مایکروسافت

شکل ۱-۳ - دستیار دیجیتال استفاده شده در وبسایت‌ها برای بهبود تجربه‌ی کاربری مشتریان

می‌توان به تحلیل احساسات[1]، تشخیص کلام نفرت‌افکن[2]، تشخیص اخبار جعلی[3] و تشخیص موضع[4] اشاره کرد. از سوی دیگر، مسئله‌ی پرکردن جای خالی یک مسئله‌ی برچسب‌زنی توالی است. سایر زمینه‌هایی که می‌توانند معماری مشترکی با این وظیفه داشته باشند شامل استخراج موجودیت، ابهام زدایی معنای واژه و برچسب زنی اجزاء سخن هستند. تشابه ساختار ورودی-خروجی و انتظارات مشابهی که از مدل زمینه‌های ذکرشده وجود دارد، باعث برجسته شدن فواید یک مدل قدرتمند در زمینه‌ی درک زبان طبیعی می‌شود.

۱-۴ اهداف پژوهش

به منظور آشنایی با محتوای پژوهش، باید با اهداف مورد نظر آن آشنا شد. در ادامه، اهدافی که این پایان‌نامه حول آن شکل گرفت معرفی می‌شوند.

اول) بسیاری از مدل‌های ارائه شده برای درک زبان طبیعی، همچنان از شبکه‌ی LSTM به عنوان رمزنگار یا رمزگشا در مدل خود استفاده می‌کنند. نخستین هدف این پژوهش، ارائه‌ی مدلی است که شبکه‌های عصبی بازگشتی را کاملا

[1] Sentiment Analysis
[2] Hate-Speech
[3] Fake-News Detection
[4] Stance Detection

کنار گذاشته و با ترنسفورمرها جابجا کند.

دوم) استفاده از ترنسفورمر، ضعف‌های ذاتی شبکه‌ی عصبی بازگشتی را پوشش می‌دهد. با این وجود شاید به نظر برسد که استفاده از شبکه‌ی عصبی کانولوشنی دیگر در رمزنگار ضروری نیست. هدف دوم پژوهش، بررسی عملکرد ترکیب شبکه‌ی عصبی بازگشتی، با ترنسفورمر در رمزنگار است.

سوم) تراز بودن مدل، موضوعی مهم در وظیفه‌ی برچسب زنی توالی است؛ اما تاکنون شیوه‌ای برای تراز کردن رمزگشای ترنسفورمر ارائه نشده است. هدف دوم پژوهش، ارائه کردن مدلی برای تراز کردن رمزگشای ترنسفورمر است.

چهارم) مدل زبانی بِرت انقلابی در یادگیری انتقالی در زبان طبیعی ایجاد کرد. پیش از برت، مدل زبانی اِلمو ارائه شده بود اما به اندازه‌ی برت مورد بررسی قرار نگرفت. علاوه بر این، تعداد کارهای اندکی در درک زبان طبیعی از المو استفاده کردند. در این کار قصد داریم عملکرد مدل زبانی المو را در شرایط یکسان، بر روی وظیفه‌ی درک زبان طبیعی بسنجیم.

پنجم) برخی از کارهای پیشین، مدل زبانی را به عنوان رمزنگار، و برخی دیگر به عنوان تعبیه‌ی کلمات استفاده کردند. یکی دیگر از اهداف این پژوهش، مقایسه‌ی این دو شیوه برای به کار گیری مدل زبانی است.

ششم) آخرین هدف در این پایان‌نامه، معرفی یک مدل جدید است که بر روی مجموعه داده‌ی SNIPS و یا ATIS، بیشترین دقت را داشته باشد.

۱-۵ ساختار پایان‌نامه

در ادامه‌ی این پایان نامه، ابتدا در فصل ۲، به مفاهیم پایه‌ی درک زبان طبیعی و شبکه‌ی عصبی پرداخته می‌شود. در ادامه‌ی فصل ۲، مطالبی برای آشنایی با شبکه‌های عصبی ارائه می‌شود. سپس انواع شبکه‌های عصبی که در زمینه‌ی درک زبان طبیعی و وظایف مرتبط، از آن‌ها استفاده شده، معرفی می‌شوند. علت توصیف گسترده‌ی این شبکه‌ها، ماهیت کار این پایان‌نامه است؛ چراکه در این پایان‌نامه، یکی از کارهایی که صورت گرفته، تلاش برای پوشش ضعف شبکه‌های مورد استفاده است. در فصل ۳، یک دسته‌بندی کلی از کارهای پیشین در نظر گرفته شده، و در قالب این دسته‌بندی، پیشین مورد بررسی قرار گرفته‌اند. در فصل ۴، مدل پیشنهادی این پایان‌نامه، یعنی CTran، و اجزای سازنده‌ی آن معرفی شده‌اند. در فصل ۵، به تنظیمات به کار برده شده برای آموزش مدل، مجموعه داده‌های مورد استفاده، نتایج مدل پیشنهادی روی مجموعه داده‌ها و تجزیه و تحلیل نتایج پرداخته می‌شود. در فصل ۶، یک نتیجه‌گیری از آزمایش‌های صورت گرفته ارائه می‌شود. در پایان، کارهایی که می‌توان در آینده برای بهبود مدل پیشنهادی انجام داد، ذکر می‌شوند.

فصل دوم
ادبیات پژوهش

۲-۱ مقدمه

در فصل قبل، پیش‌درآمدی بر مسئله‌ی درک زبان طبیعی ارائه شد. از اهمیت مسئله سخن گفته و کاربرد آن در وظیفه‌های مختلف پردازش زبان طبیعی مطرح شد. به منظور بررسی روش‌ها و مدل‌های ارائه شده در این حوزه، آشنایی با مفاهیم پایه و همچنین اجزاء سازنده‌ی مدل‌ها لازم است. در فصل پیش رو به تعریف اصطلاحات و مفاهیم پایه پرداخته می‌شود. سپس، انواع شبکه‌های عصبی شده و شیوه‌های تعبیه‌ی واژه‌ها معرفی می‌شوند.

۲-۲ مفاهیم پایه

در قسمت‌های مختلف این پایان‌نامه، اصطلاحاتی به کار برده شده که نیازمند روشنگری هستند. همچنین برخی از تعاریف مانند واژه و نشانه، مختص وظیفه‌ی درک زبان طبیعی هستند. در ادامه به معرفی ادبیات مورد استفاده در این پایان نامه پرداخته می‌شود. **یادگیری ماشین**[۱]: یادگیری ماشین مجموعه‌ای از الگوریتم‌ها و مدل‌هایی است که به کامپیوتر اجازه می‌دهد بدون برنامه نویسی صریح، الگو و دانش را از داده‌ها استخراج کند. در یادگیری ماشین، کامپیوتر می‌تواند پیش‌بینی کرده یا تصمیم‌گیری کند. هدف اصلی در یادگیری ماشین، تولید الگوریتمی است که بتواند تصمیمات خود را به داده‌های جدید نیز تعمیم دهد. یادگیری ماشین به منظور کاهش یا به حداقل رساندن دخالت انسان، در وظایف مختلف استفاده می‌شود.

یادگیری باناظر[۲]: یادگیری باناظر شیوه‌ای از یادگیری ماشین است که از مجموعه داده‌ی برچسب گذاری شده، برای پیش‌بینی خروجی داده‌های جدید استفاده می‌کند. درواقع، در یادگیری باناظر از مجموعه داده‌ی آموزشی استفاده می‌کنند که حاوی پاسخ‌های صحیح است.

یادگیری خودناظر[۳]: یادگیری خودناظر شاخه‌ای از یادگیری ماشین است که در آن، الگوریتم از خود داده‌ی آموزشی یاد می‌گیرد. در این شیوه، داده‌های آموزشی برچسب ندارند و الگوریتم باید از داده‌های آموزشی الگوها را استخراج کند.

سیستم گفت‌وگو: سیستم گفت‌وگو یک سیستم کامپیوتری است که برای شبیه‌سازی مکالمه با یک انسان به زبان طبیعی طراحی شده است. این سیستم می‌تواند اطلاعات را از طریق متن، گفتار[۴]، لمس[۵] و تصاویر دریافت کند.

سیستم گفت‌وگوی هدف محور: نوعی از سیستم گفت‌وگو است که برای کمک به کاربر در دستیابی به یک هدف خاص طراحی شده است. این سیستم معمولاً مبتنی بر مجموعه‌ای از اهداف و شرایط از پیش تعریف شده است و

---

[1] Machine Learning
[2] Supervised
[3] Self-Supervised
[4] Speech
[5] Haptic

از یک استراتژی تعریف شده برای هدایت کاربر به سمت آن اهداف استفاده می‌کند. سیستم گفت‌وگوی هدف محور برای انجام اهدافی مانند رزرو، سفارش محصولات و ارائه خدمات به مشتریان استفاده می‌شود.

درک زبان طبیعی: یک زمینه‌ی تحقیقاتی است که بر روی آموزش کامپیوترها، به منظور درک زبان انسان تمرکز دارد. برای این کار، کامپیوتر باید متن یا گفتار زبان طبیعی را دریافت کند و روابط معنایی[۱] را از آن استخراج کند. این روابط شامل تحلیل و درک نحو[۲]، معنا شناسی[۳] و عمل شناسی[۴] یک زبان است. این روابط با تعریف دو وظیفه[۵] استخراج می‌شوند؛ تشخیص هدف و تشخیص جای خالی.

تشخیص هدف: تشخیص هدف، فرآیند درک مقصود کلی[۶] از درخواست کاربر و تعیین هدف پشت آن است. هدف مشخص کننده‌ی عملی است که سیستم باید انجام دهد. به عنوان مثال، اگر کاربر بپرسد "آب و هوا امروز چگونه است؟" مقصود کلی کاربر آگاهی از آب و هوا، و هدف او به دست آوردن آب و هوای فعلی است. تشخیص هدف را می‌توان به‌عنوان طبقه‌بندی جمله‌ی کاربر، به دسته‌های از پیش تعیین‌شده تعریف کرد.

پر کردن جای خالی: پر کردن جای خالی، فرآیند استخراج اطلاعات مهم از جمله‌ی کاربر است که برای تکمیل درخواست کاربر، ضروری می‌باشد. این زیروظیفه شامل استخراج اطلاعات مرتبط مانند نام، تاریخ، مکان، اعداد، کدها، محصولات و غیره می‌شود. به عنوان مثال، اگر کاربر بپرسد "چه پروازهایی از لس آنجلس به نیویورک وجود دارد؟" وظیفه پر کردن جای خالی شامل استخراج مکان‌های لس آنجلس و نیویورک از ورودی کاربر است.

واژه[۷]**:** واژه در حوزه‌ی کاری درک زبان طبیعی، به معنای مجموعه‌ای از حروف و اعداد است که توسط یک جداکننده (فاصله‌ی خالی)[۸] از یکدیگر جدا شده‌اند. واژه می‌تواند شامل ترکیبی از حروف معنی‌دار یا ترکیبی از حروف، اعداد و علائمی باشد که حاوی اطلاعات است.

نشانه[۹]**:** از آنجا که نمی‌توان واژه‌های زبان طبیعی را مستقیما وارد شبکه‌ی عصبی کرد، ابتدا واژه را به نشانه تبدیل می‌کنند. در این فرایند، ممکن است واژه به اجزاء سازنده‌اش شکسته شود تا پیدا کردن روابط معنایی میان واژه‌ها برای شبکه ساده‌تر شود. سپس برای هر نشانه‌ی یکتا، یک شماره‌ی یکتا در نظر گرفته می‌شود. به فرایند شکستن واژه‌ها نشان کردن[۱۰] و به شماره‌ی یکتای نشان، شناسه‌ی نشان[۱۱] می‌گویند.

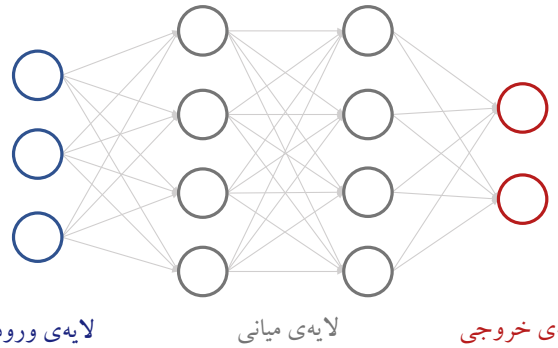
---

[1] Semantic Relations
[2] Syntax
[3] Semantics
[4] Pragmatics
[5] Task
[6] Overall Goal
[7] Word
[8] White Space
[9] Token
[10] Tokenize
[11] Token ID

۲-۳ شبکه‌ی عصبی

شبکه‌ی عصبی یکی از روش‌های یادگیری ماشین بوده که نقشی محوری در یادگیری عمیق ایفا می‌کند. این شبکه بر اساس مشاهدات قبلی، ضمن تشخیص الگوهای موجود، نسبت به چالش پیش روی خود تصمیم‌گیری کرده و پیش‌بینی‌های مورد نظر را انجام می‌دهد. منظور از مشاهدات قبلی، داده‌هایی است که شبکه در طول فرایند آموزش با آن برخورد داشته است. به منظور بررسی صحیح عملکرد، این داده‌ها نباید با سایر داده‌هایی که برای فرایند آزمایش استفاده می‌شوند اشتراکی داشته باشند. شبکه‌ی تغذیه‌به‌جلو نوعی شبکه‌ی عصبی مصنوعی است که اتصالات میان گره‌های آن تشکیل حلقه نمی‌دهد. در شبکه‌ی تغذیه‌به‌جلو، اطلاعات فقط از یک سمت جریان دارد؛ اطلاعات از گره‌های ورودی وارد شبکه شده، از لایه‌ی مخفی گذر کرده و به سمت گره‌های خروجی می‌رود. شکل ۲-۱، نوعی از شبکه‌ی عصبی تغذیه‌به‌جلو به نام شبکه‌ی تماماً متصل را ترسیم می‌کند. در این شبکه، گره‌ها در لایه‌های متفاوت قرار می‌گیرند. گره‌های یک لایه هیچ‌گونه اتصال مستقیمی به یکدیگر ندارند و گره‌های هر لایه، توسط وزن‌ها به گره‌های لایه‌ی بعد متصل می‌شوند. فرایند یادگیری در شبکه، توسط الگوریتم پس‌انتشار[1] صورت می‌گیرد. این الگوریتم، ابتدا داده‌های ورودی را به شبکه تغذیه کرده و سپس خطای شبکه را با محاسبه‌ی فاصله‌ی بین خروجی تولید شده و خروجی واقعی به دست می‌آورد. پس از آن، خطا به عقب در شبکه منتشر می‌شود و وزن‌ها را با توجه به بزرگی خطا تنظیم می‌کند. فرایند تنظیم وزن‌ها با استفاده از یک الگوریتم بهینه سازی[2] انجام می‌شود که وزن‌ها را در جهتی تنظیم می‌کند که خطا را به حداقل برساند. از میان الگوریتم‌های شناخته شده‌ی بهینه‌سازی، می‌توان به کاهش گرادیان[3]، کاهش گرادیان تصادفی و آدام اشاره کرد.

شبکه‌ی تماماً متصل چند لایه، برای کاربردهای کلاس‌بندی که ورودی آن‌ها یک بردار باشد، سودمند است؛ اما در صورتی که اطلاعات مورد نظر حاوی ترتیب باشند، امکان نگهداری اطلاعات ترتیبی مربوط به بردارها وجود ندارد.

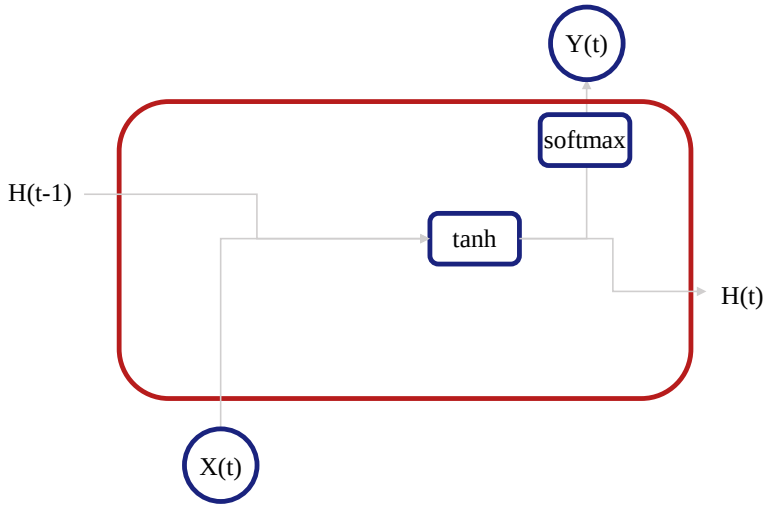


شکل ۲-۱ – شبکه‌ی عصبی تماماً متصل با دو لایه‌ی میانی. هرکدام از لایه‌های ورودی، میانی و خروجی با رنگ یکتا متمایز شده‌اند.

[1] Back-Propagation
[2] Optimization Algorithm
[3] Gradient Descent

۲-۴ شبکه‌ی عصبی بازگشتی

شبکه‌ی عصبی بازگشتی نوعی شبکه‌ی عصبی است که اتصالات میان گره‌های آن، حلقه تشکیل می‌دهند. درواقع، گره‌ها به نحوی به هم متصل هستند که به شبکه امکان به خاطر سپردن ورودی‌های گذشته را می‌دهد. به خاطر سپردن، این امکان را برای شبکه محیا می‌کند که تصمیمات آینده‌ی آن، وابسته به ورودی‌های پیشین نیز بشود [۲]. بدین ترتیب، خروجی شبکه می‌تواند حاصل ترتیبی از رویدادها باشد که در قالب بردار درآمده‌اند. از این رو، ورودی‌های یک شبکه‌ی عصبی بازگشتی، بصورت ترتیبی است. به عبارت دیگر ترتیب ورودی، خود دارای معنا و مفهوم است و باید ضمن دادن ورودی‌ها به شبکه، این مفهوم حفظ شود. بطور مثال جریان صدا، فریم‌های تصویر(فیلم)، و متن زبان طبیعی می‌توانند از شبکه‌های عصبی بازگشتی بهره ببرند. به این نوع ورودی‌ها، سری‌های زمانی[1] نیز می‌گویند.



شکل ۲-۲ - یک بلوک از شبکه‌ی عصبی بازگشتی

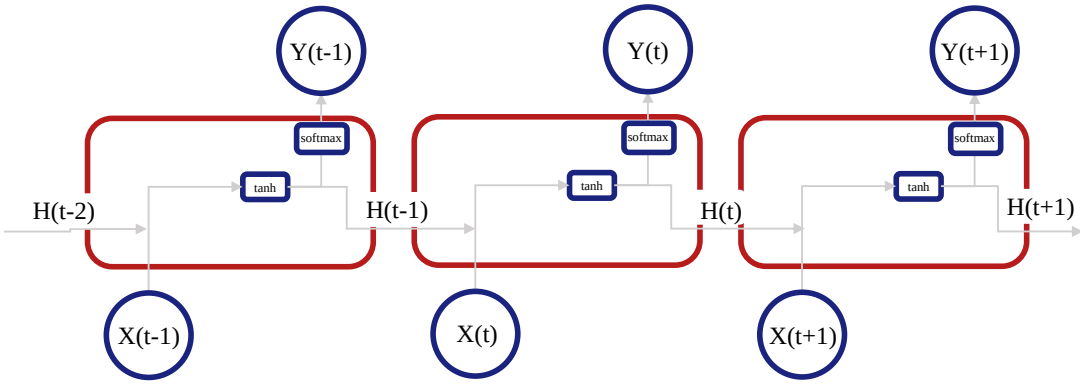
شکل ۲-۲ معماری شبکه‌ی عصبی بازگشتی را نمایش می‌دهد. در معماری این نوع شبکه‌ها، علاوه بر بردار ورودی شبکه $X_{(i)}$، بردار حالت مخفی[2] $H_{(i-1)}$ از مرحله‌ی قبلی شبکه نیز به عنوان ورودی به آن داده می‌شود. برای ورودی شبکه، وزن‌های بردار ورودی W_x و برای بردار حالت مخفی، وزن‌های بردار حالت مخفی W_h، جداگانه در نظر گرفته می‌شود. بدین ترتیب، بردار حالت مخفی هر مرحله از طریق معادله‌ی ۲-۱ و بردار خروجی از طریق معادله‌ی ۲-۲ محاسبه می‌شود.

$$H_t = tanh(W_x.X(t) + W_h.X(t-1) + b_h) \quad (۲-۱)$$

$$Y_t = softmax(W_Y.H_t + b_Y) \quad (۲-۲)$$

[1] Time Series
[2] Hidden State

در معادلات فوق، W_Y, W_H, b_h, b_Y همگی پارامترهای قابل آموزش هستند.



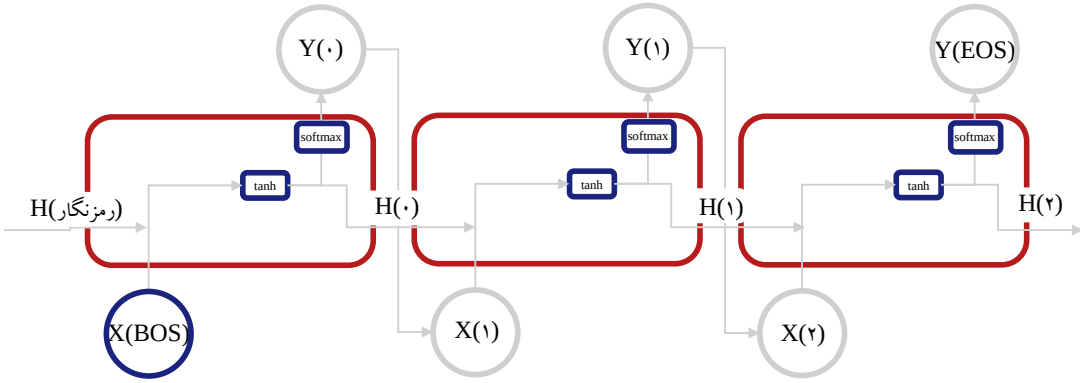
شکل ۲-۳ - شبکه‌ی عصبی بازگشتی گسترده شده در بعد زمان. در این شکل هر بلوک بیانگر یک گام زمانی است.

برای استفاده از شبکه‌ی عصبی بازگشتی در کاربرد پردازش متن، مطابق شکل ۲-۳، در هر گام زمانی[1]، یک واژه به همراه بردار حالت مخفی قبل به شبکه داده می‌شود. بردار تولید شده‌ی $Y_{(t)}$ به عنوان خروجی شبکه در گام زمانی t و بردار $H_{(t)}$ به عنوان بردار حالت مخفی به گام زمانی بعد داده می‌شود. با چنین ساختار شبکه‌ای، می‌توان ورودی با طول پویا به شبکه داد.

برای استفاده‌ی شبکه‌ی عصبی بازگشتی در وظیفه‌هایی مانند ترجمه، عموماً از معماری رمزنگار - رمزگشا استفاده می‌کنند. در رمزنگار، ورودی شبکه‌ی عصبی تبدیل به یک بردار می‌شود. این بردار در گام بعد، وارد رمزگشا می‌شود تا نشانه‌های مورد نظر را تولید کند. شکل ۲-۳، مورد استفاده‌ی شبکه‌ی عصبی بازگشتی در رمزنگار را توضیح می‌دهد. شکل ۲-۴ شیوه‌ی استفاده از شبکه عصبی بازگشتی در رمزگشا را به تصویر می‌کشد. در بخش رمزگشا، از آنجایی که نشانه‌های صحیح موجود نیست، نشانه‌ای قراردادی مانند (BOS)[2] را به عنوان اولین نشانه وارد شبکه می‌کنند. سپس در هر گام زمانی، نشانه‌ی تولید شده توسط مدل به عنوان ورودی مرحله‌ی بعد استفاده می‌شود. این کار تا زمانی تکرار می‌شود که نشانه‌ای قراردادی مانند (EOS)[3] تولید شود.

یک شبکه عصبی بازگشتی از نظر تئوری باید قادر به تولید دنباله‌هایی با هر پیچیدگی ای باشد اما در عمل مشاهده می‌کنیم که اگر تعداد گام زمانی در چنین شبکه‌ای به اندازه کافی زیاد باشد، در ذخیره سازی اطلاعات مرتبط با ورودی‌های قبلی به مدت طولانی ناتوان است [۳]. علاوه بر اینکه این خصیصه توانایی شبکه را در مدل سازی ورودی‌های طولانی تضعیف می‌کند، باعث می‌شود که شبکه در زمان تولید دنباله‌ای از کلمات، در معرض ناپایداری قرار بگیرد. مشکلی که وجود دارد این است که اگر پیش‌بینی‌های شبکه تنها وابسته به چند ورودی اخیر باشد و این ورودی ها خود

[1] Time-Step
[2] Beggining of Sentence
[3] End Of Sentence



شکل ۲-۴ – شبکه‌ی عصبی بازگشتی در نقش رمزگشا، گسترده شده در بعد زمان. در این شکل هر بلوک بیانگر یک گام زمانی است.

نیز توسط شبکه تولید شده باشند، شانس بسیار کمی برای تصحیح و جبران اشتباهات گذشته توسط شبکه وجود دارد. به طور مثال اگر ترجمه‌ی یک واژه در ابتدای جمله، به واژه‌ای در انتهای جمله وابسته باشد، و جمله‌ی یاد شده طولانی هم باشد، ممکن است شبکه نتواند به خوبی عمل کند. علت این امر دو مسئله‌ی گرادیان محو شونده[۱] و گرادیان انفجاری[۲] است [۴].

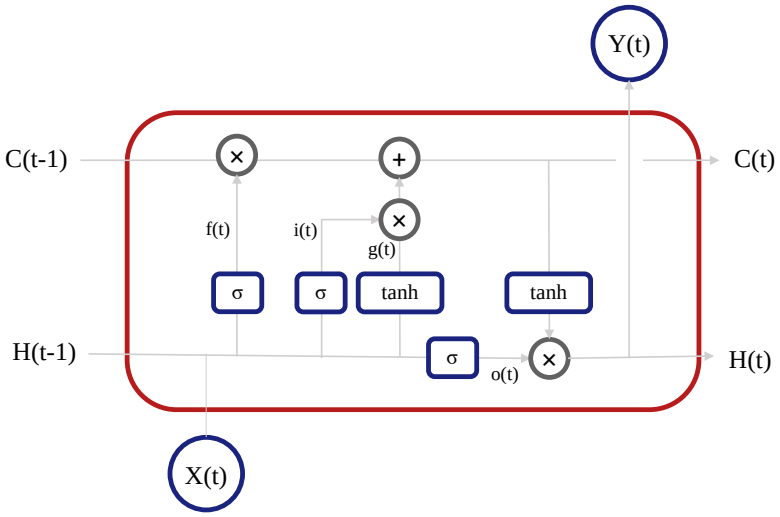
انواع مختلفی از شبکه‌های عصبی بازگشتی وجود دارد؛ بطور مثال GRU و LSTM . از آنجا که در اکثر کاربردهای زبان طبیعی شبکه‌ی LSTM مورد استفاده قرار می‌گیرد، تنها به معرفی این معماری پرداخته می‌شود.

۲-۵ حافظه‌ی کوتاه‌مدت طولانی

داشتن یک حافظه بلند مدت شبکه را به ثبات بیشتری می‌رساند؛ چراکه اگر شبکه بتواند از تاریخچه‌ی خود درک صحیحی پیدا کند، می‌تواند با مشاهده‌ی ورودی‌های ابتدایی، پیش‌بینی بهتری ارائه کند. شبکه‌ی حافظه‌ی کوتاه‌مدت طولانی[۳]، با معرفی سلول حافظه[۴] قادر است نسبت به حفظ حافظه فعلی از طریق دروازه‌های معرفی شده تصمیم‌گیری کند [۵]. ازنظر شهودی، اگر واحد LSTM داده‌ی مهمی در دنباله ورودی را در گام‌های ابتدایی تشخیص دهد، با استفاده از دروازه‌های معرفی شده می‌تواند این اطلاعات را طی گام‌های زمانی بعدی نیز منتقل کند. بدین ترتیب، LSTM می‌تواند این گونه وابستگی‌های بلندمدت را دریافت کرده و حفظ دارد. در ادامه به معرفی این دروازه‌ها پرداخته می‌شود.

شکل ۲-۵ ساختار درون یک بلوک LSTM را به‌تصویر می‌کشد. معماری LSTM دارای ۳ دروازه است؛ دروازه‌ی

[1] Vanishing Gradient
[2] Exploding Gradient
[3] Long Short-Term Memory (LSTM)
[4] Memory Cell



شکل ۲–۵ - یک بلوک از حافظه کوتاه‌مدت طولانی

ورودی[۱]، دروازه‌ی فراموشی[۲] و دروازه‌ی خروجی[۳]. این سه دروازه با همکاری یکدیگر حافظه‌ی فعلی سلول C_t را بروزرسانی و حالت فعلی شبکه H_t را تعیین می‌کنند. معادله‌ی ۲–۳ دروازه‌ی فراموشی f_t را نشان می‌دهد. این دروازه مشخص می‌کند چه میزان از اطلاعات قبلی باید دور ریخته شود.

$$f_t = \sigma(W_f.[H_{t-1}, X_t] + b_f) \tag{۲–۳}$$

در این رابطه σ تابع فعال ساز و W و b هردو پارامترهای قابل آموزش هستند. معادله‌ی ۲–۴ بیانگر دروازه‌ی ورودی است. این دروازه مشخص می‌کند چه میزان از اطلاعات ورودی در گام زمانی فعلی باید در حافظه‌ی فعلی سلول ذخیره شود.

$$i_t = \sigma(W_i.[H_{t-1}, X_t] + b_i) \tag{۲–۴}$$

معادله‌ی ۲–۵ نشانگر دروازه‌ی خروجی است. دروازه‌ی خروجی مشخص می‌کند که با توجه به سلول حافظه C_t، چه چیزی در خروجی قرار گیرد.

$$o_t = \sigma(W_o.[H_{t-1}, X_t] + b_o) \tag{۲–۵}$$

به همین ترتیب، حالت فعلی سلول حافظه ازطریق معادله‌ی ۲–۶ به دست می‌آید. در این معادله، $\odot$ بیانگر ضرب نقطه‌ای[۴]

[1] Input Gate
[2] Forget Gate
[3] Output Gate
[4] Dot Product

است.

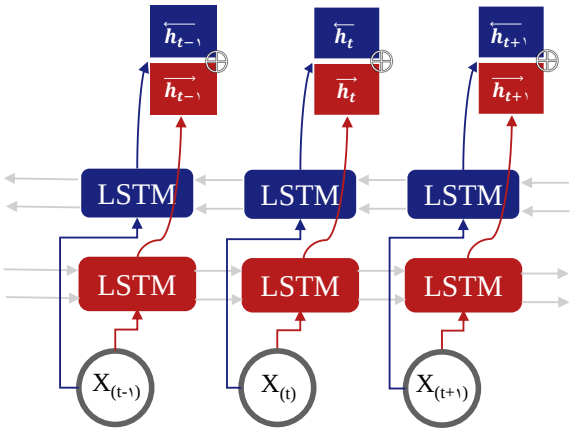
$$q_t = tanh(W_q.[H_{t-1}, X_t] + b_q)$$
$$C_t = f_t \odot C_{t-1} + i_t \odot q_t \quad (۶-۲)$$

در آخر، حالت مخفی H_t و خروجی Y_t در زمان t، توسط معادله‌ی ۷-۲ محاسبه می‌شود.

$$H_t = o_t \odot tanh(C_t)$$
$$Y_t = softmax(H_t) \quad (۷-۲)$$

۲-۵-۱ حافظه‌ی کوتاه‌مدت طولانی دوطرفه

تا کنون، پیش‌فرض استفاده از LSTM برای تعبیه‌ی کلمات به صورت ترتیبی و در یک جهت بود. اما در زبان طبیعی، وابستگی‌های معنایی دوطرفه هستند؛ به این معنا که تعبیه‌ی یک واژه می‌تواند علاوه بر وابستگی به واژه‌ی پیشین، به واژه‌ی بعد از خود نیز وابسته باشد. از این رو، بنظر می‌رسد که معماری اولیه‌ی LSTM این نیاز را برطرف نمی‌کند. یکی از راهکارهایی که برای حل این مسئله ارائه کردند در شکل ۶-۲ ترسیم شده است.

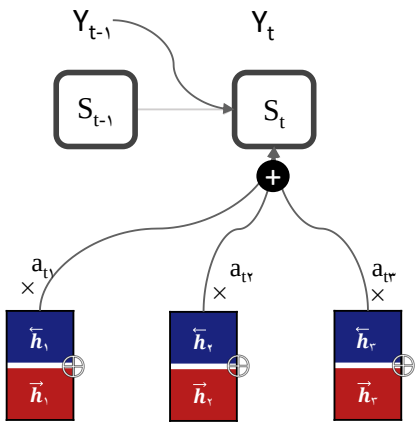


شکل ۲-۶ - شبکه‌ی LSTM دوطرفه، گسترده شده در بعد زمان.

شکل ۶-۲، یک LSTM دوطرفه را نشان می‌دهد که در بعد زمان گسترده شده است. در LSTM دوطرفه، یک لایه‌ی LSTM، ابتدا از سمت چپ به راست $\overrightarrow{h}$، و سپس لایه‌ای دیگر از سمت راست به چپ $\overleftarrow{h}$، توالی واژه‌ها را پردازش می‌کند. تعبیه‌ی حاصل به دست آمده از این دو عملیات، با یکدیگر الحاق $\oplus$ می‌شوند تا تعبیه‌ی پایانی دوطرفه $(\overrightarrow{h} \oplus \overleftarrow{h})$ برای واژه ساخته شود.

۲-۵-۲ مکانیزم توجه

روش دیگری که برای بهبود عملکرد شبکه‌ی LSTM استفاده می‌شود، مکانیزم توجه [۶] است. مکانیسم توجه به شبکه اجازه می‌دهد هنگام تصمیم گیری برای تولید هر نشانه، روی بخش‌های خاصی از ورودی تمرکز کند. این کار، با تخصیص وزن‌های مختلف به هر عنصر در دنباله ورودی انجام می شود. بدین ترتیب، مکانیزم توجه به عناصری که برای تصمیم گیری اهمیت بیشتری دارند وزن بیشتری می‌دهد. برداری که حاوی وزن‌ها است، بردار توجه نامیده می‌شود. این بردار توجه، به عنوان بخشی از ورودی، وارد شبکه‌ی LSTM می‌شود. شیوه‌ی وارد کردن این بردار، الحاق آن با بردار تعبیه‌ی ورودی است. شکل ۲-۷ شیوه‌ی محاسبه‌ی مکانیزم توجه را نشان می‌دهد.



شکل ۲-۷ - مکانیزم توجه تعریف شده در [۶].

در واقع، در یک شبکه‌ی رمزگشای LSTM که از مکانیزم توجه بهره می‌برد، حالت مخفی در هر مرحله‌ی زمانی، تابعی از حالت مخفی قبلی S_{t-1}، خروجی تولید شده‌ی قبلی Y_{t-1} و بردار توجه C_t است. این تابع را به صورت $s_i = f(S_{t-1}, Y_{t-1}, C_t)$ نمایش می‌دهند. لازم به ذکر است که در این معادله، مقدار C_t به ازاء هر واژه در خروجی متفاوت است. برای محاسبه‌ی C_t، جمع وزن‌داری از تمام حالت مخفی‌های رمزنگار ایجاد می‌شود، که وزن آن در طول فرایند یادگیری تغییر می کند. معادله‌ی ۲-۸ شیوه‌ی محاسبه‌ی C_t را نمایش می‌دهد.

$$c_t = \sum_{j=1}^{T_x} a_{tj} h_j \qquad (۲-۸)$$

در معادله‌ی ۲-۸، a_{tj} ماتریس متغیر قابل آموزش، به اندازه‌ی طول توالی ورودی T_x در طول توالی خروجی T_y است. ماتریس توجه برای همه‌ی موقعیت‌ها، درواقع متغیر بوده و در طول فرایند آموزش مدل در وظیفه‌ی مورد نظر، تغییر می کند. از این رو، وزن‌های بردار توجه برای هر نوع وظیفه‌ی پردازش زبان طبیعی، متفاوت است.

۲-۵-۳ ضعف ذاتی

با وجود برتری شبکه‌ی LSTM در برابر شبکه‌ی بازگشتی، چالش‌هایی برای استفاده از این شبکه نیز وجود دارد. با توجه به این که پارامترهای شبکه‌ی LSTM نسبت به شبکه‌ی بازگشتی بسیار بیشتر است، آموزش آن و همگرا شدن شبکه نیازمند زمان و داده‌های بیشتری است. همچنین، LSTM همیشه نمی‌تواند ظرافت‌های موجود در یک متن را درک کند. در ادامه، LSTM با وجود بهبود عملکرد شبکه‌ی بازگشتی، پوششی کوچک بر ضعف‌های ذاتی این نوع شبکه ایجاد کرده‌است [۷]؛ اما ضعف ذاتی همچنان وجود دارد و نمی‌توان عملکردی مناسب در ترتیب‌های بزرگ دید [۴، ۸]. از طرف دیگر، به‌خاطر ورود ترتیبی بردارها به شبکه و پردازش سلسله مراتبی اطلاعات، نمی‌توان به خوبی از ظرفیت سخت‌افزارهای امروزی که قابلیت پردازش موازی و سریع داده‌ها را دارند بهره برد. با توجه به مسائل ذکر شده، احتیاج به شبکه‌ای با پوشش واقعی این ضعف‌ها وجود دارد.

۲-۶ شبکه‌ی عصبی کانولوشنی

شبکه عصبی کانولوشنی[۱] از یک لایه ورودی، لایه‌های پنهان و یک لایه خروجی تشکیل شده است. در هر عملیات کانولشن، یک ماتریس هسته و یک ماتریس ورودی وجود دارد. ماتریس هسته[۲] که به آن فیلتر نیز می‌گویند، با حرکت روی ماتریس ورودی و ضرب نقطه‌ای مقادیر دو ماتریس که در معادله‌ی ۲-۹ تعریف شده است، ماتریس خروجی را تولید می‌کند. به ماتریس تولید شده نقشه‌ی ویژگی[۳] می‌گویند.

$$c = A\left(x \odot f + b\right) \tag{۲-۹}$$

در معادله‌ی فوق که عملیات کانولوشن را نشان می‌دهد، ورودی x، هسته f و تابع فعال‌ساز A، و عملیات کانولوشن با c برابر است.

حرکت هسته بر روی ماتریس ورودی می‌تواند همراه با پرش باشد، به این معنا که از محاسبه‌ی برخی از پنجره‌های ممکن صرف نظر شده و پنجره‌ی بعدی را محاسبه می‌شود. به پارامتر تعیین کننده‌ی پرش پنجره، گام[۴] می‌گویند. در صورتی که از پد[۵] استفاده نشود، ابعاد ماتریس ویژگی کوچک‌تر از ماتریس ورودی می‌شود. پد در شبکه‌ی کانولوشنی به معنای ایجاد موقعیت‌های جدید در ماتریس، با مقادیر ازپیش تعریف شده است. این مقادیر از پیش تعریف شده می‌تواند با توجه به سیاست خاصی از اعداد درون ماتریس انتخاب شود یا عدد مشخصی مانند صفر درون آن قرار گیرد.

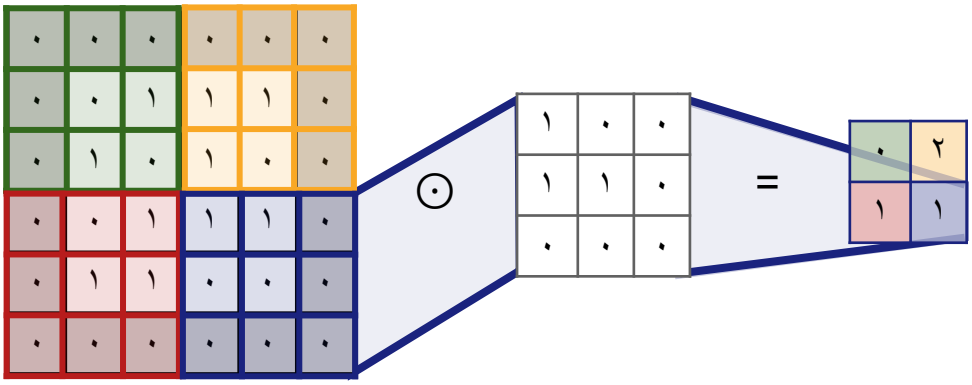
---

[1] Convolutional Neural Network (CNN)
[2] Kernel
[3] Feature Map
[4] Stride
[5] Pad

همچنین استفاده از گام بزرگتر از ۱ باعث کوچکی ماتریس ویژگی می‌شود. اندازه‌ی خروجی را می‌توان با استفاده از معادله‌ی ۲–۱۰ محاسبه کرد. در این معادله، گام با S اندازه‌ی ورودی با N، اندازه‌ی هسته با M، اندازه‌ی پد با P و اندازه‌ی خروجی با O مشخص شده است.

$$O = \frac{N - (M-1) + P}{S} \qquad (۱۰–۲)$$

ورودی شبکه‌ی کانولوشنی می‌تواند به اندازه‌ی دلخواه بعد داشته باشد. برای هر بعد می‌توان با استفاده از معادله‌ی ۲–۱۰، اندازه‌ی خروجی را محاسبه کرد. شکل ۲–۸ انجام عملیات کانولوشن یک هسته بر روی یک ماتریس را نمایش می‌دهد. برای سادگی در نمایش، کانال ورودی برابر ۱ در نظر گرفته شده است. لازم به ذکر است که کانال ورودی همواره با کانال هسته برابر است. در صورتی که کانال ورودی بزرگتر از ۱ باشد، مقدار هسته در هر کانال متفاوت با دیگری‌است.



شکل ۲–۸ – کانولوشن یک هسته بر روی ماتریس ورودی. در این شکل، هسته دارای طول ۳، عرض ۳ و کانال ۱ و ماتریس ورودی دارای طول ۳، عرض ۳ و کانال ورودی برابر ۱ است. در این عملیات گام برابر با ۳ و پد برابر ۲ در نظر گرفته شده است. همچنین، رنگ خاکستری نمایانگر پد بوده و هر رنگ سبز، زرد، قرمز و آبی برابر با یک پنجره است.

برای هر ورودی، یک بعد به عنوان تعداد کانال ورودی در نظر گرفته می‌شود. بعد از پایان اعمال یک هسته بر روی ورودی، آن بعد تبدیل به ۱ می‌شود. به عنوان مثال، در صورتی که ابعاد ورودی $(Height, Width, D)$ بوده، با فرض استفاده از پد، تعداد کانال ورودی برابر D، و و استفاده از یک هسته، خروجی برابر با $(Height, Width, 1)$ می‌شود. اما عملیات کانولوشن می‌تواند با چند هسته و با مقادیر اولیه‌ی مختلف برای هسته‌ها تکرار شود. بدین ترتیب، در صورتی که از تعداد K هسته استفاده شود، خروجی برابر با $(Heigh, Width, K)$ خواهد شد. در صورتی که شبکه‌ی کانولوشنی به صورت لایه‌ای باشد، نقشه‌ی ویژگی هر لایه، برای لایه‌ی مرحله‌ی بعد استفاده می‌شود. در

لایه‌های بعد از لایه‌ی کانولوشن، معمولاً لایه‌های ادغامی[1]، لایه‌های کاملاً متصل و لایه‌های نرمال‌سازی قرار می‌گیرند.

۲-۷ میدان تصادفی شرطی زنجیره‌ی خطی

میدان تصادفی شرطی[2] زنجیره خطی، نوعی مدل گرافیکی احتمالی است که ابزار قدرتمندی برای مدل‌سازی و پیش‌بینی روابط بین متغیرها در یک دنباله می‌باشد. مانند سایر CRFها، CRF زنجیره خطی هم، برای وظایف یادگیری با ناظر و هم بدون نظارت مانند طبقه بندی، تقسیم بندی[3] و برچسب گذاری استفاده می‌شود. CRF زنجیره خطی، به ویژه برای کارهایی که شامل داده‌های متوالی هستند، مانند پردازش زبان طبیعی مفید است. اگرچه در هیچ کجای معماری پیشنهادی، از CRF استفاده نشده است، اما بخاطر به استفاده‌ی برخی از مدل‌های پیشین از آن، در این بخش به طور خلاصه معرفی می‌شود.

با در نظر گرفتن بردار خروجی رمزنگار برای وظیفه‌ی تشخیص جای خالی $Y = \{y_1, y_2, \cdots, y_N\}$، توالی برچسب‌های جای خالی $s = \{s_1, s_2, \cdots, s_N\}$ و تمام توالی‌های برچسب ممکن برای s به عنوان $S(y)$، یک CRF احتمال وقوع دنباله‌ی ترتیبی s را در صورت مشاهده‌ی ورودی y، به ازای تمام sهای ممکن، با معادله‌ی ۲-۱۱ محاسبه می‌کند.

$$p(s|y; W, b) = \frac{\prod_{i=1}^{N} e^{W^T_{s_{i-1},s_i} y_i + b_{s_{i-1},s_i}}}{\sum_{s' \in S(y)} \prod_{i=1}^{N} e^{W^T_{s'_{i-1},s'_i} y_i + b_{s'_{i-1},s'_i}}} \qquad (۲-۱۱)$$

در معادله‌ی فوق، $W^T_{s_{i-1},s_i}$ بردار وزن و b_{s_{i-1},s_i} بایاس متناظر با جفت برچسب (s_i, s_{i-1}) است. همچنین، $W^T_{s'_{i-1},s'_i}$ بردار وزن و $b_{s'_{i-1},s'_i}$ بایاس متناظر با جفت برچسب (s'_i, s'_{i-1}) هستند. معادله‌ی فوق، احتمال وقوع دنباله‌ی s را در صورت مشاهده‌ی بردار تعبیه‌ی y حساب می‌کند. برای انتخاب برچسب‌ها در CRF، از الگوریتم Viterbi استفاده می‌شود. معادله‌ی ۲-۱۲، محتمل‌ترین دنباله برچسب‌گذاری s^* را محاسبه می‌کند.

$$s^* = \underset{s \in S(y)}{argmax}\, p(s|y; W, b) \qquad (۲-۱۲)$$

---

[1] Pooling

[2] Conditional Random Field (CRF)

[3] Segmentation

۲-۸ ترنسفورمرها

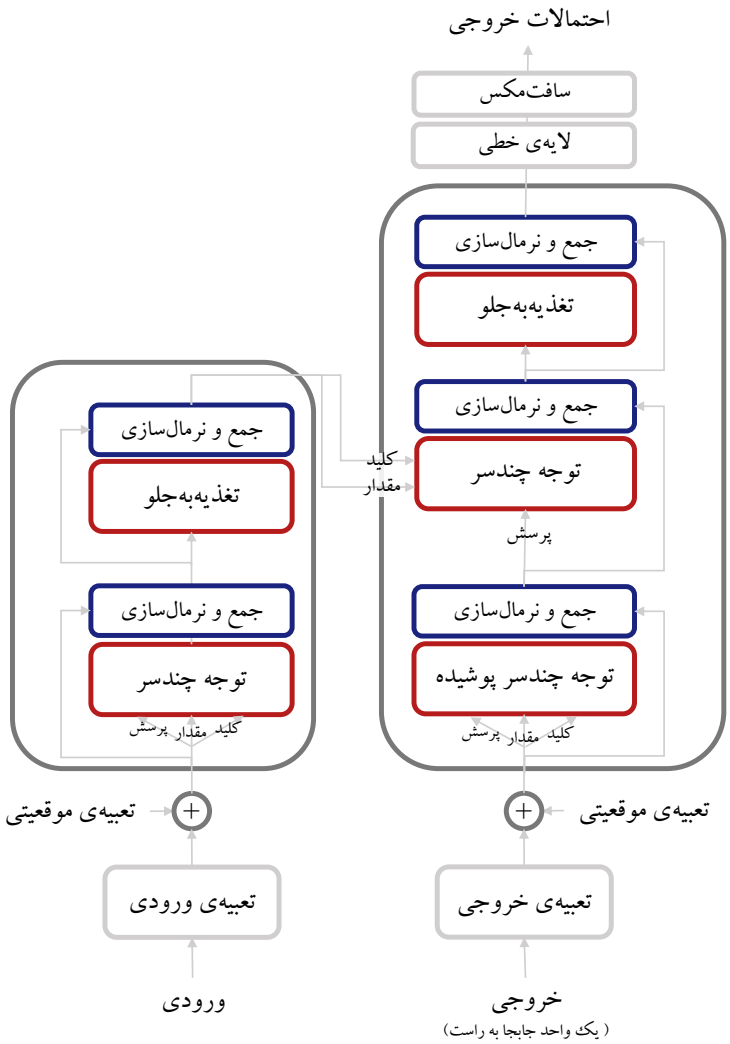
در سال‌های اخیر و با معرفی معماری ترنسفورمر[1]، جهش قابل توجهی در عملکرد مدل‌های پردازش زبان طبیعی ایجاد شده است. معماری ترنسفورمر، آغازگر سبک جدیدی از طراحی شبکه‌ی عصبی است. این معماری‌ها، صرفاً با اتکاء بر مکانیزم توجه، داده‌های ترتیبی را تجزیه، تحلیل و پردازش می‌کنند [۹]. ترنسفورمر معایب شناخته‌شده‌ی شبکه‌های عصبی بازگشتی یعنی گرادیان محو شونده و گرادیان انفجاری را تا حد زیادی برطرف می‌کند. در گذشته، برای حفظ اطلاعات موجود در توالی، داده به صورت ترتیبی وارد شبکه می‌شد؛ اما در شبکه‌ی ترنسفورمر، با تکیه بر مکانیزم توجه، داده‌ها به صورت موازی و همزمان وارد شبکه می‌شوند. این امر باعث افزایش چشمگیر سرعت آموزش شبکه و همچنین استفاده‌ی بهینه از سخت‌افزارهای امروزی شده است. از طرف دیگر، برای پردازش دوطرفه‌ی متن در شبکه‌ی عصبی بازگشتی، جمله یک بار از راست به چپ و سپس از چپ به راست وارد شبکه شده و تعبیه‌ی خروجی با هم الحاق می‌شد. اگرچه اینکار مفید بود، اما تعبیه‌ی حقیقی دوطرفه از جمله ایجاد نمی‌کرد. اتکاء شبکه‌ی ترنسفورمر بر مکانیزم توجه، موجب ایجاد یک تعبیه‌ی دوطرفه‌ی حقیقی می‌شود؛ چراکه ترنسفورمر به صورت همزمان به تمام موقعیت‌های موجود درون جمله دسترسی دارد.

با توجه به استفاده‌ی گسترده‌ی مدل پیشنهادی از این معماری، در بخش پیش رو به تفصیل به معرفی معماری ترنسفورمر پرداخته می‌شود. ابتدا ایده‌ی کلی این معماری برای پردازش داده‌های ترتیبی تشریح شده و سپس اجزایی که این ایده را محقق می‌کنند، معرفی می‌شوند. در پایان، تغییراتی که داده در جریان گذر از این معماری به خود می‌بیند، به تفصیل توضیح داده می‌شود.

شکل ۲-۹، معماری ترنسفورمر را نمایش می‌دهد. ساختار ترنسفورمر مبتنی بر سبک رمزنگار-رمزگشا است که در گذشته بصورت گسترده در وظیفه‌های پردازش زبان طبیعی مورد استفاده بوده است. این معماری بصورت پیش‌فرض برای وظیفه‌ی ترجمه کاربرد دارد. این معماری بصورت مؤلفه‌ای[2] بوده و می‌توان هرکدام از رمزنگار یا رمزگشا را جداگانه مورد استفاده قرار داد. در ساختار ترنسفورمر، به‌جای ورود ترتیبی نشانه‌ها به شبکه، کل نشانه‌ها بصورت همزمان وارد شبکه می‌شوند. با توجه به اینکه رمزنگار ترنسفورمر ذاتاً یک شبکه خود همبسته[3] نیست، برای حفظ اطلاعات مهمی که در ترتیب داده موجود است، از تعبیه‌ی موقعیتی[4] استفاده شده است. ازطرف دیگر، برای درک روابط معنایی درون جمله، از مکانیزم توجه چند سر استفاده شده است. چند سر بودن مکانیزم توجه به مدل کمک می‌کند تعداد بیشتری از روابط درون جمله را درک کند.

---

[1] Transformer
[2] Component
[3] AutoRegressive
[4] Positional Encoding

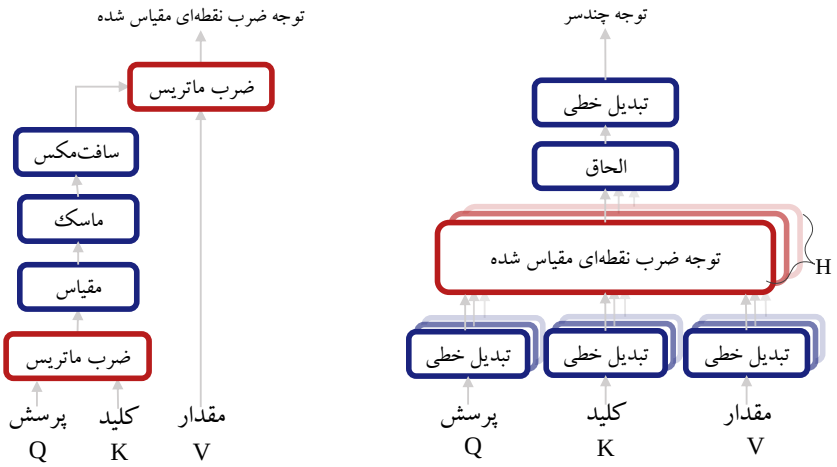


شکل ۲-۹ – معماری شبکه‌ی ترنسفورمر. سمت چپ معماری رمزنگار و سمت راست معماری رمزگشای ترنسفورمر را نمایش می‌دهد.

۲-۸-۱ مکانیزم توجه چند سر

هدف از تعریف مکانیزم توجه چند سر، اضافه کردن محتوای زمینه‌ای هر واژه از یک جمله، به هر واژه در جمله‌ی دیگر است. به عنوان مثال، فرض کنید دو جمله‌ی A و B وجود دارند و هدف، اضافه کردن پیش زمینه‌ی جمله‌ی A به جمله‌ی B است. با استفاده از مکانیزم توجه، می‌توان بردار جدیدی به ازاء هر نشانه‌ی موجود در جمله‌ی B ایجاد کرد که محتوای معنایی جمله‌ی A را در بر دارد. مکانیزم توجه این کار را با اضافه کردن جمع وزن داری از تعبیه‌ی هر نشانه‌ی موجود در A به تعبیه‌ی نشانه‌ی B انجام می‌دهد؛ یعنی ابتدا یک جمع وزن‌دار از تعبیه‌ی نشانه‌های A ایجاد کرده، آنها را با هم جمع می‌کند تا یک بردار به دست آید. بردار به دست آمده، بیانگر ارتباطات و مفاهیم به دست آمده از جمله‌ی A است که به نشانه‌ی فعلی درحال پردازش در جمله‌ی B مربوط می‌شود. این کار به ازاء تمام نشانه‌های موجود در B انجام می‌شود، تا تمام این نشانه ها حاوی پیش زمینه‌ی مربوطه شوند. مطابق شکل ۲-۱۰، این بردار پیش‌زمینه،

توسط سه بردار کلید[۱]، مقدار[۲] و پرسش[۳] ایجاد می‌شود. عموماً کلید و مقدار از جمله‌ی مبدأ (در مثال ذکر شده، جمله‌ی A) و پرسش از جمله‌ی مقصد (در مثال بالا، جمله‌ی B) ایجاد می‌شود.



شکل ۲-۱۰ - سمت چپ توجه ضرب نقطه‌ای مقیاس شده. سمت راست، توجه چندسر. در توجه چندسر، سرها به صورت موازی محاسبه می‌شوند.

معادله‌ی ۲-۱۳، نحوه‌ی محاسبه‌ی توجه ضرب نقطه‌ای مقیاس شده را نشان می‌دهد. با در نظر گرفتن d_k به عنوان بعد بردار کلید و d_v به عنوان بعد بردار مقدار در این معادله، کلید $K \in \mathbb{R}^{d_k}$، مقدار $V \in \mathbb{R}^{d_v}$، پرسش $Q \in \mathbb{R}^{d_k}$، عملیات ترانهادن[۴] T و . بیانگر ضرب ماتریس است.

$$Attention(Q, K, V) = V \times softmax(\frac{Q.K^T}{\sqrt{d_k}}) \qquad (۲-۱۳)$$

در مرجع [۱۰] نشان داده‌شده که برای مقادیر بزرگ d_k، مقادیر محصول نقطه‌ای بزرگ می‌شود که باعث شده تابع سافت‌مکس به نقطه‌ای برسد که گرادیان بسیار کوچکی دارد. برای مقابله با این اثر، آن‌ها با مقیاس $\frac{1}{\sqrt{d_k}}$ محصول نقطه‌ای را تغییر دادند؛ از این‌رو این معادله، ضرب نقطه‌ای مقیاس شده نام دارد.

برای محاسبه‌ی توجه چند سر می‌توان از معادله‌ی ۲-۱۴ استفاده کرد. در این معادله تعداد سرهای توجه H، و عملیات الحاق[۵] با علامت $\oplus$ مشخص شده است.

$$\begin{aligned} &Attention_h(Q, K, V) = (Q.W_h^Q, K.W_h^K, V.W_h^V) \\ &Attention = Attention_1 \oplus Attention \oplus ... \oplus Attention_H \\ &MultiHead = W^M.Attention \end{aligned} \qquad (۲-۱۴)$$

[1] Key
[2] Value
[3] Query
[4] Transpose
[5] Concatenation

در معادله‌ی ۲-۱۴، پیدا کردن روابط معنایی بین H سر تقسیم شده‌اند.

درصورت تمایل برای جلوگیری از اعمال توجه بر روی برخی از موقعیت‌های درون جمله‌ی مبدا، می‌توان ماسک $M \in \mathbb{R}$ را در معادله‌ی ۲-۱۳ اعمال کرد. در این صورت، معادله‌ی یاد شده به معادله‌ی ۲-۱۵ تغییر می‌یابد.

$$Attention_{Masked}(Q, K, V) = V \times softmax(\frac{Q.K^T}{\sqrt{d_k}} + M) \tag{۲-۱۵}$$

متعاقباً، مکانیزم توجه چند سر پوشیده، با معادله‌ی ۲-۱۶ و با جابجایی مکانیزم توجه عادی با پوشیده بدست می‌آید. در این صورت، ماتریس ماسک $M_{diagonal} \in \mathbb{R}^{S \times T}$ نیز باید همراه با سایر ورودی‌ها، به تابع تغذیه شود؛ که در آن S طول ترتیب مبدا و T طول ترتیب مقصد است.

$$\begin{aligned} &Attention^h_{masked}(Q, K, V) = (Q.W^Q_h, K.W^K_h, V.W^V_h) \\ &Attention_{masked} = Attention^1_{masked} \oplus Attention_{masked} \oplus ... \oplus Attention^H_{masked} \\ &MultiHead_{masked} = W^M.Attention_{masked} \end{aligned} \tag{۲-۱۶}$$

از مکانیزم توجه چند سر می‌توان برای مقاصد مختلف بهره برد. در ادامه به دو شیوه‌ی مرسوم آن و بینش پشت آن‌ها پرداخته می‌شود.

۲-۸-۱-۱ توجه به خود

زمانی که هر سه بردار کلید، مقدار و پرسش از یک جمله تولید شوند، به آن توجه به خود[1] می‌گویند. هدف از معرفی توجه به خود، ایجاد تعبیه‌ی جدیدی برای نشانه است که در آن، تعبیه‌ی سایر نشانه‌های جمله دیده شده است. به عبارتی، به استفاده از توجه به خود، تعبیه‌ی نشانه‌ها با پیش‌زمینه‌ای که نشانه در آن استفاده شده آمیخته می‌شود.

۲-۸-۱-۲ توجه متقابل

در صورتی که کلید و مقدار از یک جمله و پرسش از جمله‌ی دیگری تولید شود، آن را توجه متقابل[2] می‌نامند. توجه متقابل کاربرد مهمی در وظیفه‌ی ترجمه دارد؛ چراکه با استفاده از آن، می‌توان ارتباطات معنایی میان دو جمله، در دو زبان مختلف را پیدا کرد.

[1] Self-Attention
[2] Cross-Attention

۲-۸-۲ تعبیه‌ی موقعیتی

به علت بازگشتی نبودن شبکه‌ی ترنسفورمر، باید روش دیگری برای حفظ ترتیب نشانه‌های ورودی معرفی شود. به این دلیل، در [۱۰] تعبیه‌ی موقعیتی را معرفی کرده‌اند. برای موقعیت نشانه‌های درون جمله، مطابق معادله ۲-۱۷ دو تابع تعریف شده است. در یک جمله، N موقعیت مکانی (نشانه) و به ازاء هر نشانه، یک بردار d_{model} بعدی[1] وجود دارد. معادله‌ی ۲-۱۷ برای بعدهای زوج از تابع سینوسی، و برای بعدهای فرد تابع کسینوسی استفاده می‌کند. تعبیه‌ی موقعیتی، بعد یکسانی با بعد مدل d_{model} دارد؛ به این منظور که بتوان تعبیه‌ی موقعیتی را با بردار تعبیه جمع کرد.

$$PE_{(pos,2i)} = Sin(\frac{pos}{10000^{\frac{2i}{d_{model}}}})$$
$$PE_{(pos,2i+1)} = Cos(\frac{pos}{10000^{\frac{2i}{d_{model}}}}) \quad (۲-۱۷)$$

در معادله‌ی بالا، موقعیت نشانه در جمله pos و بعد مورد نظر آن i است. بدین ترتیب، برای بعد زوج و فرد هرکدام یک تابع وجود دارد. همچنین، طول موج‌های تولید شده یک افزایش هندسی از 2π به $2\pi.10000$ می‌باشد. این توابع در مقایسه با توابعی که موقعیت مکانی را در طول آموزش یاد می‌گیرند، برتری دارند؛ چراکه می‌توانند در طول فرایند تست، تعبیه‌ی مکانی جملاتی با طول دیده نشده در فرایند آموزش را، در زمان تست استقرا کنند [۱۰].

۲-۸-۳ جریان داده

در این بخش، تغییراتی که بر روی تعبیه‌ی اولیه‌ی نشانه‌ها صورت می‌گیرد تا به بردار نهائی تبدیل شود، تشریح می‌شوند. جریان داده به دو بخش زمان آموزش و زمان استنتاج[2] تقسیم می‌شوند. در ادامه، به توضیح این دو بخش پرداخته می‌شود.

۲-۸-۳-۱ زمان آموزش

به شکل ۲-۹ توجه کنید. در بخش رمزنگار، جمله‌ی ورودی ابتدا تبدیل به نشانه می‌شود. سپس برای این نشانه‌ها با استفاده از معادله‌ی ۲-۱۷ تعبیه‌ی موقعیتی ایجاد شده و با تعبیه‌ی اولیه نشانه‌ها جمع می‌شود. در مرحله‌ی بعد، ماتریس به دست آمده به صورت کامل و در یک مرحله وارد رمزنگار شده است؛ به این معنا که ورود ماتریس ترتیبی نیست. در بخش اول توجه به خود بر روی ماتریس ورودی ایجاد و سپس ماتریس توجه به دست آمده با ورودی جمع می‌شود. به عمل جمع کردن ماتریس توجه با ماتریس اولیه، اتصال باقی‌مانده[3] می‌گویند [۱۱]. آخرین مرحله‌ی رمزنگار، یک شبکه‌ی تغذیه‌به‌جلو است که روی ماتریس حاصل از این شبکه نیز یک اتصال باقی‌مانده اعمال می‌شود.

[1] Dimension
[2] Inferrence
[3] Residual Connection

در زمان آموزش، در بخش رمزگشا از شیوه‌ی اجبار معلم[1] استفاده می‌شود. در این شیوه نشانه‌های هدف نیز همراه با ورودی به شبکه داده می‌شود. بدین ترتیب، به جای استفاده از خروجی شبکه برای تولید نشانه‌های احتمالی و آموزش ترتیبی مدل، فرایند آموزش را می‌توان به صورت موازی و بسیار سریع‌تر انجام داد. برای ورود نشانه‌های هدف در معماری رمزگشا، همه‌ی نشانه‌ها یک واحد به سمت راست جابجا می‌شوند؛ این کار به‌این منظور صورت می‌گیرد که شبکه بتواند در زمان استنتاج، که نشانه‌ی هدف را ندارد، با وارد کردن یک نشانه (معمولا $[BOS]$)، فرایند تولید نشانه را در رمزگشا آغاز کند. در گام نخست، ماتریس موقعیت مکانی با ماتریس تعبیه‌ی نشانه‌ها جمع شده و سپس وارد رمزگشا می‌شود. سپس عملیات توجه به خود چندسر پوشیده بر روی ماتریس تعبیه‌ی نشانه‌ها انجام و یک اتصال باقی‌مانده اعمال می‌شود. پوشیده در اینجا، به منظور استفاده از یک ماسک ماتریس مثلثی بالایی[2] است که از پردازش موقعیت‌های غیرمجاز توسط مکانیزم توجه جلوگیری می‌کند. این موقعیت‌های غیرمجاز در زمان آموزش، موقعیت‌های جلوتر از نشانه‌ی درحال پردازش هستند؛ چراکه کمک گرفتن از آن‌ها در زمان آموزش، برای مدل وابستگی ایجاد می‌کند، و از آن‌جا که در زمان استنتاج نشانه‌های موقعیت‌های جلوتر ناشناخته هستند، عملکرد مدل با اختلال روبرو می‌شود. در گام دوم، کلید و مقدار از آخرین لایه‌ی رمزنگار، و پرسش از تعبیه‌ی نشانه‌ها، ایجاد و وارد واحد توجه متقابل می‌شود. در ادامه، ماتریس ایجاد شده از توجه متقابل با ماتریس تعبیه جمع، و وارد یک لایه‌ی تغذیه‌به‌جلو می‌شود. بر روی ماتریس حاصل از تغذیه‌به‌جلو نیز یک اتصال باقی‌مانده اعمال شده و بعد از یک لایه‌ی خطی و تابع سافت‌مکس، خروجی نهائی رمزگشای ترنسفورمر ایجاد می‌شود.

۲-۸-۳-۲ زمان استنتاج

تفاوت زمان استنتاج با آموزش در عدم وجود نشانه‌های هدف است؛ به همین دلیل، این تفاوت خود را در شیوه‌ی استفاده از رمزگشا نشان می‌دهد. در زمان استنتاج، رمزگشا تولید نشانه‌های خروجی را به صورت ترتیبی انجام می‌شود. بدین منظور، بعد از اتمام فرایند رمزنگاری، با دادن نشانه شروع $[BOS]$ به‌عنوان اولین نشانه به رمزگشا فرایند تولید خروجی آغاز می‌شود. سپس تمام فرایندهای ذکرشده در زمان آموزش تکرار می‌شوند تا یک نشانه به عنوان خروجی تولید شود. خروجی تولید شده مجددا با نشانه‌ی $[BOS]$ الحاق شده و به عنوان ورودی مرحله بعد به رمزگشا تغذیه می‌شود. مراحل ذکر شده، تا رسیدن به نشانه‌ی $[EOS]$ تکرار می‌شوند.

---

[1] Teacher Forcing

[2] Upper Triangular Matrix

۲-۹ تعبیه‌ی نشانه‌ها

برای این که جملات وارد شبکه‌ی عصبی شوند، باید ابتدا به واژه‌ها شکسته و واژه‌ها تبدیل به نشانه شوند. برای وارد کردن هر نشانه به شبکه، باید تعبیه‌ای از آن نشان وجود داشته باشد. تعبیه برداری از اعداد حقیقی است که بیانگر مفهوم آن نشانه در زبان طبیعی است. روش‌های تعبیه‌ی نشانه‌ها را می‌توان به دو گروه تقسیم نمود؛ تعبیه‌ی برداری ثابت و مدل‌های زبانی. در ادامه، این دو شیوه معرفی می‌شوند.

۲-۹-۱ تعبیه‌ی برداری ثابت

تعبیه برداری ثابت، نوعی از تعبیه است که در آن نمایش یک واژه یا عبارت معین از روی مجموعه‌ی متنی[۱] ایجاد می‌شود و در طول زمان تغییر نمی‌کند. پس از محاسبه‌ی تعبیه‌ی هر نشانه، فارغ از این که نشانه در چه جمله‌ای استفاده شده باشد، تعبیه‌ی آن ثابت در نظر گرفته می‌شود. دو شیوه‌ی مطرح تعبیه‌ی برداری ثابت Word2Vec و GloVe هستند. در ادامه، به صورت خلاصه این دو تعبیه معرفی می‌شوند.

۲-۹-۱-۱ Word2Vec

Word2Vec ، مجموعه‌ای از مدل‌های شبیه به هم است که برای تعبیه‌ی واژه‌ها استفاده می‌شود. این مدل‌ها از شبکه‌ی تغذیه به جلوی کم‌عمق استفاده می‌کنند که برای بازسازی پیش‌زمینه‌ی زبانی واژه‌ها آموزش دیده‌اند. Word2Vec مجموعه بزرگ متنی را به عنوان ورودی خود گرفته و یک فضای برداری با چند صد بعد تولید می‌کند. به هر واژه‌ی یکتا در مجموعه‌ی متنی، یک بردار متناظر در فضای برداری اختصاص داده می‌شود. Word2Vec از دو روش برای تولید این بردارها بهره می‌برد؛ کیسه‌ی واژه‌های ممتد[۲] و اسکیپ‌گرام ممتد[۳]. در هر دو روش، یک پنجره لغزان[۴] از واژه‌های پیش‌زمینه، حول یک واژه‌ی اصلی در نظر گرفته می‌شود و به ازاء تمام واژه‌های موجود در مجموعه‌ی متنی عملیات محاسبه تکرار می‌شود. در روش کیسه‌ی واژه‌ها، مدل واژه‌ی اصلی را با استفاده از واژه‌های پیش‌زمینه‌ی موجود در پنجره حدس می‌زند. در این روش، ترتیب واژه‌های پیش‌زمینه تفاوتی ندارد. در روش اسکیپ‌گرام، مدل با گرفتن واژه‌ی اصلی، واژه‌های پیش‌زمینه‌ی آن را پیش‌بینی می‌کند. در این روش، واژه‌هایی که در پنجره ازنظر فاصله به واژه‌ی اصلی نزدیک‌تر هستند وزن بیشتری دارند. نویسندگان Word2Vec اظهار کرده‌اند که کیسه‌ی واژه‌ها سرعت بیشتری دارد و در مقابل عملکرد اسکیپ‌گرام در مواجه با واژه‌های کمیاب بهتر است [۱۲].

---

[1] Corpus
[2] Continuous bag-of-words
[3] Continuous Skip-Gram
[4] Sliding Window

GloVe ۲-۱-۹-۲

GloVe از شبکه‌ی عصبی استفاده نمی‌کند. در این شیوه‌ی تعبیه، ابتدا یک مجموعه‌ی متنی بزرگ پیمایش و ماتریس هم‌آیی واژه‌ها[1] تشکیل می‌شود. سپس با این فرض که محصول نقطه‌ای بردار تعبیه‌ی دو واژه که در ماتریس هم‌آیی موجود هستند، باید با تعداد هم‌آیی آن دو واژه رابطه داشته باشد، تعبیه‌ی واژه‌ها را در فضای برداری شکل می‌دهد [۱۳].

۲-۹-۲ مدل‌های زبانی

نقص اصلی تعبیه برداری ثابت، این است که قادر به درک پیش‌زمینه‌ی مربوط به واژه نیست. این مشکل، عملکرد آن را برای پیش‌بینی دقیق معنای جمله و همچنین تولید متن محدود می‌کند. علاوه بر این، تعبیه‌های تولید شده ثابت‌اند؛ به این معنی که پس از پایان فرایند آموزش، نمی‌توان آنها را برای کاربر خاص بهینه یا با واژه‌های جدید که وارد زبان می‌شوند سازگار کرد. یک مدل زبانی[2] نسبت به زمینه کلام آگاه است؛ یعنی معنای جمله‌ای که واژه در آن‌ها آمده را درک می‌کند. این ویژگی باعث شده مدل زبانی قدرت پیش‌بینی دقیق‌تری داشته باشد. از این‌رو، یک مدل زبانی با دیدن یک جمله و در چالش حدس زدن یک جای خالی، می‌تواند واژه‌های مناسب تری را برای آن جای خالی پیشنهاد کند. در واقع، هدف اصلی مدل زبانی، محاسبه‌ی احتمال وقوع یک واژه، درون یک جمله است. به عنوان مثال، اگر عبارت "من قصد خواندن یک..." به مدل زبانی داده شود، مدل احتمال وقوع "کتاب" در ادامه‌ی این جمله را بیشتر از "خودرو" در نظر می‌گیرد. از طرف دیگر، تعبیه‌ی واژه‌ها در مدل زبانی، یک بردار ثابت نیست؛ بلکه تابعی است از سایر واژه‌هایی که در جمله حضور دارند. آموزش مدل‌های زبانی معمولا کاری هزینه‌بر و گران است. معمولا در فرایند آموزش آن‌ها، یک یا چند وظیفه بر روی حجم زیادی از داده‌های متنی تعریف می‌شود.

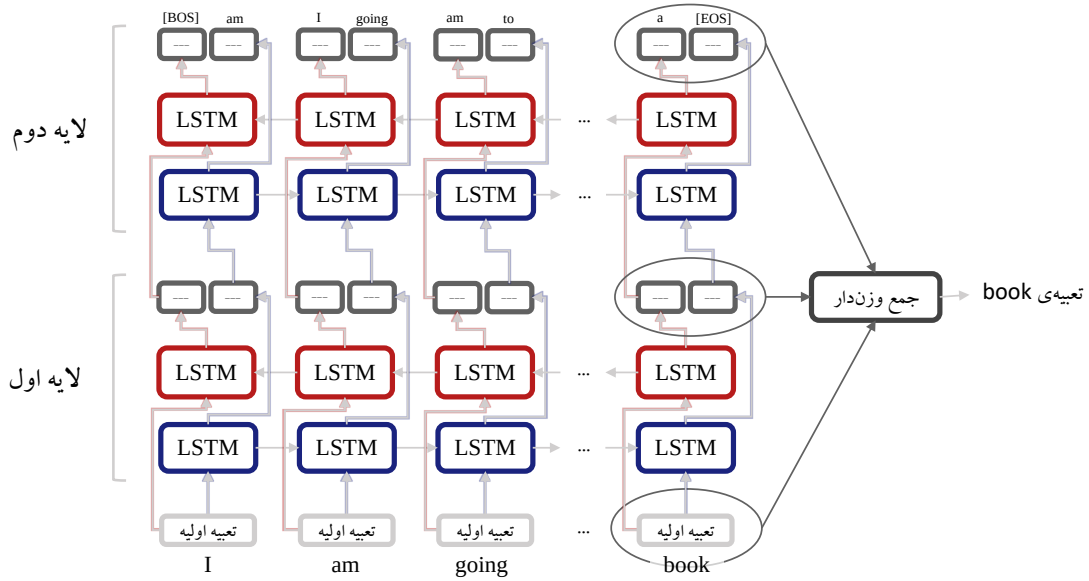
در ادامه، به معرفی دو مدل زبانی اِلمو که مبتنی بر LSTM و بِرت که مبتنی بر ترنسفورمر است پرداخته می‌شود.

۱-۲-۹-۲ المو

همانطور که گفته شد، برای ورود واژه‌ها به شبکه‌ی عصبی، باید واژه تبدیل به شناسه و شناسه تبدیل به بردار شود. در اِلمو[3]، برای ایجاد بردار اولیه‌ی نشانه، از تعبیه‌ی الفبایی[4] استفاده می‌شود. بدین منظور، ابتدا واژه به سطح الفبا شکسته شده، برای هر حرف الفبا یک شناسه‌ی یکتا، و برای هر شناسه یک بردار در نظر گرفته می‌شود. در مرحله‌ی بعد، مجموعه تعبیه‌ی الفبای واژه وارد شبکه‌ی کانولوشنی با اندازه هسته‌ی گوناگون می‌شود. مدل اصلی المو از هسته‌هایی با اندازه‌های ۱ تا ۷، با کانال‌های ۳۲، ۳۲، ۶۴، ۱۲۸، ۲۵۶، ۵۱۲ و ۱۰۲۴ استفاده می‌کند. خروجی‌های هر لایه کانولوشن

---

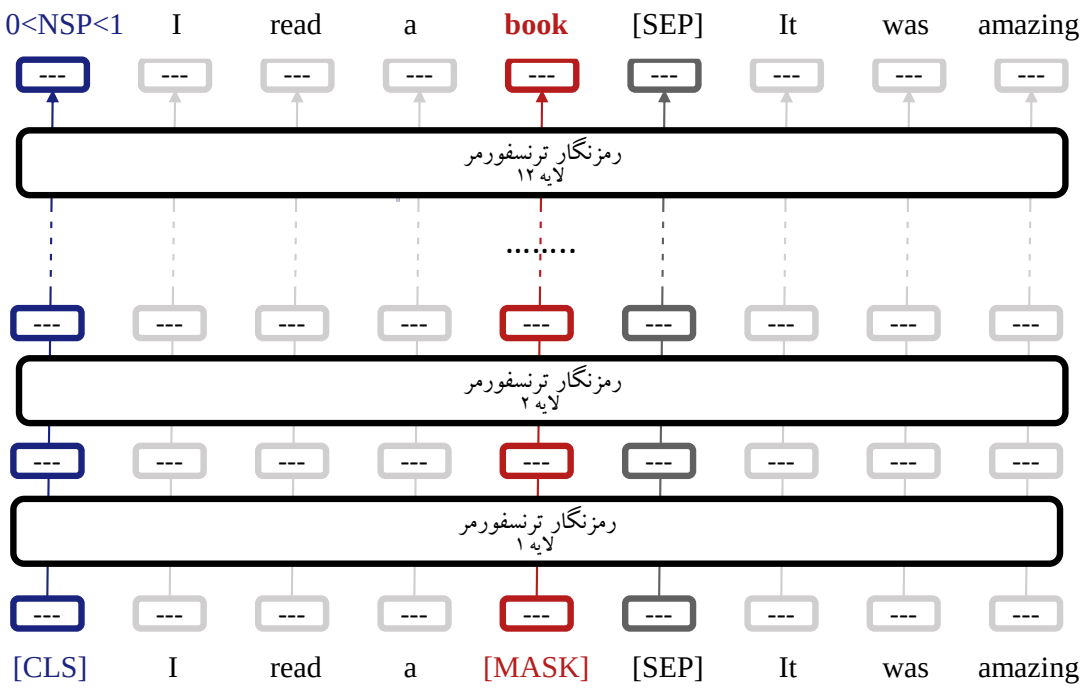
[1] Word Co-occurance Matrix
[2] Language Model
[3] ELMo: Embedding From Language Models
[4] Character Embedding



شکل ۲-۱۱ - معماری مدل زبانی المو. ورودی شبکه "I am going to read a book" است. رنگ آبی برای تعبیه‌ی چپ به راست و رنگ قرمز برای راست به چپ اختصاص دارد. برای جلوگیری از پیچیدگی شکل، اتصال باقی‌مانده در این شکل رسم نشده است.

پس از آن Max-Pool شده و به هم الحاق می‌شوند تا برداری با تعداد بعد ۲۰۴۸ را بسازند. این خروجی به عنوان تعبیه‌ی اولیه واژه در نظر گرفته می‌شود تا مطابق شکل ۲-۱۱ وارد شبکه‌ی LSTM شود. المو در ساختار شبکه‌ی خود از LSTM در دو جهت و به صورت دو لایه استفاده می‌کند. باید توجه داشت که ساختار شبکه با دو لایه‌ی Bi-LSTM متفاوت است، چراکه ابتدا دو لایه متن را از چپ به راست و دو لایه را از راست به چپ پردازش می‌کند. در لایه‌ی چپ به راست، در هر گام زمانی، وظیفه‌ی LSTM حدس زدن واژه‌ی بعدی، و در لایه‌ی راست به چپ، وظیفه‌ی آن حدس زدن واژه‌ی پیشین است. میان تعبیه‌ی اولیه و خروجی لایه‌ی اول، یک اتصال باقی‌مانده اضافه می‌شود. این عمل به مقاومت بیشتر شبکه درمقابل گرادیان محو شونده کمک می‌کند [۱۱]. در نهایت، برای تولید تعبیه‌ی پایانی، در هر لایه به صورت جداگانه، تعبیه‌ی چپ به راست با راست به چپ به یکدیگر الحاق می‌شوند. در پایان، جمع وزن‌دار میان تعبیه اولیه، خروجی الحاقی لایه‌ی اول و خروجی الحاقی لایه‌ی دوم را محاسبه می‌کنند. وزن‌های مربوط به این جمع وزن‌دار، برای هر وظیفه در پردازش زبان طبیعی متفاوت است.

خروجی هر لایه‌ی المو معنا و کاربرد خاص خود را دارد. خروجی لایه‌ی اول، بیشتر درمورد صرف و نحو زبان و خروجی لایه‌ی دوم، مرتبط به معنای جمله و واژه‌ها است [۱۴].



شکل ۲-۱۲ – معماری مدل زبانی برت. ورودی شبکه دو جمله‌ی "I read a book. It was amazing" است. واژه‌ی book با استفاده از نشانه‌ی $[MASK]$ پوشیده شده است. این تصویر مربوط به نسخه‌ی $BERT_{base}$ می باشد.

۲-۹-۲-۲ برت

برت[1] یک مدل زبانی پیش‌آموز شده بر روی ترنسفورمرها است که برای استفاده در امور مختلف پردازش زبان طبیعی مورد استفاده قرار می‌گیرد [۱۵]. با معرفی ترنسفورمرها [۱۰]، امکان ارائه یک تعبیه‌ی دوطرفه‌ی حقیقی فراهم شد. در [۱۵] برای بهره‌گیری از قدرت ترنسفورمر، دو وظیفه‌ی خودناظر بر روی یک معماری لایه‌ای معرفی شده و ورودی‌ها به ترنسفورمر ساختارمند گردید. برای استفاده از برت، می‌توان مدل پیش‌آموز شده را دریافت کرد و با توجه به نیاز، آن را روی دیتاست مورد نظر تنظیم دقیق کرد. این شبکه در مدل پایه‌ی خود از ۱۲ لایه ترنسفورمر و در مدل بزرگ خود از ۲۴ لایه ترنسفورمر استفاده می‌کند. مدل برت در شکل ۲-۱۲ نمایش داده شده است.

همانطور که بالا تلویحا گفته شد، برت در دو مرحله آموزش داده‌می‌شود؛ پیش‌آموزش[2] و تنظیم دقیق[3].

پیش‌آموزش: در این مرحله، برت یاد می‌گیرد که زبان و پیش زمینه‌ی متن چیست. در مرحله پیش‌آموزش، مدل دو وظیفه‌ی متفاوت اما مرتبط را به صورت همزمان روی یک مجموعه متنی بدون برچسب یاد می‌گیرد. این دو وظیفه پیش‌بینی جمله‌ی بعد و مدل‌سازی زبانی پوشیده هستند.

---

[1] "Bert: Bidirectional Encoder Representations from Transformers"

[2] Pre-Training

[3] Fine-Tune

الف) پیش‌بینی جمله‌ی بعد[1]: این وظیفه بر پایه‌ی این پیش‌فرض که دو جمله‌ی درون یک پاراگراف ازنظر معنایی به هم مرتبط هستند تعریف شده است. بر این اساس، در وظیفه‌ی NSP، برای دو جمله‌ی درون پاراگراف عدد ۱ و دو جمله‌ای که در یک پاراگراف نیستند عدد ۰ در نظر گرفته شده است. در این بخش، دو جمله‌ای که به مدل داده می‌شوند توسط نشانه‌ی $[SEP]$ از یکدیگر جدا شده‌اند. هدف مدل در این بخش تعیین ارتباط این دو جمله با یکدیگر است. نتیجه‌ی این تشخیص در اولین نشانه از خروجی (در شکل ۲-۱۲ نشانه‌ی آبی رنگ) قرار دارد.

ب) مدل‌سازی زبانی پوشیده[2]: در این وظیفه مدل سعی می‌کند زمینه‌ی محتوایی متن را به صورت دوطرفه یاد بگیرد. به این منظور برخی واژه‌ها از جمله حذف شده و نشانه‌ی $[MASK]$ جای آن‌ها را می‌گیرد. سپس مدل سعی می‌کند با توجه به سایر واژه‌های موجود در جمله، جای خالی را با واژه‌ی مناسب پر کند. حذف واژه، با احتمال ۱۵ درصد روی تمام واژه‌های ورودی به جز واژه‌های کلیدی $[NSP]$ و $[SEP]$ صورت می‌گیرد.

تنظیم دقیق: پس از مرحله‌ی پیش‌آموزش، برت زبان را یاد می‌گیرد اما نحوه‌ی استفاده از آن برای حل مسئله را بلد نیست. در مرحله‌ی تنظیم دقیق، مدل روش حل مسئله را آموزش می‌بینند. بدین منظور، با توجه به وظیفه‌ی مورد انتظار از مدل زبانی، نحوه‌ی آموزش مدل و ساختار رمزگشای آن مشخص می‌شود. به طور مثال می‌توان برای وظیفه‌ی پیش‌بینی احساسات، با قرار دادن یک شبکه‌ی تغذیه به جلو روی خروجی $[NSP]$ و آموزش مدل روی یک مجموعه داده‌ی پیش‌بینی احساسات، مدل را برای این وظیفه، تنظیم دقیق کرد. این کار نسبت به پیش‌آموزی، مدت زمان کمتری طول خواهد کشید؛ زیرا که شبکه به آموزش وزن‌های شبکه‌ی تغذیه به جلو که به تازگی اضافه شده‌اند می‌پردازد و بقیه‌ی پارامترهای مدل برت اندکی تغییر می‌کنند.

۲-۱۰ جمع‌بندی

در این فصل، ابتدا مفاهیم پایه‌ی این پایان‌نامه ذکر شد. سپس، مقدمه‌ای بر شبکه‌ی عصبی مطرح گردید. برای پردازش داده‌های ترتیبی، شبکه‌ی عصبی بازگشتی و LSTM معرفی شدند. در ادامه، گفته شد که این نوع شبکه‌ها از مشکل گرادیان محو شونده رنج می‌برند. برای حل این مشکل مکانیزم توجه معرفی شد. اگرچه مکانیزم توجه مفید بود، اما اصل مسئله را حل نمی‌کرد. بنابراین، معماری ترنسفورمر معرفی گشت که از اساس ترتیبی نبوده و داده‌ها را به صورت همزمان وارد شبکه می‌کرد. همچنین، CRF زنجیره خطی و شبکه‌ی عصبی کانولوشنی معرفی شدند. در پایان این فصل نیز، انواع تعبیه و مدل‌های زبانی معرفی گردیدند.

---

[1] Next Sentence Prediction (NSP)

[2] Masked Language Modeling (MLM)

فصل سوم
کارهای پیشین

۳-۱ مقدمه

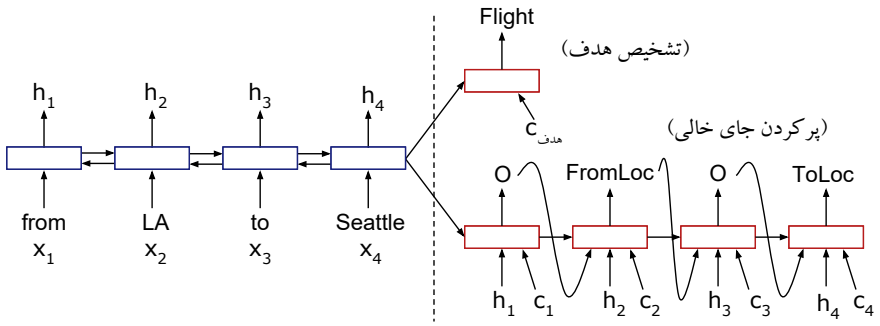
در دو فصل گذشته، مسئله‌ی درک زبان طبیعی تعریف شده و پایه‌های سازنده‌ی آن معرفی شدند. سپس مفاهیم پایه که برای فهم موضوع لازم بود ذکر شده و در ادامه، ساختار شبکه‌های عصبی گوناگون، که در حل مسئله‌ی درک زبان استفاده می‌شوند، تشریح شدند. برای بستر سازی لازم جهت معرفی مدل، لازم است کارهای پیشین معرفی شوند تا تلاش‌هایی که تا کنون برای بهبود دو وظیفه‌ی تشخیص هدف و پر کردن جای خالی صورت گرفته است، هویدا شود. بر این اساس، در فصل پیش رو، تحقیقات پیشین در این زمینه و مدل‌های ارائه شده توصیف می‌شوند. از آنجا که روش‌های ارائه شده با یکدیگر متفاوت بوده و اشتراکات اندکی دارند، دسته‌بندی آن‌ها به دسته‌های معنادار، کار دشواری است. در فصل پیش رو، مدل‌های ارائه شده بر پایه‌ی سبک تعبیه و اساس معماری شبکه تقسیم بندی می‌شوند.

در گذشته، دو وظیفه‌ی پر کردن جای خالی و تشخیص هدف جداگانه مورد تحقیق قرار می‌گرفتند [۱۶، ۱۷]؛ اما اخیراً، نشان داده شده که آموزش مشترک این دو وظیفه، اثر مثبتی بر عملکرد هر دوی آن‌ها دارد [۱۸-۲۳]. این امر نشان داده است که رابطه‌ای قوی بین این دو وجود دارد. آموزش همزمان این دو وظیفه به این صورت انجام می‌گیرد که برای هر دوی آن‌ها یک رمزنگار مشترک استفاده شده، اما برای هر وظیفه یک رمزگشای مستقل تعریف می‌شود. تمامی مدل‌هایی که در این زمینه‌ی درک زبان طبیعی وجود دارند، از این ساختار بهره می‌برند چراکه برتری آن ثابت شده است.

۳-۲ مبتنی بر نمایش تعبیه برداری ثابت

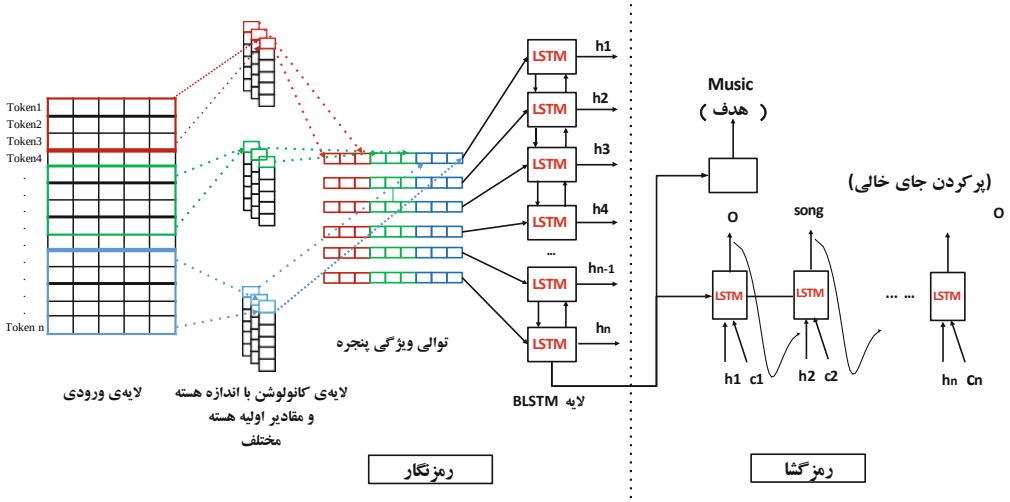
مدت‌ها است که مکانیزم توجه به همراه LSTM مورد استفاده قرار می‌گیرد [۲۲، ۲۴]. ترکیب این دو در وظیفه‌ی درک زبان طبیعی نیز مورد استفاده قرار گرفته و در زمان خود بهترین عملکرد را در پی داشت. از طرفی، به علت رابطه‌ی یک به یک واژه‌های ورودی به برچسب‌های خروجی، نیازی به فضای باز احتمالاتی خروجی یک LSTM نبود. به عبارت دیگر، نیاز نبود طول جمله‌ی خروجی پویا باشد؛ بلکه خروجی با طول ثابت و هم اندازه با ورودی مورد انتظار است. از طرف دیگر، آگاهی از اینکه کدام موقعیت از ورودی، به خروجی مربوط است، خود یک مزیت محسوب می‌شود و باید از آن در طراحی معماری شبکه استفاده کرد [۲۵]. در طراحی مدل LSTM تراز شده، از رابطه‌ی یک به یک ورودی با خروجی بهره برده شد [۱۸]. شکل ۳-۱ معماری مدل تراز شده را نمایش می‌دهد. در این معماری، برای رمزنگار، از یک LSTM دوطرفه که به صورت مشترک در دو رمزگشا مورد بهره‌وری قرار می‌گرفت. در رمزگشای تشخیص هدف، از بردار توجه هدف، و همچنین آخرین حالت مخفی LSTM رمزنگار برای کلاس بندی اهداف بهره برده شده بود. اما در رمزگشای پر کردن جای خالی، یک LSTM یک طرفه تعریف شده و از رابطه‌ی یک به یک ورودی و خروجی استفاده شد. به این منظور، در هر گام زمانی، علاوه بر بردار توجه، بردار حالت مخفی مربوط به

گام زمانی ورودی که متناظر با گام زمانی خروجی بود، به مدل تغذیه می‌شد. با این ترتیب از ورودی‌ها، آموزش دو وظیفه‌ی تشخیص هدف و پر کردن جای خالی، به صورت همزمان و اشتراک آن ازطریق مشترک بودن تابع خطای آن دو وظیفه بود.



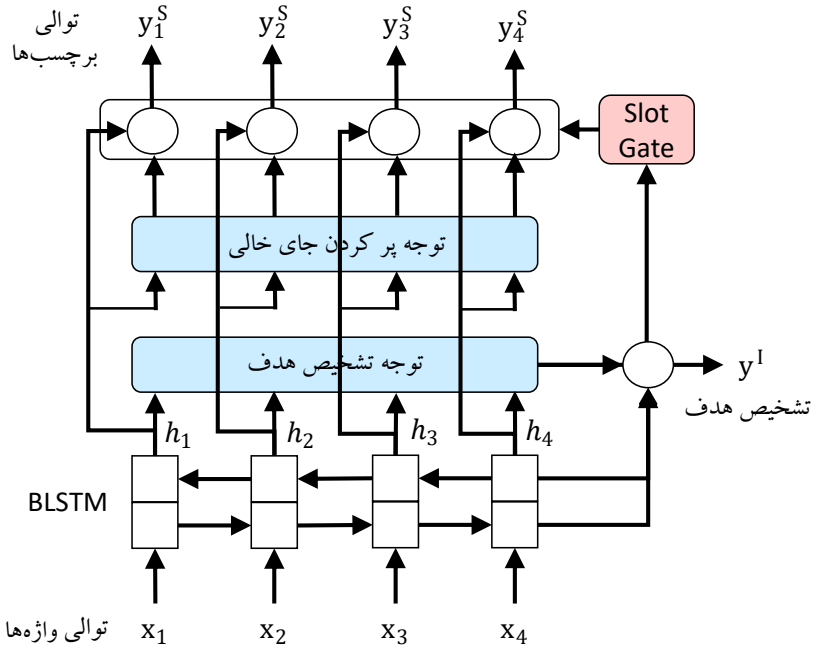
شکل ۳-۱ - ساختار مدل Aligned BLSTM [۱۸].

در پژوهشی دیگر، با ترکیب معماری LSTM تراز شده با شبکه‌ی عصبی کانولوشنی و ساختار توالی ویژگی پنجره، نتایج بهتری به دست آمد [۱۹]. در این شبکه که آن را Aligned CNN-BLSTM نامیدند، در بخش رمزنگار، عملیات کانولوشن با اندازه هسته‌های مختلف صورت گرفته، و نتیجه‌ی حاصل پس از ترانهاده شدن، به یکدیگر الحاق می‌شد. این ساختار را توالی ویژگی پنجره نامیدند. سپس خروجی این ساختار به یک LSTM دوطرفه برای تعبیه داده می‌شد. رمزگشای پیشنهادی این مدل در هر دو وظیفه، با Aligned BLSTM یکسان در نظر گرفته شد. شکل ۳-۲ ساختار این شبکه را نمایش می‌دهد. این ساختار به عنوان پایه‌ی کار CTran درنظر گرفته شد؛ از این رو در بخش ۴، شباهت زیادی میان ساختار رمزنگار پیشنهادی و این مدل خواهید دید. بخش رمزنگار این شبکه نیز کاملا با شبکه‌ی [۱۸] یکسان بود. متعاقباً، اشتراک آموزش دو وظیفه نیز با اشتراک گذاری تابع خطا اتفاق می‌افتاد.



شکل ۳-۲ - ساختار مدل Aligned CNN-BLSTM [۱۹].

اما در مدل Slot-Gated [۲۰]، علاوه بر آموزش مشترک دو وظیفه از طریق اشتراک تابع خطا، با به اشتراک گذاری بردارهای توجه نیز اطلاعات وظیفه‌ی تشخیص هدف را در اختیار وظیفه‌ی پر کردن جای خالی گذاشتند. در معماری Slot-Gated که در شکل ۳-۳ ترسیم شده، از یک شبکه‌ی LSTM دوطرفه برای تعبیه‌ی جمله‌ی ورودی استفاده شد. این LSTM میان هر دو وظیفه مشترکاً و به عنوان رمزنگار مدل به کار گرفته شد. سپس در رمزگشا، مکانیزم توجه [۶] برای هر وظیفه به صورت جداگانه روی آن لایه‌ی LSTM اعمال گردید. برای تولید احتمالات تشخیص هدف، آخرین خروجی LSTM همراه با بردار توجه [۱] جمع شده، سپس به یک لایه‌ی تماماً متصل تغذیه شد و با استفاده از سافت‌مکس، احتمالات خروجی برای تشخیص هدف به دست آمد. اما برای وظیفه‌ی پر کردن جای خالی، یک دروازه‌ی جدید معرفی شد. این دروازه درواقع یک امتیاز بود که در بردار توجه جای خالی ضرب می‌شد. نتیجه‌ی ضرب، با بردار حالت مخفی برای هر مرحله از تولید برچسب جمع می‌شد و سپس بردار نهائی بدست آمده با استفاده از یک لایه‌ی تماما متصل و سافت‌مکس، احتمالات خروجی را تولید می‌کرد. به عبارتی، در معماری مدل Slot-Gated، هیچ شبکه‌ی بازگشتی در رمزگشا استفاده نشد. امتیاز ذکر شده، از جمع وزن‌دار بردار توجه تشخیص هدف، که در بعد زمانی گسترده شده بود، با بردار توجه جای خالی بدست می‌آمد.

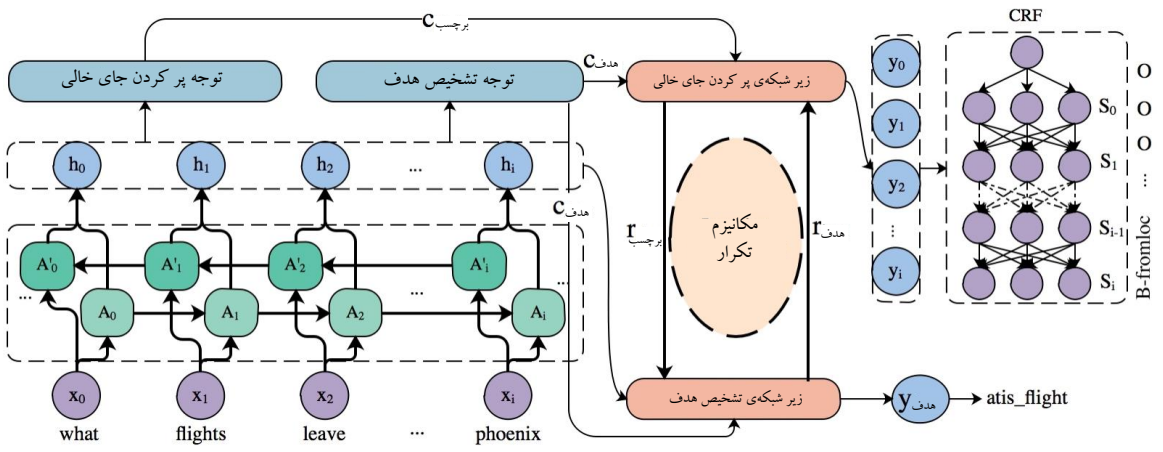


شکل ۳-۳ - ساختار مدل Slot-Gate [۲۰].

در روش Slot-Gate، اطلاعات برچسب‌های تولید شده در تشخیص هدف مورد بهره‌وری واقع نمی‌شد؛ از این رو، ارتباطات مستقیم دوطرفه برقرار نمی‌گردید. مدل Inter-Related [۲۶] که در شکل ۳-۴ ترسیم شده، شبکه‌ی

[۱] لازم به ذکر است که معمولا مکانیزم توجه از یک ماتریس $\mathbb{R}^{i\times j}$ تشکیل شده که در آن i به عنوان طول جمله مبدا و j طول جمله‌ی مقصد است؛ اما چون در تشخیص هدف تنها یک مقصد وجود دارد، یک بردار $\mathbb{R}^{i}$ تولید می‌شود.

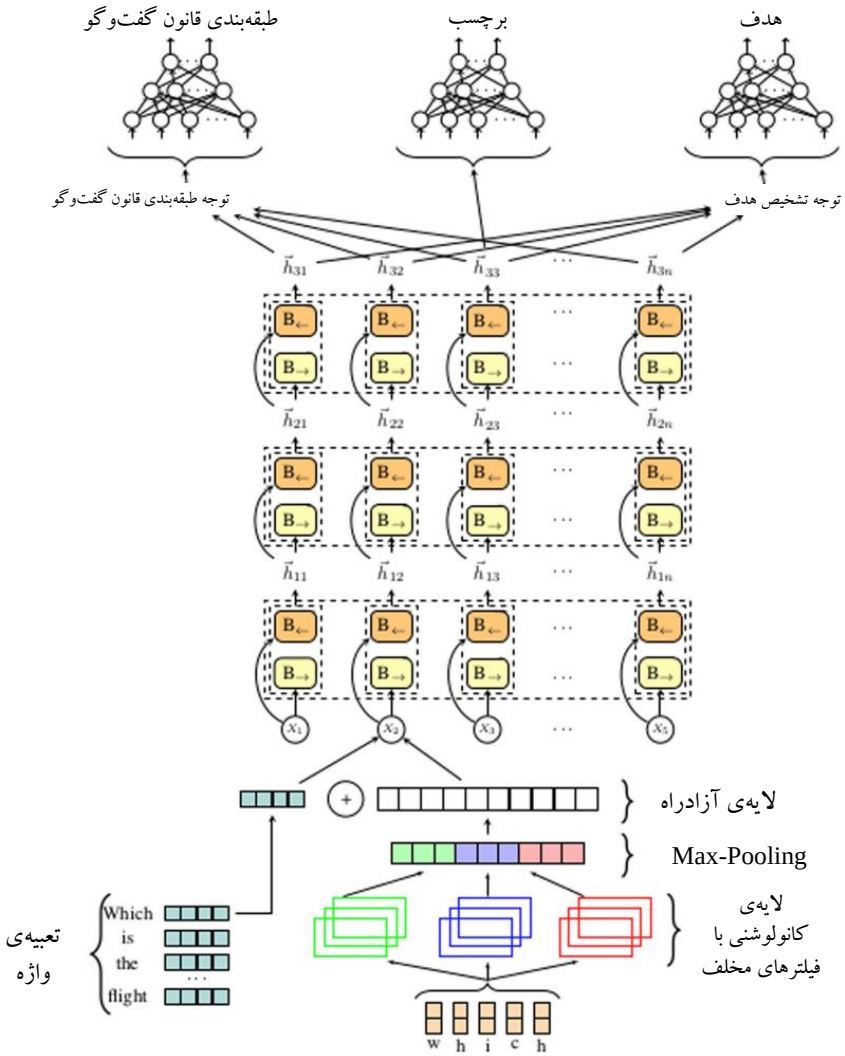
SF-ID را معرفی کرد که از یک زیر شبکه‌ی پر کردن جای خالی و یک زیر شبکه‌ی تشخیص هدف ساخته می‌شد. زیر شبکه‌ی پر کردن جای خالی، اطلاعات تشخیص هدف را وارد پر کردن جای خالی کرده و در مقابل، زیر شبکه‌ی تشخیص هدف، اطلاعات برچسب‌های تولید شده را بر شبکه‌ی تشخیص هدف اعمال می‌کرد. اما برای تولید هرکدام، نیاز به داشتن دیگری است. از این رو، این مدل، مکانیزم تکرار را معرفی کرد که در آن، یک وظیفه به عنوان نقطه‌ی آغاز انتخاب می‌شد، اما فرایند انتخاب تا تعداد دفعه‌ی مشخصی تکرار می‌شد. به این ترتیب اطلاعات معنایی هر دو وظیفه می‌توانستند با یکدیگر ترکیب شوند. ازنظر ساختار شبکه‌ای، این مدل در رمزنگار خود یک LSTM دوطرفه را به صورت مشترک برای دو وظیفه استفاده می‌کند. سپس در این مدل، مکانیزم توجه جداگانه برای هر وظیفه تعریف می‌شود. در رمزگشای آن، مکانیزم تکرار ابداعی قرار گرفته است. در پایان، برای تولید احتمالات هدف از لایه‌ی تماماً متصل، و برای تولید برچسب از CRF استفاده گردیده است.



شکل ۳-۴ - ساختار مدل Inter-Related [۲۶].

در CharEmbed+GRU [۲۷] تعبیه‌ی سطح الفبا برای درک زبان طبیعی پیشنهاد شد. به این منظور، ابتدا کلمات را به الفبای سازنده شکسته، و حروف الفبای واژه را به یک LSTM دوطرفه دادند تا تعبیه‌ی سطح الفبای دوطرفه ایجاد شود. در پایان، از الحاق تعبیه‌ی ایجاد شده از LSTM با یک تعبیه‌ی سطح واژه مانند Word2Vec، تعبیه‌ی نهائی واژه را ساختند. در معماری این مدل، از LSTM دو طرفه برای رمزنگار و از CRF برای رمزگشا استفاده شد. شکل ۳-۵ نشان دهنده‌ی معماری CharEmbed+GRU است که شیوه‌ی جدیدی برای ایجاد تعبیه‌ی واژه‌ها ایجاد کرد [۲۷]. این مدل با قراردادن شبکه‌ی کانولوشنی بر روی الفبای کلمات و سپس عملیات Max-Pooling، یک تعبیه‌ی سطح الفبا برای واژه ایجاد کرد. سپس تعبیه‌ی سطح الفبا را با تعبیه‌ی واژه‌ی مورد پردازش، الحاق کرد. حاصل این الحاق، به عنوان تعبیه‌ی یک واژه در نظر گرفته شد. این تعبیه برای تمام واژه‌های جمله ایجاد شده و وارد پشته‌ی LSTM گردید

تا یک تعبیه برای جمله ایجاد شود. سپس با استفاده از مکانیزم توجه [۶] و تغذیه‌ی آن به یک لایه‌ی خطی و سافت مکس، احتمالات خروجی تشخیص هدف، پر کردن جای خالی و طبقه بندی قانون گفتگو[۱] تولید شد.

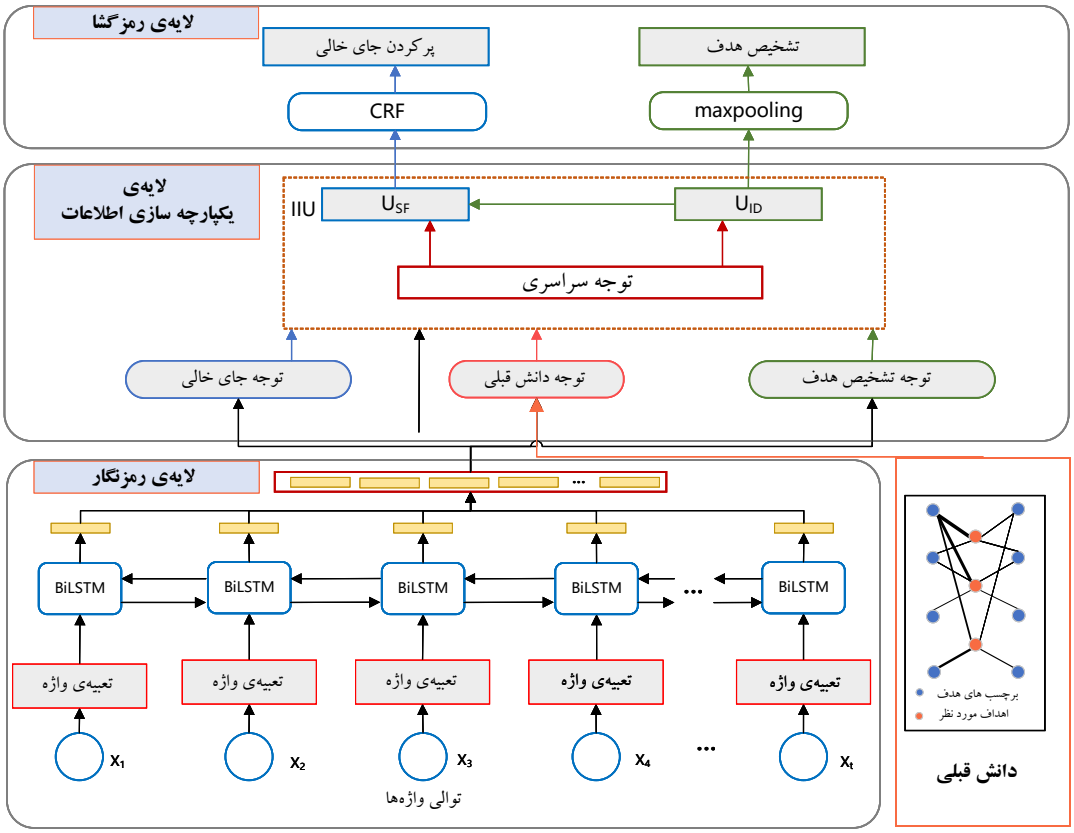


شکل ۳-۵ - ساختار مدل پیشنهادی CharEmbed+GRU [۲۷].

مدل PKJL[۲] برای افزایش به اشتراک گذاری اطلاعات قبلی با دو وظیفه‌ی پر کردن جای خالی و تشخیص هدف معرفی شد [۲۸]. منظور از اطلاعات قبلی، احتمال وقوع هر برچسب هدف با هر برچسب جای خالی در یک نمونه‌ی آموزشی است. در معماری پیشنهادی PKJL که در شکل ۳-۶ ترسیم شده، مدل PKJL با محاسبه‌ی این احتمالات و تغذیه‌ی آن به مدل خود در یک لایه‌ی ابداعی، دانش قبلی را با مدل به اشتراک گذاشت. معماری این مدل که در شکل ۳-۶ به تصویر کشیده شده است، از یک LSTM دو طرفه در رمزنگار و بردار احتمالات وقوع هر برچسب با هر هدف تشکیل می‌شد. یک ماژول ابداعی به نام لایه‌ی یکپارچه سازی اطلاعات بعد از رمزنگار ایجاد شد که سه ماتریس توجه

[1] Dialogue Act Classification
[2] Prior Knowledge Joint Learning

برای تشخیص هدف، پر کردن جای خالی و دانش قبلی در آن تعریف می‌شدند. این سه ماتریس با یکدیگر تجمیع و تبدیل خطی شدند تا خروجی ماژول ابداعی تکمیل شود. در رمزگشای پر کردن جای خالی، یک CRF و در رمزنگار تشخیص هدف، یک لایه‌ی خطی همراه با سافت‌مکس استفاده شد.

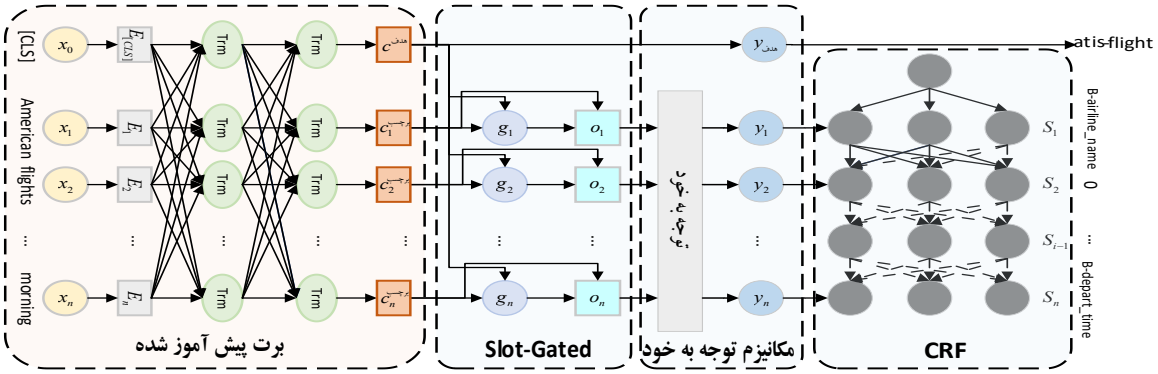


شکل ۳-۶ – ساختار مدل PKJL [۲۸].

۳-۳ مبتنی بر مدل زبانی به عنوان رمزنگار

طراحی مدل زبانی برت به نحوی است که استفاده از آن در وظایف مختلف پردازش زبان طبیعی بسیار ساده است. در مدل پایه که توسط [۲۹] معرفی شد، یک لایه‌ی تماماً متصل بر روی هریک از خروجی‌های برت قرار گرفت تا خروجی آن که به بعد آن، همان ابعاد تعبیه‌ی است، تبدیل به تعداد کلاس‌های برچسب‌ها و هدف‌های مجموعه داده شود. بدین ترتیب، خروجی نشانه‌ی $[CLS]$ برای مشخص کردن هدف کاربر، و نشانه‌های بخش اول ورودی به عنوان نتیجه‌ی برچسب‌زنی در نظر گرفته شد. بدین ترتیب، رمزنگار شبکه خود مدل برت و رمزگشا یک لایه‌ی تماماً متصل در نظر گرفته شد. در معماری [۲۹]، از برت به عنوان رمزنگار استفاده شد. اما برت در درک روابط منطقی میان برچسب‌های هدف، خوب عمل نمی‌کند و این مشکل باعث می‌شود عملکرد پر کردن جای خالی شدیداً تحت تاثیر قرار گیرد [۳۰].

از این رو، مدل SASGBC [۳۰] معرفی شد، که حاوی دو راهکار برای مقابله با مشکل ذکر شده بود. این دو راهکار مکانیزم Slot-Gate و رمزگشای CRF بودند که هر دو مختص رمزگشای پر کردن جای خالی به کار گرفته شدند. مطابق شکل ۳-۷ در رمزگشای پر کردن جای خالی، نخست مکانیزم Slot-Gate برای آمیختن هدف پیش بینی شده، با خروجی برت پیشنهاد شد. در مکانیزم Slot-Gate پیشنهادی، خروجی تعبیه‌ی برت برای هر واژه، با بردار خروجی نشانه‌ی $[CLS]$ الحاق شده، و سپس با استفاده از یک لایه‌ی خطی بدون بایاس، ابعاد آن کاهش می‌یابد. دوم، در آخرین لایه از رمزگشا، از CRF برای پیش‌بینی احتمالات برچسب‌ها و برای تولید برچسب‌های خروجی از الگوریتم Viterbi استفاده شد. برای رمزگشای تشخیص هدف، صرفا یک لایه‌ی خطی همراه با سافت‌مکس، احتمالات خروجی را تعیین می‌کرد.

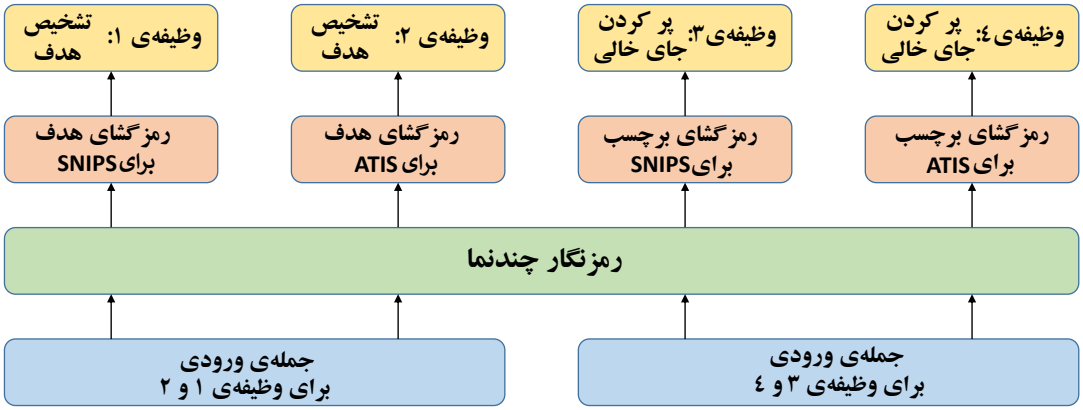


شکل ۳-۷ - ساختار مدل SASGBC [۳۰].

۳-۴ مبتنی بر مدل زبانی به عنوان تعبیه‌گر

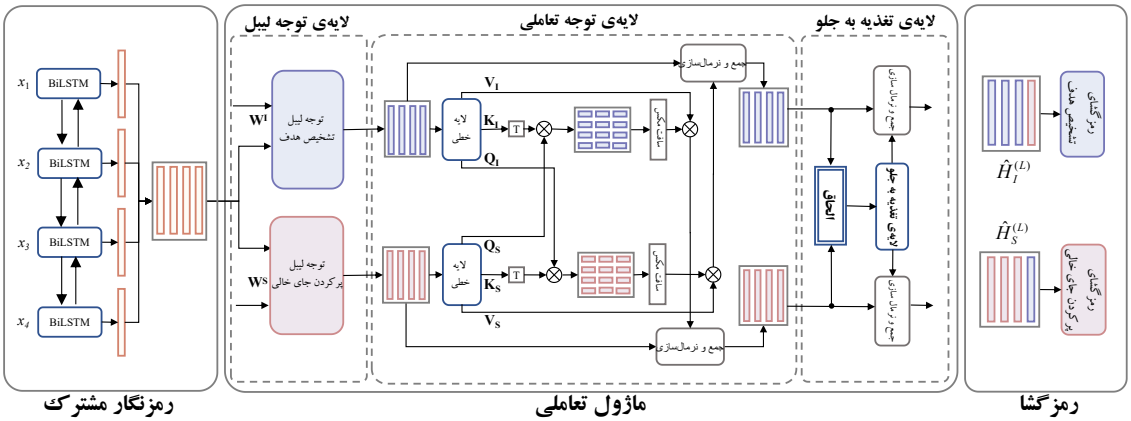
از طرفی، Federated-Learning [۳۱] رمزنگار چند نما را معرفی کرد. در این رمزنگار، یک رمزنگار برای تعبیه‌ی موقعیت، یک رمزنگار برای تعبیه‌ی اطلاعات محلی، یک تعبیه برای اطلاعات کلی (در سطح جمله) و یک رمزنگار اطلاعات توالی (سری زمانی) استفاده شد. از طرف دیگر، در برای آموزش مدل، یادگیری فدرال معرفی شد. مطابق شکل ۳-۸، در این شیوه‌ی یادگیری، یک رمزنگار بین دو مجموعه داده‌ی ATIS و SNIPS مشترک بود، به این ترتیب، برای هر مجموعه داده، به صورت جدا، رمزگشای تشخیص هدف و پر کردن جای خالی تعریف شد. در نهایت، مدل باید پارامترهای رمزنگار را میان ۴ وظیفه‌ی مختلف (دو وظیفه برای هر مجموعه داده) به اشتراک می‌گذاشت. نکته‌ی پایانی در مورد مدل Federated-Learning، استفاده‌ی آن از برت به عنوان تعبیه‌ی کلمات بود.

معماری Co-Interactive Transformer [۳۲] برای ادغام پیش زمینه‌ی تشخیص هدف و پر کردن جای خالی در یکدیگر معرفی شد و از مکانیزم توجه چند سر [۱۰] استفاده کرد. در Co-Interactive به جای معماری رمزنگار - رمزگشا، معماری رمزنگار - ماژول تعاملی - رمزگشا معرفی شد. در رمزنگار، از یک LSTM دوطرفه استفاده



شکل ۳-۸ - ساختار مدل Federated-Learning [۳۱].

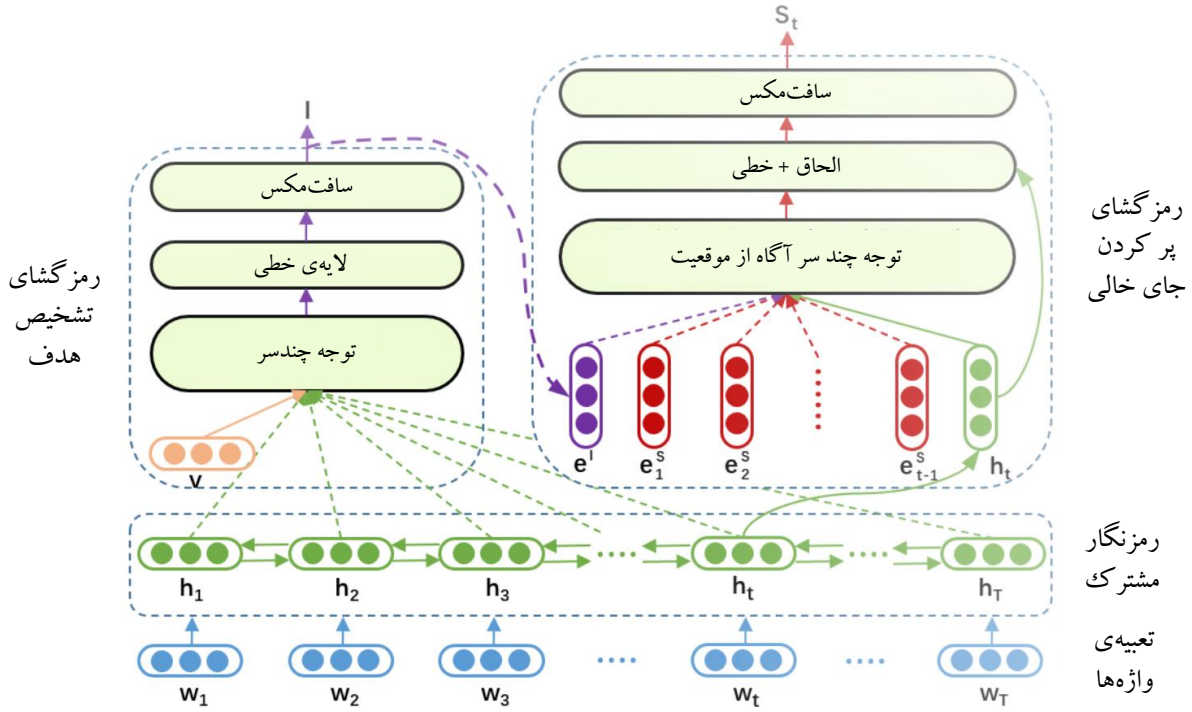
شد. سپس با استفاده از مکانیزم توجه [۶]، برای تشخیص هدف و پر کردن جای خالی، به صورت جداگانه ماتریس توجه به دست آمد. در ماژول تعاملی پیشنهادی، از مکانیزم توجه چند سر [۱۰] استفاده شد؛ به این نحو که به ازای هر وظیفه‌ی، یک تبدیل خطی از ماتریس توجه آن وظیفه برای کلید، مقدار و پرسش ایجاد شد. سپس، ماتریس توجه برای هر وظیفه، با استفاده از کلید و مقدار همان وظیفه و پرسش وظیفه‌ی دیگر، محاسبه شد. بدین ترتیب در این ماژول، اطلاعات ماتریس توجه مربوط به هر دو وظیفه با یکدیگر ترکیب شد. در گام بعدی، مانند معماری ترنسفورمر، ماتریس ورودی با خروجی مکانیزم توجه توصیفی برای هر وظیفه، جمع و نرمالایز شد. در لایه‌ی تغذیه به جلوی ماژول تعاملی، خروجی هرکدام از داده‌های نرمالایز شده مربوط به وظایف با یکدیگر الحاق شده و به یک لایه‌ی تغذیه به جلو داده می‌شود. حاصل خروجی این لایه‌ی تغذیه به جلو با خروجی لایه‌ی توجه تعاملی، جمع و نرمالایز شد. در پایان، برای رمزگشای پر کردن جای خالی از CRF، و برای رمزگشای تشخیص هدف از Max-Pooling استفاده شد.



شکل ۳-۹ - ساختار مدل Co-Interactive Transformer [۳۲].

در تلاشی دیگر برای ادغام بهینه‌ی دو وظیفه‌ی تشخیص هدف و پر کردن جای خالی، AISE [۳۳] معرفی شد.

معماری AISE[1] که در شکل ۳-۱۰ به تصویر کشیده شده، متشکل از یک رمزنگار مشترک، یک رمزنگار تشخیص هدف و مکانیزم پیشنهادی توجه چند سر پوشیده‌ی آگاه از موقعیت بود. مدل AISE بر مبنای LSTM و مکانیزم توجه چند سر بنا شد. در رمزگشای تشخیص هدف، از مکانیزم ادغام توجه چند سر استفاده کردند. ادغام توجه چند سر، نوعی از ادغام میانگین است که در آن وزن‌ها در میانگین گیری قابل تنظیم است. در این مکانیزم، کلید و مقدار از تعبیه‌ی رمزنگار و پرسش یک بردار، با مقادیر اولیه تصادفی و قابل آموزش است. خروجی این مکانیزم نیز، یک بردار با بعد تعبیه‌ی ورودی می‌باشد. در AISE، برای رمزگشای پر کردن جای خالی، مکانیزم چند سر پوشیده‌ی آگاه از موقعیت معرفی شد. در مکانیزم پیشنهادی، از توجه چند سر پوشیده و برای تعبیه‌سازی ورودی نیز از تعبیه‌ی نسبی [۳۴] استفاده شد. با توجه به اینکه خروجی‌های رمزگشای AISE به صورت ترتیبی تولید شدند، در مکانیزم چند سر پوشیده، از تعبیه‌ی رمزنگار در موقعیت مربوط به نشانه‌ی درحال تولید به عنوان بردار پرسش استفاده می‌شد. همچنین از الحاق بردار زمینه‌ی تشخیص هدف و ماتریس تعبیه‌ی ابتدایی جمله به عنوان کلید و مقدار استفاده شد. در پایان، برچسب تولیدی در هر موقعیت، با استفاده از الحاق و سافت‌مکس بردار خروجی توجه چند سر پوشیده با بردار تعبیه‌ی مربوط به آن نشانه به دست آمد.



شکل ۳-۱۰ - ساختار مدل AISE [۳۳].

همچنین مدل پیشنهادی در [۳۵] از المو به عنوان تعبیه‌ی اولیه، LSTM دو طرفه به عنوان رمزنگار و CRF به عنوان رمزگشا استفاده کرد. در این مقاله، به ارائه یک مدل زبانی تنظیم شده برای وظیفه‌ی برچسب زنی پرداخته شد و چند

[1] Attending to Intent and Slots Explicitly

استراتژی برای سبک کردن مدل زبانی المو ارائه شد.

۳-۵ جمع‌بندی

در این فصل، انواع مدل‌های پیشین برای وظیفه‌ی درک زبان طبیعی ارائه شدند. از میان مدل‌های مبتنی بر تعبیه ثابت، Aligned LSTM [۱۸] با مکانیزم توجه، نخستین مدلی بود که اهمیت تراز بودن ورودی و خروجی را بیان کرد. سپس، مدل Aligned CNN-BLSTM [۱۹] از ساختار لایه‌ی کانولوشنی-توالی ویژگی پنجره برای تعبیه‌ی ورودی خود استفاده کرد. مدل Slot-Gate [۲۰] آموزش ضمنی دو وظیفه با یک رمزنگار مشترک را کافی نمی‌دانست؛ از این رو مکانیزمی برای کارگزاری هدف تشخیص داده شده در ورودی رمزگشای پر کردن جای خالی ارائه کرد. مدل Inter-Related [۲۶] این مکانیزم را یک طرفه دانست و مکانیزمی دو طرفه برای دخالت هر دو رمزگشا در یکدیگر ارائه کرد. همچنین، مدل PKJL [۲۸] برای دخالت دانش پیش‌زمینه در رمزگشا و مدل CharEmbed+GRU [۲۷] برای ارائه یک تعبیه‌ی جدید معرفی گشتند. از طرف دیگر، در میان کارهایی که از مدل زبانی به عنوان رمزنگار استفاده کردند، برت و CRF [۲۹] برای تولید برچسب استفاده شد. همچنین، SASGBC [۳۰] مکانیزم Slot-Gate را پس از برت استفاده کرد. برخی از معماری‌های ارائه شده نیز مدل زبانی را به عنوان تعبیه‌ی کلمات استفاده کردند. در این میان، Federated Learning [۳۱] رمزنگار چند نما و همچنین آموزش همزمان رمزنگار بر روی چند مجموعه داده را معرفی کرد. معماری Co-Interactive Transformer [۳۲] روشی جدید برای ادغام پیش زمینه‌ی تشخیص هدف و پر کردن جای خالی با استفاده از به اشتراک گذاری کلید و مقدار معرفی کرد. در پایان نیز، مدل AISE [۳۳] شیوه‌ای جدید برای تولید برچسب، با استفاده از توجه چند سر آگاه از موقعیت ارائه کرد.

فصل چهارم
مدل پیشنهادی

۴-۱ مقدمه

در فصل گذشته، روش‌های مطرح درک زبان طبیعی معرفی شده و به بررسی هرکدام از روش‌ها پرداخته شد. از میان این روش‌ها، مدل Aligned CNN-BLSTM [۱۹] به عنوان نقطه‌ی شروع این پایان نامه انتخاب شد. علت این امر، عملکرد مناسب این مدل در وظیفه‌ی پر کردن جای خالی، با وجود عدم استفاده از مدل‌های زبانی بود. به نظر می‌رسید مدل مذکور، با ایجاد تغییرات و استفاده از معماری‌های امروزی، بتواند عملکرد بهتری از خود نشان دهد. از طرف دیگر، [۱۹] در معماری خود از شیوه‌ی تراز کردن رمزگشا برای تولید تگ استفاده می‌کرد. شیوه‌ی تراز کردن LSTM با رمزگشای ترنسفورمر سازگار نیست؛ از این رو جای خالی یک رمزگشای ترنسفورمر تراز شده نیز حس می‌شد. به همین دلیل، در این پایان‌نامه، مدل CTran[1] با تکیه بر شبکه‌ی کانولوشنی و ترنسفورمرها طراحی شد. در ادامه، ابتدا ساختار کلی معماری پیشنهادی را نمایش داده و سپس به تفصیل، اجزای سازنده‌ی آن توصیف می‌شود.

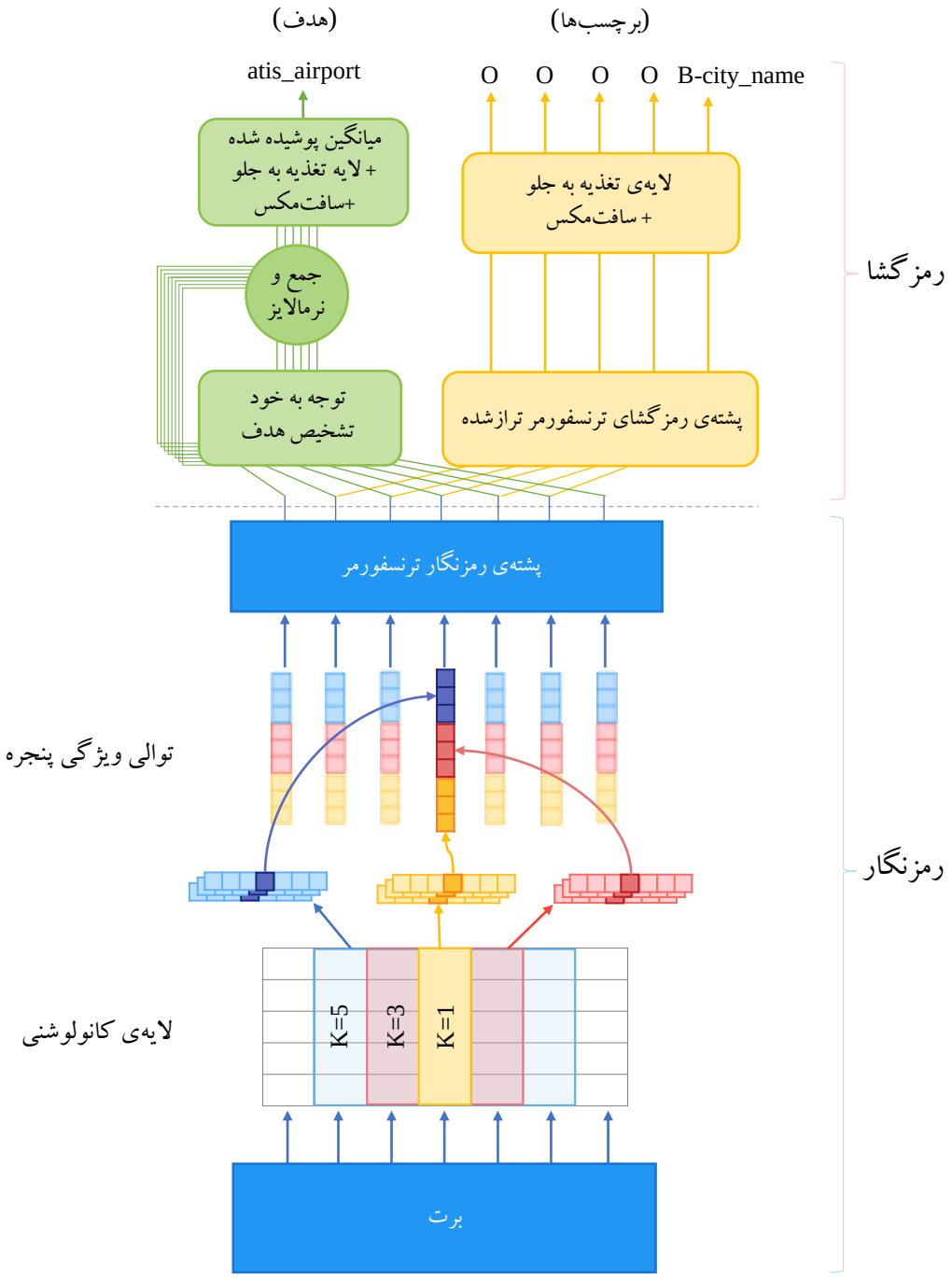
۴-۲ طرح مدل

معماری CTran از یک رمزنگار مشترک، یک رمزگشای تشخیص هدف و یک رمزگشای پر کردن جای خالی تشکیل می‌شود. ابتدا جمله‌ی کاربر به عنوان ورودی، وارد رمزنگار شده و تعبیه‌ی میانی تولید می‌شود. این تعبیه‌ی میانی، به صورت همزمان وارد دو رمزگشا می‌شود. وظیفه‌ی رمزگشا در معماری، تولید خروجی‌های مدل است. هر رمزگشا، معماری منحصر به فردی دارد که برای وظیفه‌ی مورد نظرش طراحی گردیده است. برای اصلاح وزن‌های شبکه و آموزش مدل،خروجی‌های تولید شده با خروجی‌های مورد انتظار مقایسه شده و خطا محاسبه می‌شود. در مرحله‌ی بعد، وزن‌های هر رمزگشا به صورت جداگانه اصلاح می‌شود؛ اما از آن جا که رمزنگار بین دو وظیفه مشترک است، خطاهای هر دو وظیفه روی وزن‌های آن اثر می‌گذارند. در این بخش، ابتدا به معماری رمزنگار، سپس معماری رمزگشای تشخیص هدف و در پایان به معماری رمزگشای پر کردن جای خالی پرداخته می‌شود.

۴-۳ رمزنگار

همانطور که گفته شد، CTran از رمزنگار مشترک بهره می‌برد. رمزنگار شامل یک مدل زبانی پیش‌آموز شده، شبکه‌ی عصبی کانولوشنی، ساختار توالی ویژگی پنجره و پشته‌ی رمزنگار ترنسفورمر می‌شود. مطابق شکل ۴-۱ در گام نخست، نشانه‌ها وارد مدل زبانی می‌شود تا تعبیه‌ای آگاه از پیش زمینه به دست آید. این تعبیه، به عنوان نقطه‌ی شروعی برای رمزنگار است. در گام بعد، این تعبیه‌ها وارد لایه‌ی کانولوشنی با اندازه هسته‌های مختلف می‌شود. استفاده از عملیات Max-Pooling بعد از کانولوشن رایج است، اما با هدف حفظ توالی واژه‌ها، از ساختاری به نام توالی ویژگی پنجره

[1] CNN-**Tran**sformer



شکل ۴-۱ – معماری مدل پیشنهادی CTran. دو نشانه‌ی اضافه‌شده $[CLS]$ و $[SEP]$ از نشانه‌های ساختاری برت هستند. در زمان ورود تعبیه‌ی ایجاد شده به رمزگشای پر کردن جای خالی، تعبیه‌های متناظر با این دو نشانه، دور ریخته می‌شوند.

استفاده شده است. در این ساختار، تمام بردارهای تولید شده برای یک واژه، ترانهاده شده و به یکدیگر الحاق می‌شوند. آخرین بخش رمزنگار، پشته‌ی رمزنگار ترنسفورمر است، که برای ایجاد تعبیه‌ی جدیدی که حاصل توالی ویژگی پنجره است، استفاده شده است. در ادامه، اجزاء سازنده‌ی رمزنگار تشریح می‌شوند.

۴-۳-۱ مدل زبانی پیش‌آموز شده

اثبات شده که مدل‌های زبانی پیش‌آموز شده مانند برت و المو، برای بهبود عملکرد وظیفه‌های پردازش زبان طبیعی مفید هستند [۳۶]. در این پژوهش، هر دو مدل زبانی یاد شده در معماری پیشنهادی استفاده شده و نتایج آن‌ها مقایسه شدند. ورودی برت جمعی تعبیه‌ی نشانه‌های WordPiece [۳۷]، تعبیه‌ی موقعیتی و تعبیه‌ی بخش[1] می‌باشد. تعبیه‌ی بخش در مورد استفاده‌ی تک جمله‌ای معنا ندارد، چرا که تنها یک جمله وارد برت می‌شود. برای نشانه‌های ورودی $X = \left[x_1, x_2, x_3, ..., x_L\right]$، برت یک تعبیه‌ی آگاه به زمینه $H = \left[h_1, h_2, h_3, ..., h_L\right]$ برای مدل فراهم می‌کند. توجه داشته باشید که نشانه ساز[2] WordPiece ممکن است واژه‌ها را به چند نشانه بشکند. این امر رابطه‌ی یک به یک واژه‌ی ورودی با برچسب را برهم میزند. همچنین، برچسب زدن نشانه‌های شکسته شده بیهوده است؛ چراکه همه‌ی آن‌ها به یک واژه تعلق داشته و برچسب یکسانی دارند. علاوه بر این، WordPiece واژه‌های دارای علائم را نیز به چند نشانه می‌شکند. در روش پیشنهادی، این مشکل با حذف علائم درون واژه‌ها برطرف گردید. برای جلوگیری از حذف واژه‌هایی که فقط دارای علائم بوده و فاقد عدد و الفبا هستند، استثنا ایجاد گردید و علائم درون این واژه‌ها حذف نشدند. همچنین، همانطور که در بخش دو گفته شد، ورودی برت دارای نشانه‌های خاص است. این نشانه‌ها برای وظیفه‌ی پر کردن جای خالی در نظر گرفته نمی‌شوند.

برخلاف برت، نشانه ساز زبانی المو واژه‌ها را نمی‌شکند. از این رو ترتیب ورودی و خروجی یکسان است. در معماری پیشنهادی، از تعبیه‌های لایه‌ی آخر المو که حاوی تعبیه معنایی واژه‌ها است بهره برده شد.

۴-۳-۲ شبکه‌ی کانولوشنی

در [۳۸،۱۹]، شبکه‌ی عصبی کانولوشنی برای پوشش کاستی‌های LSTM استفاده شد. از آنجا که ترنسفورمر کاستی‌های یاد شده را برطرف کرده است، ممکن است استفاده از شبکه‌ی کانولوشنی بیهوده به نظر برسد. اما از طرفی، اثر استفاده از شبکه‌ی کانولوشنی در کنار ترنسفورمر برای وظیفه‌ی درک زبان طبیعی بررسی نشده است. در این پژوهش، از شبکه‌ی کانولوشنی، برای استخراج اطلاعات معنایی و همچنین ادغام آن‌ها در پنجره‌های مشخص استفاده شده است. این ویژگی از آن سو مورد توجه این پژوهش قرار گرفت، که ترنسفورمر، ذاتاً تفاوتی میان واژه‌های نزدیک به واژه‌ی اصلی و واژه‌های دورتر قائل نیست. این در حالی است که فرض اولیه‌ی مدل‌هایی مانند n-gram [۳۹]، GloVe و Word2Vec، بر نقش کلیدی واژه‌های مجاور است. به همین علت، در معماری پیشنهادی، از چند لایه‌ی کانولوشنی، با اندازه هسته و مقادیر اولیه‌ی متفاوت استفاده شده، تا تعبیه‌ی واژه‌های همسایه با واژه‌ی مورد نظر آمیخته شود. این امر به منظور تاکید اثرگذاری واژه‌های همسایه بر معنای واژه‌ی فعلی است؛ امری که در LSTM برای پوشش نقش دروازه‌ی

[1] Segment Embedding
[2] Tokenizer

ورودی صورت می گرفت.

اگر d_{lm} نمایانگر بُعد تعبیه و جمله ورودی حاوی L نشانه باشد، تعبیه‌ی جمله‌ی ورودی به صورت $h \in \mathbb{R}^{L \times d_{lm}}$ تعریف می‌شود و بردار تعبیه برای i–امین نشانه با $h_i \in \mathbb{R}^{d_{lm}}$ نشان داده می‌شود. اجازه دهید k اندازه‌ی هسته باشد و $f \in \mathbb{R}^{k \times d_{lm}}$ یک هسته را نشان دهد. با این فرضیات، برای هر موقعیت i در جمله، یک پنجره $w_i \in \mathbb{R}^{k \times d_{lm}}$ وجود دارد که حاوی k بردار نشانه بوده و به صورت $w_i = \left[h_i, h_{i+1}, h_{i+2} \cdot h_{i+k-1}\right]$ تعریف می‌شود. در صورتی که $\odot$ ضرب عنصری، $b \in \mathbb{R}$ پارامتر بایاس و A یک تابع فعال ساز باشد، عملیات کانولوشن c_i برای هر پنجره w_i توسط معادله‌ی ۴–۱ محاسبه می‌شود. مدنظر داشته باشید که بعد از ضرب عنصری، تمام مقادیر ماتریس با یکدیگر جمع می‌شوند تا یک عدد به دست آید.

$$c_i = A\left(w_i \odot f + b\right) \tag{۴–۱}$$

این عملیات به ازاء $0 < i < k$، یعنی تمام موقعیت‌های درون جمله انجام می شود. بر همین اساس، خروجی کانولوشن f بر روی x با $C = \left[c_1, c_2, c_3, ..., c_i, ..., c_L\right]$ نمایش داده می‌شود. لازم به ذکر است که در CTran از پد صفر استفاده گردید؛ به همین دلیل اندازه‌ی ورودی و خروجی لایه‌ی کانولوشنی همواره برابر است. پد صفر برای حفظ برابر طول جمله‌ی خروجی با ورودی استفاده شد؛ چرا که در گام بعد، این بردارها با یکدیگر الحاق می‌شوند. از این رو، برابر بودن اندازه‌ی ورودی و خروجی اهمیت بالایی دارد.

۴–۳–۳ توالی ویژگی پنجره

استفاده از عملیات Pooling بعد از لایه‌ی کانولوشنی مرسوم است، اما نمونه بَرداری ناپیوسته[1] باعث از دست رفتن برخی از اطلاعات می‌شود، و در نتیجه، این عملیات‌ها باعث از بین رفتن اطلاعات ترتیبی، در توالی داده‌ها می‌شود [۱۹]. در این پژوهش، به منظور حفظ ترتیب اصلی نشانه‌ها، از توالی ویژگی پنجره استفاده شد. در این ساختار، تعبیه‌ی نشانه‌ها با اندازه‌ی هسته‌ی مختلف و تعداد کانال خروجی مختلف، به یکدیگر الحاق می‌شوند تا یک بردار تعبیه برای آن نشانه تشکیل دهند.

فرض کنید d_{conv} تعداد کل هسته‌ها، با اندازه‌ی هسته و مقادیر اولیه‌ی متفاوت برای هسته و j بیانگر موقعیت j–امین فیلتر باشد. با در نظر گرفتن $\oplus$ به عنوان عملیات الحاق، $r_i \in \mathbb{R}^{d_{conv}}$ تعریف شده در معادله ۴–۲، نمایش نهائی واژه‌ی i–ام را تعریف می‌کند و C_i^j نشان دهنده i–امین عنصر فیلتر j–ام است.

$$r_i = C_i^1 \oplus C_i^2 \oplus C_i^3 \oplus ... \oplus C_i^j \oplus ... \oplus C_i^{d_{conv}} \tag{۴–۲}$$

[1] Discontinuous Sampling

بنابراین r_i تمام مقادیر کانولوشن متعلق به موقعیت i را برای همه هسته‌ها به هم الحاق می‌کند. این عملیات برای همه موقعیت‌ها انجام می‌شود تا زمانی که این نمایش برای همه واژه‌های ورودی تولید شود. خروجی ساختار توالی ویژگی پنجره، به شکل $R = \{r_1, r_2, r_3, \cdots, r_L\}$ می‌باشد. در مرحله‌ی آخر، خروجی این ساختار به پشته‌ی رمزنگار ترنسفورمر تغذیه می‌شود تا نمایش نهائی تولید گردد.

۴-۳-۴ پشته‌ی رمزنگار ترنسفورمر

امکان دسترسی مستقیم به تمام موقعیت‌های درون جمله، به رمزنگار ترنسفورمر کمک می‌کند تا بر از دست دادن اطلاعات غلبه کند. برتری آشکار ترنسفورمر باعث شد که از این شبکه، به جای LSTM استفاده شود. پشته‌ی رمزگشای ترنسفورمر، همانطور که در شکل ۲-۱۲ نشان داده شد، از چندین لایه تشکیل شده است که در آن، خروجی هر لایه به عنوان ورودی لایه بعدی استفاده می‌شود. با توجه به ساختار رمزنگار ترنسفورمر، ابعاد خروجی آن برابر با ورودی است؛ یعنی خروجی این لایه برابر با $H \in \mathbb{R}^{L \times d_{conv}}$ می‌باشد.

۴-۴ رمزگشا

ماتریس ایجاد شده در رمزنگار H، که در این پایان‌نامه حافظه نامیده شده، به صورت همزمان به دو رمزگشا تغذیه می‌شود. این ماتریس به عنوان نقطه‌ی آغاز شبکه‌های رمزگشا است. هرکدام از این دو رمزگشا، تغییرات متناسب با ساختار احتمالات خروجی را، بر روی ماتریس ایجاد می‌کنند. در ادامه، شبکه‌های پیشنهادی برای رمزگشای تشخیص هدف و رمزگشای پر کردن جای خالی معرفی می‌شوند.

۴-۴-۱ رمزگشای تشخیص هدف

همانطور که در شکل ۴-۱ ترسیم شده، رمزگشای تشخیص هدف از یک لایه‌ی توجه به خود با اتصال باقی‌مانده و لایه‌ی نرمال سازی تشکیل شده است. معادله‌ی ۴-۳ بیانگر عملیات یاد شده است.

$$D_{intent} = H + LayerNorm(MultiHead(H)) \tag{۴-۳}$$

در ادامه، یک لایه‌ی تغذیه به جلو و سافت‌مکس برای محاسبه‌ی احتمالات خروجی مورد بهره‌وری قرار گرفته است. بر این اساس، ابتدا میانگین تعبیه‌ی نشانه‌های درون جمله محاسبه می‌شود. در محاسبه‌ی این میانگین، تعبیه‌ی نشانه‌هایی که برای پد کردن جمله اضافه شده‌اند نادیده گرفته می‌شود؛ بدین جهت، این میانگین، پوشیده نامیده شده است. بعد از اعمال میانگین پوشیده، طول جمله L در خروجی مرحله قبل $D_{intent} \in \mathbb{R}^{L \times V}$، فرو می‌ریزد و این ماتریس تبدیل به یک بردار $D'_{intent} \in \mathbb{R}^{1 \times V}$ می‌شود. با در نظر گرفتن محصول نقطه‌ای با علامت ۰، تعداد اهداف یکتای

مجموعه داده با I، وزن‌ها $w \in \mathbb{R}^{V \times I}$ و بایاس $b \in \mathbb{R}^{1 \times I}$ که هردو مورد اخیر قابل آموزش هستند، احتمالات خروجی $P_{intent} \in \mathbb{I}^{1 \times I}$ [۱] برای وظیفه‌ی تشخیص هدف، با معادله‌ی ۴-۴ محاسبه می‌شود.

$$P_{intent} = softmax(w \cdot D'_{intent} + b) \quad (۴-۴)$$

۴-۴-۲ رمزگشای پر کردن جای خالی

رمزگشای پیشنهادی برای پر کردن جای خالی از دو قسمت تشکیل شده است؛ یک رمزگشای ترنسفورمر تراز شده و یک لایه‌ی تغذیه به جلو برای کاهش ابعاد خروجی به تعداد برچسب‌های یکتای مجموعه داده.

از آنجایی که واژه‌های ورودی با برچسب‌های هدف رابطه‌ی یک به یک دارند، بهتر است رمزگشا تراز شده باشد. تراز بودن به این معناست که مدل از نشانه، یا موقعیتی که درحال حاضر برچسب آن را تولید می‌کند، آگاه باشد. این آگاهی باید صریح باشد؛ یعنی مدل مستقیماً به بردار تعبیه‌ی آن نشانه دسترسی داشته باشد. این آگاهی به معنای دسترسی مستقیم به تعبیه‌ی آن نشانه است. تا به حال، روشی برای تراز کردن رمزگشای ترنسفورمر پیشنهاد نشده است؛ از این رو، در این پژوهش، رمزگشای ترنسفورمر تراز شده ابداع می‌شود. ترازی مورد نظر، در بخش توجه متقابل صورت می‌گیرد؛ جایی که کلید و مقدار از حافظه تولید می‌شود. برای پوشاندن حافظه، یک ماسک با اندازه‌ی (S, T) نیاز است، که S نشان دهنده‌ی طول ترتیب مبدا و T نشان دهنده‌ی طول ترتیب مقصد است. از آنجا که طول مبدا و مقصد در وظیفه‌ی برچسب زنی ترتیب یکسان است، ماسک مورد استفاده‌ی به شکل (S, S) تعریف می‌شود. در این پژوهش، رمزنگار ترنسفورمر با استفاده از ماتریس قطری[۲] $M_{diagonal} \in \mathbb{R}^{S \times S}$ برای ماسک حافظه، تراز شد. ماتریس قطری، نوعی ماتریس است که تمام موقعیت‌ها بجز $s = t$ صفر هستند. در این تعریف s بیانگر موقعیت ردیف و t موقعیت ستون ماتریس است. در واقع، تمام عناصر صفر این ماتریس تبدیل به $-inf$ شده و قطر آن تبدیل به صفر می‌شود. این کار به قصد آن صورت می‌گیرد که پس از جمع کردن آن با ماتریس وزن‌های توجه، باعث نامعتبر شدن تمام موقعیت ها، بجز موقعیت متناظر با برچسب درحال تولید شود. از این رو، مکانیزم توجه متقابل و متعاقباً رمزگشای ترنسفورمر، تنها بردار تعبیه‌ی موقعیتی که در جمله‌ی مبدا، مربوط به برچسب درحال تولید است را در نظر می‌گیرد.

مانند رمزگشای ترنسفورمر، نسخه‌ی تراز شده‌ی آن نیز ازنظر کاربرد به سه بخش تقسیم می‌شود. با تعریف تبدیل خطی پرسش $Q \in \mathbb{R}^{d_{conv}}$، کلید $K \in \mathbb{R}^{d_{conv}}$ و مقدار $V \in \mathbb{R}^{d_{conv}}$ از تعبیه‌ی برچسب‌های هدف $E_{target} \in \mathbb{R}^{L \times d_{conv}}$، نخستین بخش رمزگشای ترنسفورمر تراز شده، مطابق معادله‌ی ۵-۴ مکانیزم توجه به خود، بر روی تعبیه‌ی

---

[۱] علامت I بیانگر فضای واحد (Unit Interval) می‌باشد.

[2] Diagonal Matrix

برچسب‌ها است که برای درک روابط میان برچسب‌های پیشین استفاده شده است.

$$D_{slots} = E_{target} + LayerNorm(MultiHead_{masked}(Q, K, V, M_{upper})) \quad (۴-۵)$$

علت برابری تعداد بُعد تعبیه‌ی برچسب‌های هدف با بعد حافظه، سازگاری این دو برای انجام عملیات توجه متقابل است. بخش دوم رمزگشای ترنسفورمر، توجه متقابل است. ترازی در این بخش تعریف شد تا از توجه رمزگشا به موقعیت‌هایی که متناظر با برچسب درحال پردازش نیستند، جلوگیری کند. با در نظر گرفتن تبدیل خطی پرسش $Q \in \mathbb{R}^{d_{conv}}$ از C، کلید $K \in \mathbb{R}^{d_{conv}}$ و مقدار $V \in \mathbb{R}^{d_{conv}}$ از حافظه H، خروجی لایه‌ی توجه متقابل تراز شده پیشنهادی، با استفاده از معادله‌ی ۴-۶ به دست می‌آید.

$$D_{slots}{}' = D_{slots} + LayerNorm(MultiHead_{masked}(Q, K, V, M_{diagonal})) \quad (۴-۶)$$

بخش آخر رمزگشای ترنسفورمر تراز شده، یک لایه‌ی تغذیه به جلو است.

$$D_{slots}{}'' = D_{slots}{}' + LayerNorm(FeedForward(D_{slots}{}')) \quad (۴-۷)$$

در پایان، برای به دست آوردن احتمال برچسب‌های خروجی $P_{slots} \in \mathbb{I}^{L \times T}$، با استفاده از معادله‌ی ۴-۸، به تمام برچسب‌های موجود در واژه‌نامه‌ی هدف T، احتمالی بین ۰ و ۱ اختصاص داده می‌شود.

$$P_{Slots} = Softmax(LinearLayer(D_{slots}{}'')) \quad (۴-۸)$$

۴-۵ تفسیر خروجی

همانطور که در بخش رمزگشا گفته شد، مدل یک خروجی احتمالاتی تولید می‌کند؛ به این معنا که به ازاء تمام برچسب‌های یکتا و همچنین به ازاء تمام هدف‌های یکتای موجود در مجموعه داده، عددی بین ۰ تا ۱ تولید می‌کند. اما برای استفاده از خروجی شبکه، باید سیاستی برای انتخاب خروجی شبکه از میان احتمالات وجود داشته باشد. در این پایان نامه، سیاست حریصانه برای انتخاب برچسب و هدف استفاده شده است. سیاست حریصانه در انتخاب هدف، به معنای انتخاب کلاسی است که بیشترین احتمال وقوع را دارد. برای بهره‌وری از این سیاست در وظیفه‌ی پر کردن جای خالی، در هر موقعیت از جمله، برچسبی که بیشترین احتمال وقوع را دارد انتخاب می‌شود.

فصل پنجم
پیاده‌سازی و نتایج

۵-۱ مقدمه

در فصل گذشته، معماری پیشنهادی، که مبتنی بر شبکه‌ی عصبی کانولوشنی و ترنسفورمر بود، به تفصیل معرفی گردید. برای بررسی عملکرد آن، مدل پیشنهادی پیاده‌سازی شده و نتایج آن با سایر مدل‌های مطرح در درک زبان طبیعی مقایسه گردید. در فصل پیش رو، تنظیمات و شرایط استفاده شده برای انجام آزمایش‌ها شرح داده می‌شود. سپس، نتایج پیاده سازی مدل پیشنهادی با سایر مدل‌های مطرح مقایسه می‌گردد. در انتهای فصل، به تحلیل میزان اثربخشی هر بخش از مدل پیشنهادی، در نتیجه‌ای که به دست آمده پرداخته می‌شود.

۵-۲ مجموعه داده

به منظور بررسی عملکرد مدل پیشنهادی، آزمایش‌های این فصل بر روی دو مجموعه داده‌ی شناخته شده‌ی ATIS و SNIPS انجام می‌شود. این دو مجموعه داده، در میان مدل‌های درک زبان طبیعی به عنوان یک چالش، برای سنجش عادلانه‌ی عملکرد مدل‌ها و مقایسه‌ی نتایج با کارهای پیشین استفاده می‌شوند. برچسب‌های پر کردن جای خالی، گاهی دارای مفهوم هستند، اما مرسوم است برای آن‌ها، تعبیه‌ی مبتنی بر واژه ایجاد نشود. برای استفاده از این مجموعه داده، مدل بر روی مجموعه داده‌ی آموزشی، آموزش داده شده و ابر پارامترها بر روی مجموعه داده‌ی ارزیابی تنظیم شدند. سپس، نتایج پایانی روی مجموعه‌ی تست گزارش شد. برخی از برچسب‌های هدف و اهداف کاربر، در زمان آموزش مشاهده نشده‌اند. در چنین مواردی، کلاس "Unknown" به آن‌ها اختصاص داده شد. در ادامه، این دو مجموعه داده معرفی می‌شوند.

۵-۲-۱ ATIS

ATIS[1] که به صورت گسترده برای سنجش عملکرد مدل‌های درک زبان طبیعی مورد استفاده قرار گرفته است [۴۰]، یک مجموعه داده در مورد انسان‌هایی است که از سیستم‌های خودکار استعلام سفر هواپیمایی[2] سؤال مورد نظر خود را می‌پرسند [۴۱]. زمینه‌های تحت پوشش این مجموعه داده شامل پرسش در مورد پروازها، هزینه‌ی پرواز، خطوط هواپیمایی، شهرها، فرودگاه‌ها و خدمات زمینی می‌شود. این مجموعه داده شامل ۴۴۷۸ نمونه برای آموزش، ۵۰۰ نمونه برای ارزیابی و ۸۹۳ نمونه برای آزمایش می‌باشد. ATIS دارای ۱۲۷ نوع برچسب یکتا، و همچنین ۲۱ هدف کاربر یکتا است. توزیع این نمونه‌ها به صورت نامتوازن بوده و میانگین طول جملات آن، ۱۱ واژه است.

---

[1] Airline Travel Information Systems

[2] Automated Airline Travel Inquiry Systems

۵-۲-۲ SNIPS

SNIPS یک مجموعه داده‌ی تولید شده از دستیار شخصی دیجیتال SNIPS است، که دارای ۱۳۰۸۴ نمونه برای آموزش، ۷۰۰ نمونه برای ارزیابی و ۷۰۰ نمونه برای آزمایش است [۴۲]. همچنین، این مجموعه داده دارای ۷ نوع هدف یکتا و ۷۲ برچسب یکتا می‌باشد. در تضاد با ATIS، مجموعه داده‌ی SNIPS شامل پرسش درباره‌ی آب و هوا، رستوران‌ها، سرگرمی و غیره بوده و دایره‌ی واژگانی آن گسترده‌تر است. توزیع نمونه‌های این مجموعه داده به صورت متوازن بوده و میانگین تعداد واژه‌ها در جمله، برابر ۹ است.

۵-۳ معیارهای ارزیابی

بعد از آشنایی با مجموعه داده‌های مورد استفاده، معیارهایی که در زمینه‌ی درک زبان طبیعی برای سنجش عملکرد مدل‌ها استفاده شده‌اند، معرفی می‌شوند. برای سنجش عملکرد در وظیفه‌ی پر کردن جای خالی، امتیاز micro f1، و برای سنجش عملکرد تشخیص هدف، معیار دقت[۱] استفاده می‌گردد.

۵-۳-۱ امتیاز f1 برای پر کردن جای خالی

امتیاز f1 معیاری است که برای هر کلاس محاسبه می‌شود؛ به این معنی که اگر مقصود محاسبه‌ی امتیاز کلی f1 را برای مجموعه داده‌ای با بیش از یک کلاس باشد، باید به نحوی امتیازات به دست آمده جمع شود. امتیاز f1 به دو شیوه‌ی micro f1 و macro f1 جمع می‌شود. در شیوه‌ی macro f1، ابتدا درستی[۲] و به یاد آوری[۳] هر کلاس محاسبه شده و سپس، میان درستی هر کلاس و به یاد آوری کلاس‌ها میانگین گرفته می‌شود. اما در مقابل، در شیوه‌ی micro f1 که در معادله‌ی ۵-۱ تعریف شده، به جای میانگین گیری میان دقت و به یاد آوری، امتیاز f1 با استفاده از تعداد کل مثبت واقعی TP، مثبت کاذب FP و منفی کاذب FN حساب می‌شود. در این معادله، c بیانگر تعداد کلاس‌ها است.

$$Micro\, f1 = \frac{\sum^{c} TP}{\sum^{c} TP + \frac{1}{2}(\sum^{c} FP + \sum^{c} FN)} \qquad (۱-۵)$$

در وظیفه‌ی پر کردن جای خالی، از امتیاز micro f1 استفاده شده است. علت استفاده از معیار micro به جای macro، در وظیفه‌ی پر کردن جای خالی، این است میزان توانایی حدس زدن یک کلاس برای این وظیفه کم اهمیت است؛ زیرا با وجود نامتوازن بودن برچسب‌ها، درصورت انتخاب اشتباه کلاس پر استفاده‌ی O در مجموع داده‌ها، خروجی سیستم گفت‌وگو تحت تاثیر قرار می‌گیرد. به این معنا که در تشخیص اشتباه کلاس پر مثال O و یک کلاس کم مثال، تفاوتی نیست. از این جهت، تمامی کلاس‌ها برای این وظیفه از اهمیت یکسان برخوردارند.

[1] Accuracy
[2] Precision
[3] Recall

۵-۳-۲ دقت برای تشخیص هدف

برای سنجش وظیفه‌ی تشخیص هدف، از معیار دقت استفاده می‌شود. معیار دقت، حاصل تقسیم تعداد نمونه‌های صحیح پیش‌بینی شده، بر تعداد کل پیش‌بینی ها است.

۵-۴ تنظیمات آزمایش

ابر پارامترها[1] نقش اساسی در عملکرد یک مدل شبکه عصبی دارند. از آن‌جا که هر جزء CTran از انواع شبکه‌های عصبی متفاوت استفاده می‌کند، از نرخ‌های یادگیری متفاوت α با بهینه ساز AdamW [۴۳] برای هر لایه استفاده شد. CTran با بهره بری از زمان‌بند StepLR، نرخ یادگیری پارامترها در هر دور آموزش، با ضریب γ کاهش می‌یابد. همچنین، حذف تصادفی[2] روی لایه‌ها اعمال می‌شود، تا میزان بیش‌برازش[3] را کاهش دهد [۴۴]. همچنین مشاهده شد که استفاده از برش گرادیان[4] با مقدار ۰/۵ به عملکرد نهائی مدل کمک می‌کند [۴۵]. جدول ۵-۱، α، γ و ρ را برای هر لایه نشان می‌دهد. همچنین اندازه‌ی دسته[5] در هر دو مجموعه داده برابر ۱۶ می‌باشد.

برای اندازه‌ی هسته،اندازه‌های ۱، ۲، ۳، [۱،۳]، [۱،۳،۵]، [۲،۳،۵]، و [۱،۲،۳،۵] امتحان شد، که در آن براکت‌ها نشان‌دهنده‌ی استفاده‌ی چند اندازه‌ی هسته به صورت همزمان است. تعداد هسته‌ها به صورت تجمعی همیشه برابر ۵۱۲ می‌باشد و تعداد هسته‌ها به طور مساوی بین هسته‌های مختلف پخش می‌شود.

لایه / پارامتر	α	γ	ρ
$BERT^{base}_{large}$	$۱۰^{-۴}$ $۱۰^{-۵}$	۰/۹۶	۰/۱
المو	$۱۰^{-۴}$	۰/۹۶	۰/۵
شبکه‌ی کانولوشن و توالی ویژگی پنجره	$۱۰^{-۳}$	۰/۹۶	۰/۰
پشته‌ی رمزگشای ترنسفورمر	$۱۰^{-۴}$	۰/۹۶	۰/۱
رمزگشای تشخیص هدف	$۱۰^{-۴}$	۰/۹۶	۰/۵
رمزگشای تشخیص جای خالی	$۱۰^{-۴}$	۰/۹۶	۰/۵

جدول ۵-۱ - ابر پارامترهای استفاده شده در فاز آموزش. در این جدول، نرخ یادگیری α، نرخ زمان بند λ و احتمال حذف تصادفی ρ استفاده شده برای هر لایه در زمان آموزش مدل پیشنهادی هستند.

روش‌های گوناگونی برای توقف فرایند آموزش وجود دارد. از میان این روش‌ها می‌توان به مکانیزم توقف درصورت

[1] Hyper-parameters
[2] dropout
[3] Overfitting
[4] Gradient Clipping
[5] Batch-Size

عدم رشد، توقف زودرس و توقف با تعداد دور مشخص اشاره کرد. تمامی مدل‌هایی که در فصل ۳ از آن‌ها نام برده شد، از شیوه‌ی توقف با دور مشخص استفاده کردند. در مدل پیشنهادی نیز، از این مکانیزم برای آموزش مدل استفاده گردید. بدین منظور، برای هر آزمایش، مدل برای ۵۰ دور روی مجموعه داده آموزش داده شد و در هر دور از آموزش، نتیجه‌ی آن گزارش گردید. زمانی که دو وظیفه‌ی پر کردن جای خالی و تشخیص هدف، همزمان حداکثر شوند، به عنوان حداکثر دقت مدل در نظر گرفته می‌شوند. برای اطمینان از صحت هر آزمایش، هرکدام از آن‌ها ۱۰ بار اجرا شدند، که هر یک شامل ۵۰ دور آموزش بود. در نهایت، برای گزارش نتیجه‌ی آزمایش، مقدار میانه، از میان ۱۰ بار انجام آزمایش، گزارش شد.

۵-۵ نتایج و آنالیز

مدل	دقت هدف	f1 جای خالی
[۳۰] SASBGCβ	۹۸/۲۱	۹۶/۶۹
[۲۹] Joint Bertβ	۹۷/۵۰	۹۶/۱۰
[۱۹] CNN-BLSTM-Aligned	۹۷/۱۷	۹۷/۷۶
[۴۶] CharEmbed+CNN-LSTM-CRF	۹۹/۰۹	۹۷/۳۲
[۳۵] Elmo+BiLSTM+CRF	۹۷/۴۲	۹۵/۶۲
[۲۶] Bi-directional Interrelated	۹۷/۷۶	۹۵/۷۵
[۳۲] Co-interactive Transformerβ	۹۸/۰۰	۹۶/۱۰
[۳۱] Federated Learningβ	۹۸/۲۸	۹۶/۴۱
[۲۰] Slot-Gated	۹۴/۱۰	۹۵/۲۰
[۲۸] Prior Knowledge	۹۷/۴۰	۹۷/۳۰
[۴۷] CoBiC	۹۹/۴۳	۹۷/۸۲
CTRANβ	**۹۸/۰۷**	**۹۸/۴۶**

جدول ۵-۲ - مقایسه میان مدل پیشنهادی و سایر مدل‌های شناخته شده بر روی مجموعه داده‌ی ATIS. علامت β نمایانگر استفاده‌ی برت در طراحی مدل است. مقادیر همگی به درصد هستند.

زمانی که مدل‌ها به دقت ۱۰۰ درصد نزدیک می‌شوند، پیشرفت‌ها کوچک‌تر شده و سخت‌تر به دست می‌آیند.

جدول ۵-۲، نتایج مدل پیشنهادی در این پایان‌نامه را، با مدل‌های شناخته شده‌ی فعلی، که بالاترین عملکرد را در مجموعه داده‌ی ATIS دارند، مقایسه می‌کند. مدل $\text{CTran+BERT}_{\text{large}}$ در وظیفه‌ی پر کردن جای خالی و بر روی مجموعه‌داده‌ی ATIS، از همه‌ی مدل‌های موجود عملکرد بهتری نشان می‌دهد. مدل پیشنهادی ما، نسبت به بالاترین عملکرد در گذشته، ۰/۶۴ درصد بهبود را نشان می‌دهد. همچنین، مدل پیشنهادی برای وظیفه‌ی تشخیص هدف، نسبت به مدل پایه که مشابه این پایان‌نامه، از ساختار کانولوشن-توالی پنجره ویژگی استفاده کرده است، ۰/۹ درصد بهبود را نشان می‌دهد.

مدل	دقت هدف	f1 جای خالی
[۳۰] SASBGCβ	۹۸/۸۶	۹۶/۴۳
[۲۹] Joint Bertβ	۹۸/۶۰	۹۷/۰۰
[۴۸] CM-netβ	۹۹/۳۲	۹۷/۳۱
[۴۹] Masked Graph+CRFβ	۹۹/۷۰	۹۷/۲۰
[۳۵] Elmo+BiLSTM+CRF	۹۹/۲۹	۹۳/۹۰
[۳۲] Co-interactive Transformerβ	۹۸/۸۰	۹۷/۱۰
[۳۱] Federated Learningβ	۹۹/۳۳	۹۷/۲۰
[۲۰] Slot-Gated	۹۷/۰۰	۸۸/۸۰
[۲۸] Prior Knowledge	۹۸/۷۰	۹۴/۷۰
[۳۳] AISEβ	۹۸/۷۰	۹۷/۲۰
CTRANβ	**۹۹/۴۲**	**۹۸/۳۰**

جدول ۵-۳- مقایسه‌ی مدل پیشنهادی و سایر مدل‌های شناخته شده بر روی مجموعه داده‌ی SNIPS. علامت β نمایانگر استفاده‌ی برت در طراحی مدل است. مقادیر همگی به درصد هستند.

برای سنجیدن صحت برتری مدل CTran نسبت به کارهای پیشین، عملکرد آن روی مجموعه داده‌ی SNIPS نیز سنجیده‌شد. جدول ۵-۳ دقت مدل‌های شناخته شده را با CTran مقایسه می‌کند. برای مجموعه داده‌ی SNIPS، f1 مدل پیشنهادی در پر کردن جای خالی، نسبت به بهترین مدل موجود، ۱ درصد افزایش دارد. اگرچه عملکرد CTran در وظیفه‌ی تشخیص هدف، نسبت به اکثر مدل‌های پیشین برتری دارد، اما از بهترین نتیجه‌ی موجود پیشی نمی‌گیرد و رتبه‌ی دوم را در این وظیفه به خود اختصاص می‌دهد. در زیر بخش‌های بعدی، بررسی می‌شود که چقدر هر ایده،

CTran را بهبود داده است.

۵-۵-۱ اثر لایه کانولوشن با رمزنگار ترنسفورمر

از آن‌جا که در کارهای گذشته، از شبکه‌ی عصبی کانولوشنی برای رفع نقص شبکه عصبی بازگشتی استفاده می‌کرده‌اند، ممکن است بهره‌مندی از رمزنگار ترنسفورمر باعث کاهش اهمیت شبکه‌ی کانولوشنی شود. هدف از استفاده از معماری کانولوشن-توالی ویژگی پنجره با رمزنگار ترنسفورمر، آمیخته کردن تعبیه‌ی نشانه‌های مجاور در یکدیگر می‌باشد. جدول ۵-۴، عملکرد پشته‌ی رمزنگار ترنسفورمر تنها را، در مقابل معماری رمزنگار CTran می‌سنجد. همانطور که مشهود است، برای هر دو مجموعه داده‌ی ATIS و SNIPS، هر دو معیار تشخیص هدف و پرکردن جای خالی با استفاده از معماری پیشنهادی، افزایش یافته است. این نتایج، سودمندی فرایند آمیختگی را تایید می‌کند. همچنین، این نتیجه نشان می‌دهد که با وجود توانمندی ترنسفورمر در درک دوطرفه‌ی معنای جمله، می‌توان با اضافه کردن معنای واژه‌های اطراف، به تعبیه‌ای غنی‌تر دست یافت.

مجموعه داده	ATIS		SNIPS	
مدل	دقت هدف	f1 جای خالی	دقت هدف	f1 جای خالی
کانولوشن-توالی ویژگی پنجره-پشته ترنسفورمر	۹۷/۹۵	۹۸/۳۹	۹۹/۴۲	۹۸/۲۱
فقط پشته‌ی ترنسفورمر	۹۷/۸۸	۹۸/۳۶	۹۹/۰۱	۹۷/۹۱

جدول ۵-۴ - مقایسه‌ی عملکرد رمزنگار ترنسفورمر تنها، با ساختار کانولوشن-توالی ویژگی پنجره-پشته‌ی رمزنگار ترنسفورمر. آزمایش‌ها با مدل زبانی $BERT_{base}$ صورت گرفته است. مقادیر همگی به درصد هستند.

۵-۵-۲ اندازه‌ی هسته

اندازه‌ی هسته، بازه‌ای را که نشانه‌های در آن، با یکدیگر آمیخته می‌شوند را مشخص می‌کند. همچنین، می‌توان اندازه‌ی هسته را در کانولوشن، با n-gramها مقایسه کرد؛ به نحوی که اندازه‌ی هسته ۲ مانند bi-gram است. تاکنون، اندازه هسته‌ی ۱ برای مدل‌های درک زبان طبیعی که مانند CTran از شبکه‌ی کانولوشنی بهره می‌برند، استفاده نشده است. CTran از اندازه‌ی ۱ برای حفظ صریح تعبیه‌ی نشانه استفاده می‌کند. همچنین، استفاده از اندازه‌ی ۱، باعث شده ابعاد تعبیه‌ی نشانه به اندازه‌ی سایر اندازه هسته‌ها شود. این امر سبب می‌شود که ضمن حفظ تعبیه‌ی صریح نشانه، تعبیه‌ی آمیخته با نشانه‌های همسایه نیز به آن اضافه شود. جدول ۵-۵، اثر اندازه‌های مختلف هسته را بر روی عملکرد CTran نشان می‌دهد. در میان هسته‌های تک سایزی، ۱ از سایر موارد برای وظیفه‌ی پر کردن جای خالی عملکرد بهتری نشان داد. علت این امر، این است که اندازه‌ی هسته‌ی ۱، به نوعی مشابه 1-gram است، که این با رابطه‌ی یک به یک ورودی

به برچسب خروجی تداخل ندارد. در همین راستا، مقایسه‌ی [۲،۳،۵] و [۱،۲،۳،۵]، گفتار قبلی این پاراگراف را تایید می‌کند. بر مبنای این نتایج، می‌توان نتیجه گرفت که استفاده‌ی همزمان از چند اندازه‌ی هسته بهتر از هسته‌ی تک سایز است. در پایان، اندازه هسته‌ی [۱،۲،۳،۵] بهترین عملکرد را در هر دو وظیفه تشخیص هدف و پر کردن جای خالی دارد.

مجموعه داده	ATIS		SNIPS	
اندازه‌ی هسته	دقت هدف	f1 جای خالی	دقت هدف	f1 جای خالی
۱	۹۷/۶۱	۹۸/۳۹	۹۸/۶۹	۹۸/۱۴
۲	۹۷/۷۳	۹۸/۳۷	۹۸/۸۴	۹۸/۱۳
۳	۹۷/۸۴	۹۸/۳۵	۹۸/۸۴	۹۸/۱۳
[۱،۳]	۹۷/۸۴	۹۸/۴۰	۹۸/۹۸	۹۸/۱۵
[۱،۳،۵]	۹۷/۹۵	۹۸/۴۳	۹۹/۱۳	۹۸/۲۵
[۱،۲،۳،۵]	۹۸/۰۷	۹۸/۴۶	۹۹/۱۳	۹۸/۳۰
[۲،۳،۵]	۹۸/۰۷	۹۸/۳۸	۹۸/۹۸	۹۸/۲۰

جدول ۵-۵- عملکرد مدل با اندازه هسته‌های متفاوت در لایه‌ی کانولوشنی. براکت نشان دهنده‌ی استفاده از چند اندازه‌ی هسته به صورت همزمان است. مقادیر همگی به درصد هستند.

۵-۵-۳ ترازی رمزگشای ترنسفورمر

رمزگشای ترنسفورمر اصلی، از هیچ ماسکی به جز ماسک‌های پد در بخش توجه متقابل استفاده نمی‌کند؛ به این معنی که در تولید هر نشانه هدف، بردار کلید تبدیل شده از تمام موقعیت‌های حافظه، در محاسبه توجه متقابل استفاده می‌شود. در مقابل، در رمزگشای ترنسفورمر تراز شده، تنها کلید تولید شده از زمینه مربوط به هر نشانه‌ی متناظر بر خروجی، روی توجه متقابل تأثیر می‌گذارد. به منظور تایید اثربخشی تراز پیشنهادی، رمزنگار CTran با یک رمزگشای ترنسفورمر معمولی و در آزمایشی دیگر با یک LSTM تراز شده با مکانیسم توجه که توسط [۱۹] ارائه داده شده است، ترکیب شد. در جدول ۵-۶، f1 پر کردن جای خالی را از در هر سه مدل توصیف شده، مقایسه می‌کند، که در آن تراز پیشنهادی به طور متوسط ۰/۹ درصد بهبود را نسبت به رمزگشای ترنسفورمر معمولی نشان می‌دهد. علاوه بر این، مقایسه رمزگشای ترنسفورمر تراز شده‌ی پیشنهادی با رمزگشای [۱۹] در وظیفه‌ی پر کردن جای خالی ۰/۱ درصد بهبود را نشان می‌دهد. این امر، بیانگر این است که به طور کلی، معماری رمزگشای ترنسفورمر تراز شده، معماری بهتری برای وظیفه‌ی پر

کردن جای خالی است.

مجموعه داده / مدل	ATIS	SNIPS
LSTM تراز شده با مکانیزم توجه	۹۸/۳۰	۹۸/۱۰
رمزگشای ترنسفورمر	۹۷/۴۲	۹۷/۳۷
رمزگشای ترنسفورمر تراز شده	۹۸/۴۰	۹۸/۲۱

جدول ۵-۶ - تاثیر ترازی رمزگشای ترنسفورمر بر روی مدل پیشنهادی. اعداد نمایانگر میزان امتیاز f1 در وظیفه‌ی پر کردن جای خالی است. آزمایش‌ها با مدل زبانی $BERT_{base}$ صورت گرفته است. مقادیر همگی به درصد هستند.

۵-۵-۴ شیوه‌ی استفاده از مدل زبانی در معماری مدل

جدول ۵-۷، دو استراتژی مدل زبانی به عنوان رمزنگار و مدل زبانی به عنوان تعبیه‌ی واژه را با هم مقایسه می‌کند. در حالت اول، مدل زبانی به عنوان رمزنگار در معماری قرار گرفته، و برای رمزگشا، از رمزگشای پیشنهادی CTran استفاده شده است. در مورد دوم، مدل زبانی نقش تعبیه‌گر واژه‌ها را دارد و تعبیه‌های تولید شده توسط مدل زبانی وارد رمزنگار CTran می‌شود.

بر روی هر استراتژی، دو مدل زبانی برت و المو آزمایش گردید. بر اساس نتایج ما، استفاده از المو به عنوان رمزنگار، همراه با رمزگشای CTran، از پیشرفته‌ترین مدل پر کردن جای خالی در ATIS پیشی گرفته است. همچنین، هنگامی که المو تنها نقش تعبیه‌گر واژه‌ها را دارد، عملکرد مدل حتی از حالت قبلی نیز بهتر می‌شود و در هر دو وظیفه‌ی تشخیص هدف و پر کردن جای خالی، افزایش امتیاز مشاهده شد.

در مجموعه داده‌ی SNIPS، استفاده از $\text{BERT}_{\text{base}}$ به عنوان تعبیه‌ی واژه، عملکرد بهتری را در مقایسه با $\text{BERT}_{\text{base}}$ به عنوان رمزنگار نشان می‌دهد. در مقابل، مدل زبانی به عنوان تعبیه‌ی واژه در ATIS عملکرد خوبی نشان نداد و در نتیجه باعث بهبود نتایج نشد. علت این امر ممکن است کوچک بودن دایره‌ی واژگانی و همچنین تعداد کمتر مثال‌های آموزشی مجموعه داده‌ی ATIS باشد. از این رو، داشتن لایه‌های شبکه اضافی می‌تواند باعث بیش برازش مدل شود. برای $\text{BERT}_{\text{large}}$، مدل زبانی به عنوان جاسازی کلمه، در هر دو مجموعه داده عملکرد خوبی داشت. علاوه بر این، معماری پیشنهادی با $\text{BERT}_{\text{large}}$ به حداکثر کارایی دست یافت. اگرچه نتایج نشان می‌دهد که معماری پیشنهادی، بهترین عملکرد را نشان می‌دهد، اما بار محاسباتی بیشتری دارد. آزمایش‌های این بخش، نشان می‌دهد که برت در تمامی موارد، در قیاس با المو بهتر عمل می‌کند؛ اما المو می‌تواند برای مجموعه داده‌هایی که دامنه‌ی واژگانی محدودی دارند،

با دقت عملکرد تقریباً مشابهی استفاده شود. این درحالی است که الگو پیچیدگی محاسباتی و زمان آموزش کمتری دارد. علت وقوع این امر در این مورد، این است که مجموعه داده‌های هدف محور واژگان متنوعی ندارند؛ بنابراین حضور یک مدل زبانی پیش آموز شده را کمتر معنادار می‌کند. همچنین در آزمایش‌های صورت گرفته، $BERT_{base}$ و $BERT_{large}$ برای کار تشخیص هدف مزیتی نسبت به یکدیگر ندارند. اما تفاوت آن‌ها در پر کردن جای خالی مشخص می‌شود، به نحوی که $BERT_{large}$ نسبت به $BERT_{base}$ برای همه مجموعه داده‌ها برتری خود را نشان می‌دهد.

	مجموعه داده	ATIS		SNIPS	
استراتژی	مدل	دقت هدف	f1 جای خالی	دقت هدف	f1 جای خالی
مدل زبانی به عنوان رمزنگار	ELMo + CTRAN's decoder	۹۷/۵۴	۹۸/۱۷	۹۷/۲۵	۹۶/۰۱
	BERT $_{base}$ + CTRAN's decoder	۹۷/۹۹	۹۸/۴۴	۹۸/۸۶	۹۸/۰۰
	BERT $_{large}$ + CTRAN's decoder	۹۷/۹۹	۹۸/۴۳	۹۸/۸۶	۹۸/۱۸
مدل زبانی به عنوان تعبیه	CTRAN + ELMo	۹۷/۸۸	۹۸/۲۵	۹۷/۷۳	۹۶/۶۸
	CTRAN + BERT $_{base}$	۹۷/۹۵	۹۸/۴۰	۹۹/۴۲	۹۸/۲۱
	CTRAN + BERT $_{large}$	۹۸/۰۷	۹۸/۴۶	۹۹/۱۳	۹۸/۳۰

جدول ۵-۷ - دو استراتژی آزمایش شده برای پیدا کردن طرح مناسب برای معماری. از امتیاز F1 برای پر کردن جای خالی و معیار دقت برای تشخیص هدف استفاده گردید. مقادیر همگی به درصد هستند.

۵-۶ هزینه‌ی محاسبات

بار محاسباتی انجام تعبیه‌ی اضافی بعد از مدل زبانی، یک نگرانی به جا می‌باشد؛ زیرا در این استراتژی، پارامترهای بیشتری در شبکه تعریف می‌شوند. با توجه به اینکه از یک Nvidia RTX 3080 ۱۰ گیگابایتی برای محاسبات خود استفاده شد، جدول ۵-۸ زمان آموزش را قبل و بعد از رمزنگار اضافی نشان می‌دهد. محاسبات نشان می‌دهد که بین ۶ تا ۱۱ درصد زمان اضافی برای همگرایی مدل مورد نیاز است. برای این محاسبه، هر مدل را ۱۰ بار، برای ۱۰ دور آموزشی اجرا نموده و مقدار میانه گزارش شد. جدول ۵-۹، زمان استنتاج مدل را نشان می‌دهد. برای این آزمایش، مدل ۱۰ بار برای ۲۰۰ نمونه و بدون دسته کردن ورودی اجرا و مقادیر میانگین گزارش شده است. اگرچه افزایش زمان تمرین محسوس بود اما افزایش زمان استنتاج بسیار ناچیز و تقریباً هم مقیاس با افزایش دقت مدل بود. با مقایسه قبل و بعد از استفاده از رمزنگار اضافی، بین ۱ تا ۱/۵ درصد تاخیر اضافی در استنتاج مشاهده می‌شود. این افزایش در زمان آموزش

مجموعه داده	ATIS		SNIPS	
مدل زبانی / نقش مدل	رمزنگار	تعبیه	رمزنگار	تعبیه
ELMo	۴۲/۵۱	۴۷/۵۵ (۱۱٪)	۹۲/۳۷	۱۰۳/۵۱ (۱۱٪)
BERT $_{base}$	۴۲/۴۰	۴۶/۷۱ (۱۰٪)	۱۱۰/۳۹	۱۱۹/۵۰ (۸٪)
BERT $_{large}$	۷۳/۵۷	۷۹/۴۴ (۷٪)	۱۸۲/۴۲	۱۹۴/۲۴ (۶٪)

جدول ۵-۸ - زمان اضافه شده به فرایند آموزش، با توجه به نقش مدل زبانی در معماری مدل. مقدار زمانی اضافه، ناشی از حاصل استفاده از معماری کانولوشن-توالی ویژگی پنجره-ترنسفورمر است. مقادیر درون جدول نشان دهنده‌ی زمانی است برای یک دور آموزش مدل روی مجموعه داده‌ی مربوطه لازم است. واحد شمارش مقادیر، ثانیه است. مقادیر درون پرانتز نشانگر درصد افزایش زمان است.

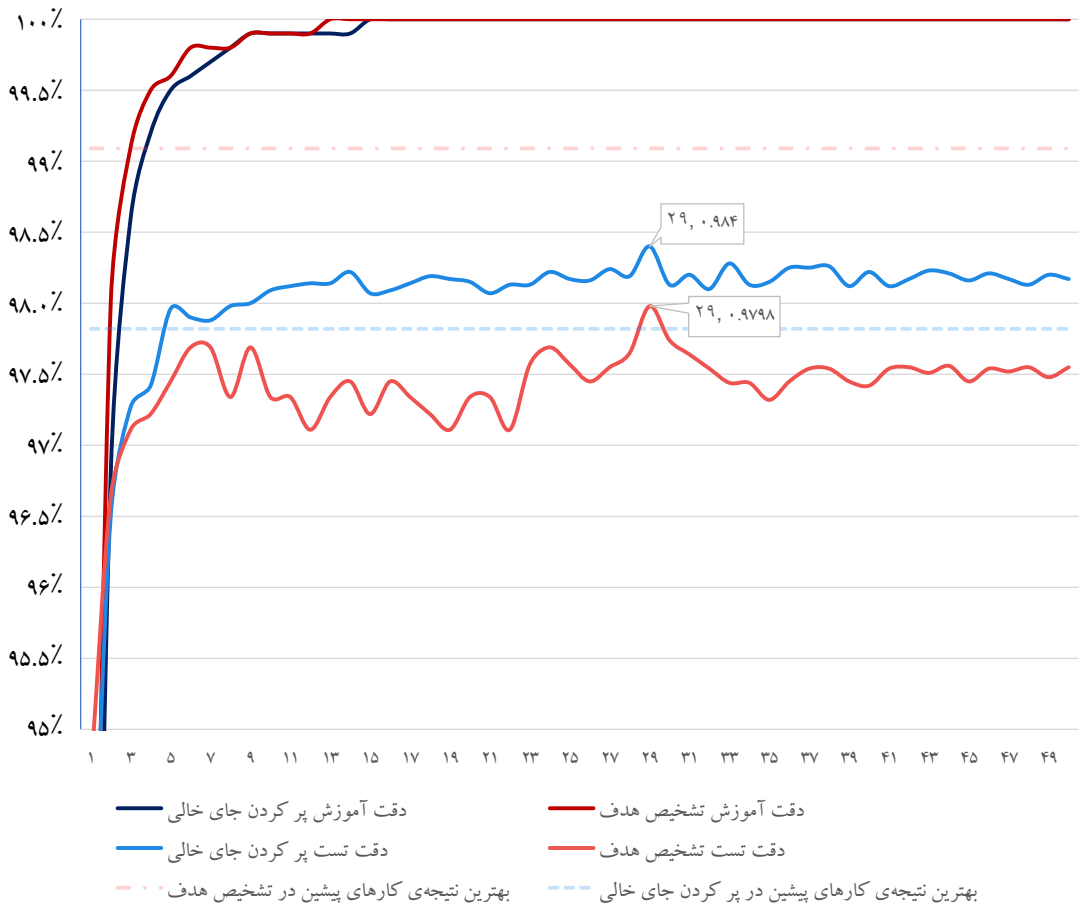
و استنتاج قابل پیش‌بینی بود، زیرا معقول است که مدلی با پارامترهای بیشتر تاخیر اضافی داشته باشد.

مجموعه داده	ATIS		SNIPS	
مدل زبانی / نقش مدل	رمزنگار	تعبیه	رمزنگار	تعبیه
ELMo	۲۰۸	۲۱۱ (۱/۵٪)	۱۹۲	۱۹۵ (۱/۵٪)
BERT $_{base}$	۱۷۸	۱۸۱ (۱/۵٪)	۱۷۷	۱۷۹ (۱٪)
BERT $_{large}$	۱۸۶	۱۸۹ (۱/۵٪)	۱۸۴	۱۸۷ (۱/۵٪)

جدول ۵-۹ - مقایسه‌ی زمان استنتاج مدل، با توجه به نقش مدل زبانی در معماری مدل. مقادیر نشان دهنده‌ی تعداد میلی ثانیه‌ای است که برای استنتاج یک جمله‌ی ورودی لازم است. مقدار درون پرانتز نشان دهنده‌ی درصد افزایش است.

۵-۷ بررسی شیب آموزش

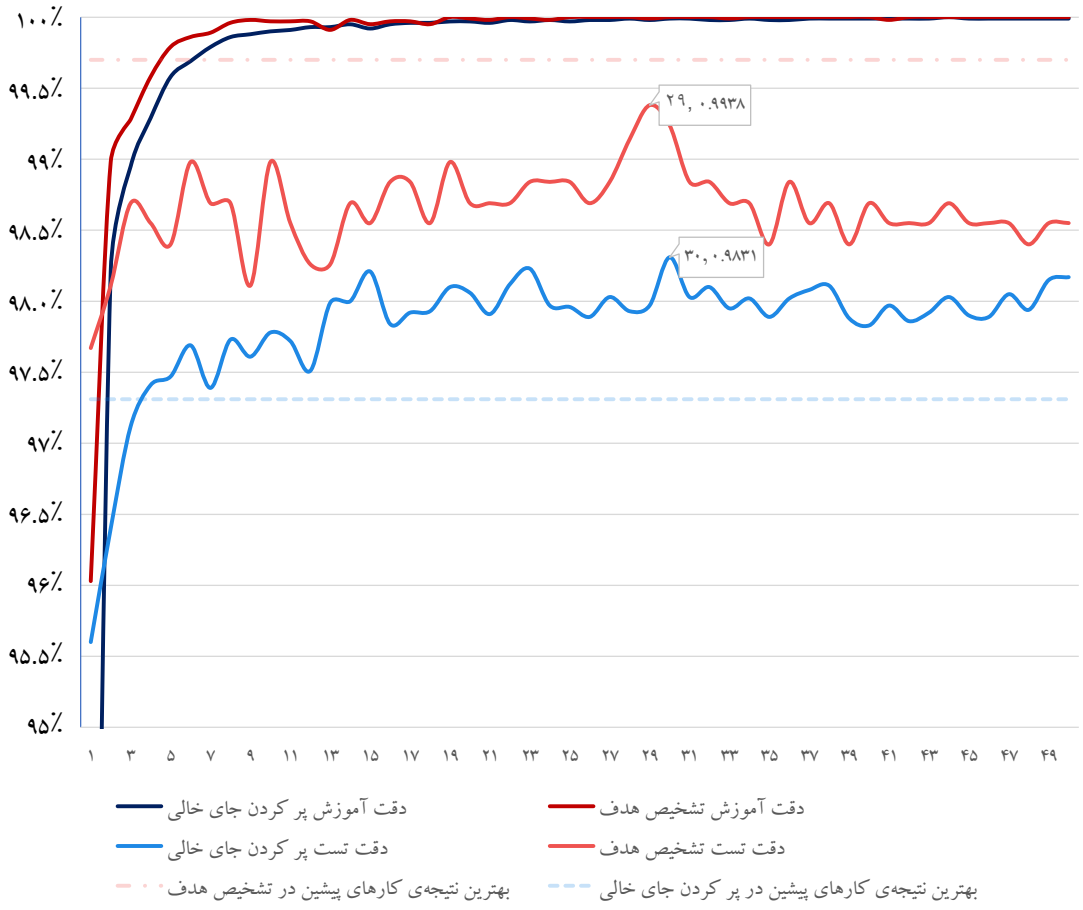
شکل ۵-۱ شیب آموزش مدل پیشنهادی به همراه $BERT_{large}$ را بر روی مجموعه داده‌ی ATIS نشان می‌دهد. در راستای تحلیل وظیفه‌ی پر کردن جای خالی، در طول شیب آموزش مشاهده می‌شود که مدل در بازه‌ی دور ۱ تا ۱۰، دارای یک حداکثر محلی است. مدل در همین بازه نیز از کارهای پیشین سبقت می‌گیرد. برخی از کارهای پیشین محدودیت ۱۰ دور را بر روی آموزش مدل اعمال کردند. در طول آزمایش‌ها مشاهده شد که مدل پس از اینکه در طول ۱۰ دور اول، مدل به حداکثر محلی رسیده و سپس رشد آن متوقف شده و گاهی کاهش می‌یابد. در این بازه، مدل شروع به حداکثر برازش خود رسیده و دقت مدل روی مجموعه‌ی آموزشی، به ۱۰۰درصد رسیده است. محدودیت تعداد دور آموزشی روی ۵۰ دور تنظیم شد. از این رو، پس از ۱۰ دور آموزش، مشاهده می‌شود که دقت مدل ثابت



شکل ۵-۱ - شیب آموزش مدل پیشنهادی بر روی مجموعه داده‌ی ATIS.

بوده و گاهی کاهش می‌یابد؛ اما در ادامه و در دور ۲۹-ام، حداکثر سراسری اتفاق می‌افتد و بیشترین دقت مدل کسب می‌شود. امتیاز مدل در وظیفه‌ی پر کردن جای خالی، پس از دور ۲۹-ام نیز به طور متوسط از دور ۱۰ تا ۲۹ بیشتر است. این پدیده، در [۵۰] نیز مورد بررسی قرار گرفته و نام آن را فرود عمیق دوگانه[1] گذاشتند. فرود عمیق دوگانه بیان می‌کند که با بزرگ شدن اندازه‌ی مدل، دقت مدل بر روی داده‌ی آموزشی، ابتدا کاهش و سپس افزایش پیدا می‌کند. این قضیه برای تعداد دورهای آموزشی وجود دارد؛ جایی که مدل ابتدا به یک قله‌ی محلی می‌رسد، سپس دچار سکون یا افت دقت می‌شود، و سپس با ادامه دادن فرایند آموزش، دقت آن به تدریج زیادتر می‌شود و عملکرد بهتری از قله‌ی اولیه نشان می‌دهد [۵۰]. این قضیه، به وضوح در وظیفه‌ی تشخیص هدف در شکل ۵-۱ مشهود است؛ جایی که مدل بعد از دور ۱۰-ام آموزش، تا دور ۲۴-ام دچار افت شدید دقت شده، و سپس در دور ۲۹-ام حداکثر دقت را نمایش می‌دهد. همچنین، بعد از این قله، میانگین دقت مدل، از پیش از قله بیشتر است. شکل ۵-۲ شیب آموزش مدل پیشنهادی، بر روی مجموعه داده‌ی SNIPS را به همراه $BERT_{large}$ ترسیم می‌کند. به نظر می‌رسد اندازه‌ی بزرگتر مجموعه داده‌ی SNIPS، باعث کمتر شدن اثر افت دقت در طول فرایند آموزش شده است. آنطور که در شکل ۵-۲ مشهود است، مدل

[1] Deep Double Descent



شکل ۵-۲ - شیب آموزش مدل پیشنهادی بر روی مجموعه داده‌ی SNIPS.

در ۱۰ دور ابتدایی آموزش، برای هر دو وظیفه، دچار نوسان بوده و بلافاصله بعد از دور ۱۰-ام دچار افت شدید دقت شد. همچنین، دقت مدل در مجموعه‌ی آموزشی، در دور ۱۰-ام به ۱۰۰درصد رسیده است. در چنین شرایطی، مدلی که از مکانیزم توقف زودرس استفاده کند، آموزش را متوقف کرده و دچار حداکثر دقت محلی می‌شود. در همین راستا، مدل‌هایی مانند [۲۰، ۲۶] که محدودیت ۱۰ دور را بر روی دور آموزش اعمال کردند، دچار این مسئله هستند. مدل پیشنهادی بعد از دور ۱۳-ام مجددا دچار افزایش دقت شد و در دور ۲۹ و ۳۰ بهترین عملکرد را از خود نمایش داد. علاوه بر این، میانگین دقت مدل بعد از قله، بیشتر از قبل قله است. دقت مدل پیشنهادی بر روی وظیفه‌ی پر کردن جای خالی، از دور ۴-ام از بیشترین دقت کارهای قبلی، پیشی می‌گیرد که این بیانگر توانایی تعمیم مدل پیشنهادی است.

۵-۸ خطاهای باقی مانده

همانطور که در جدول نتایج مشخص است، دقت مدل‌های پیشنهادی بسیار بالا و نزدیک به ۱۰۰ درصد است. مدل پیشنهادی در این پایان‌نامه، بالاترین دقت را در وظیفه‌ی پر کردن جای خالی به دست آورد. با توجه به این که تعداد نمونه‌های موجود در مجموعه‌ی تست، که مدل قادر به پیش‌بینی صحیح آن نیست، کم است، بهتر است برای ارائه یک

مدل بهتر، ضعف‌های موجود در مدل را شناخته شود و با آگاهی نسبت به آن‌ها، معماری جدید پیشنهاد شود. از این جهت، در پیوست پایان‌نامه، تمام خطاهایی که برای وظیفه‌ی تشخیص هدف رخ داده، فهرست شده‌اند. همچنین، در بخش پیش رو، نمونه‌هایی را که مدل اشتباه پیش‌بینی می کند، تحلیل می‌شوند.

۵-۸-۱ تحلیل خطاهای مدل پیشنهادی در ATIS

در مجموعه داده‌ی ATIS، ۱۸ خطا در وظیفه‌ی تشخیص هدف، توسط مدل پیشنهادی رخ داد. ۵ مورد از این خطاها شامل مواردی است که برچسب هدف در زمان آموزش دیده نشد است. این برچسب‌ها، atis_airfare#atis_flight، atis_day_name، atis_flight#atis_airline و atis_flight_no#atis_airline هستند. با توجه به ماهیت چالش درک زبان طبیعی، نباید در طراحی معماری، از تجزیه‌گر برچسب، یا واژه‌های توصیفی برای برچسب‌ها و اهداف استفاده شود. از این رو، در زمان تست هرگز امکان حدس صحیح این اهداف، یا تشخیص آن‌ها از یکدیگر، وجود ندارد. همچنین، تعداد زیادی از خطاها به خاطر حدس یک برچسب برای هدف‌هایی است که ترکیبی هستند. در چنین شرایطی، مدل پیشنهادی تنها برچسب هدف اول را پیش‌بینی می‌کند و در پیش‌بینی برچسب ترکیبی ضعف دارد. با مشاهده‌ی لیست خطاها، مشهود است که در مجموعه داده‌ی ATIS، با وجود حدس صحیح و کامل برچسب‌ها، هدف نهائی اشتباه حدس زده شده است. این موضوع، ممکن است بخاطر ادغام ضعیف دو وظیفه‌ی تشخیص هدف و پر کردن جای خالی، در مدل پیشنهادی باشد. در برخی از خطاها، مشاهده می‌شود که مدل، اشتباهاتی را به علت ابهام در مثال‌های آموزشی می‌دهد. به طور مثال، با مشاهده‌ی جمله‌ی "how many northwest flights leave st. paul"، مدل، هدف atis_quantity را پیش‌بینی می کند. با بررسی دقیق‌تر نمونه‌های آموزشی مشاهده می‌شود که این تصمیم، با نمونه‌هایی مانند "how many united flights are there from san francisco please" که دارای مقدار تشخیص هدف atis_quantity است، همخوانی دارد؛ از این رو، نمی‌توان این نوع خطا ها را متوجه معماری دانست.

۵-۸-۲ تحلیل خطاهای مدل پیشنهادی در SNIPS

مدل پیشنهادی ما، در مجموعه داده‌ی SNIPS، ۴ تشخیص اشتباه در وظیفه‌ی تشخیص هدف دارد. چیزی که میان خطاهای مذکور مشترک است، طول کوتاه آن‌ها است. در مجموعه داده‌ی SNIPS، میانگین طول جمله برابر ۹ است و در خطاهای موجود، میانگین طول برابر ۵.۲۵ است. همچنین، مجدداً در خطاها مشاهده می‌شود که با وجود تشخیص صحیح تمام برچسب‌ها، هدف پیش‌بینی شده با هدف صحیح تفاوت دارد.

فصل ششم
نتیجه‌گیری

در این پایان‌نامه، CTran، که مدلی مبتنی بر شبکه‌ی کانولوشنی و ترنسفورمر است، برای درک زبان طبیعی ارائه شد. مدل پیشنهادی از معماری رمزنگار-رمزگشا بهره می‌برد و برای آموزش دو وظیفه‌ی تشخیص هدف و پر کردن جای خالی، از یک رمزنگار اشتراکی استفاده می‌کرد. در معماری پیشنهادی، مدل زبانی برت به عنوان تعبیه‌گر واژه‌ها استفاده شد. سپس نمایش جدیدی از کلمات با استفاده از شبکه‌ی کانولوشنی ایجاد و به جای عملیات Pooling، از ساختار توالی ویژگی پنجره استفاده شد. برای حفظ رابطه‌ی یک به یک واژه‌های ورودی با برچسب‌های هدف، اندازه هسته‌ی ۱ در شبکه‌ی کانولوشنی در کنار سایر اندازه‌ها هسته اضافه شد. همچنین برای ادغام معنای واژه‌های همسایه در یکدیگر، اندازه هسته‌های ۱ تا ۵، به صورت ترکیبی استفاده شد. در نهایت، از پشته‌ی رمزنگار ترنسفورمر برای ایجاد تعبیه‌ی نهائی واژه‌ها بهره برده شد. در معماری CTran برای هر وظیفه، یک رمزگشا تعریف شد. برای وظیفه‌ی تشخیص هدف، از توجه به خود و به دنبال آن یک لایه کاملاً متصل استفاده گردید. همچنین، رمزگشای ترنسفورمر تراز شده برای وظیفه‌ی پر کردن جای خالی معرفی و ضمن ساخت یک پشته از آن، برچسب‌های هدف تولید شدند. در پایان، مدل پیشنهادی با مدل‌های شناخته‌شده مقایسه شد و نتایج نشان داد که مدل CTran در وظیفه‌ی پر کردن جای خالی از مدل‌های موجود بهتر عمل می‌کند.

در این پایان‌نامه، دو سیاست مدل زبانی به جای رمزنگار و مدل زبانی به عنوان تعبیه برای به کارگیری مدل‌های زبانی در معماری شبکه آزمایش شد و نتایج نشان داد که استفاده از مدل‌های زبانی به عنوان تعبیه‌ی کلمه، استراتژی بهتری نسبت به ترکیب آن‌ها در ساختار شبکه دارد. در مقابل، استفاده از مدل‌های زبانی به عنوان تعبیه، به معنای معرفی یک رمزنگار جدا برای شبکه و در نتیجه، معرفی پارامترهای جدید به شبکه است. این امر ذاتاً باعث ایجاد بار محاسباتی جدید می‌شود. از این رو، تاثیر این دو استراتژی در سرعت آموزش و سرعت استنتاج، اندازی گیری شد و تاخیر اضافه‌ی ایجاد شده در شبکه گزارش گردید.

در پایان کار، مواردی هست که می‌توان در آینده آن‌ها را بررسی کرد.

❖**معرفی یک مکانیزم برای تزریق صریح هدف به وظیفه‌ی پر کردن جای خالی یا برعکس:** در این شیوه، با تزریق بردار تعبیه‌ی اهداف یا بردار توجه آن‌ها، به رمزگشای پر کردن جای خالی، مدل را به منظور انتخاب برچسب‌های صحیح راهنمایی می‌کنند. ما در این راستا، دو سیاست را برای ترکیب خروجی‌ها آزمایش کردیم؛ سیاست اول، میانگین‌گیری از ماتریس تفکر برچسب، تبدیل آن به بردار و ترکیب آن با بردار توجه تشخیص هدف، بود. سیاست دوم، الحاق خروجی تشخیص هدف با هر برچسب، قبل از لایه‌ی خطی که ابعاد آن را کاهش می‌دهد، بود. ما هیچ بهبودی پس از اعمال این دو سیاست در مدل مشاهده نکردیم؛ از این رو نتایج آن در این پایان نامه ذکر نشده است. تحقیقات آینده می‌تواند به معرفی مکانیزمی برای ترکیب بهینه‌ی این دو وظیفه با یکدیگر بپردازد.

❖**مدل زبانی مبتنی بر شبکه‌ی کانولوشنی و ترنسفورمر:** در این پژوهش، مشاهده شد که استفاده از شبکه‌ی

کانولوشنی با پشته‌ی رمزنگار ترنسفورمر عملکرد شبکه را بهبود می‌بخشد. این نتیجه، نوید از سودمندی این معماری می‌دهد؛ بنابراین در آینده می‌توان یک مدل زبانی مبتنی بر کانولوشن-ترنسفورمر معرفی کرد و عملکرد آن را با مدل زبانی برت سنجید.

❖**آزمایش معماری در سایر وظیفه‌ها:** همان‌طور که در فصل اول گفته شد، معماری معرفی شده برای درک زبان طبیعی، کاربرد گسترده‌ای در پردازش زبان طبیعی دارد؛ چرا که خروجی آن به شکلی است که امکان استفاده از این ساختار شبکه را در بسیاری از وظایف پردازش زبان طبیعی می‌دهد. در پژوهش‌های آینده، می‌توان عملکرد این معماری در سایر وظایف را نیز سنجید. این سنجش نیازی به تغییر ساختار معماری ندارد و مدل عیناً قابلیت جابجایی برای وظایف یاد شده در فصل اول را دارد. همچنین، با بسط دادن معماری، می توان آن را در سایر وظایفی که نامی از آن‌ها برده نشده است نیز به کار گرفت.

❖**بررسی دقیق‌تر مدل‌های زبانی:** ما در پژوهش خود، عملکرد دو مدل زبانی المو و برت را سنجیدیم. ما همچنین عملکرد مدل را با سایر مدل‌های زبانی مطرح که شامل اکس ال نت[۱]، روبرت[۲]، الکترا[۳] هستند نیز سنجیدیم. عملکرد این مدل‌ها در مقابل برت بر روی معماری ما بهبودی نداشت؛ اما از طرفی مقایسه‌ی انجام شده عادلانه نبود. علت ناعادلانه بودن آن، آموزش برت بر روی داده‌هایی با حروف کوچک بود؛[۴]. متن جملات در هر دو مجموعه داده‌ی ATIS و SNIPS، با حروف کوچک است. این درحالی است که مدل‌های زبانی روبرت، الکترا و اکس ال نت، بر روی مجموعه داده‌ی با حروف کوچک و بزرگ[۵] آموزش داده شده‌اند. از این رو، در جدول نتایج این پایان نامه، نتایج مربوط به سه مدل زبانی یاد شده، درج نشده است. در آینده می‌توان به دو روش، مقایسه‌ی عادلانه‌ای، بین این مدل‌های زبانی برای وظیفه‌ی درک زبان طبیعی انجام داد. نخست، آموزش مدل‌های زبانی مذکور بر روی داده‌های حروف کوچک است که این مورد از نظر هزینه‌ی محاسباتی، بار سنگینی دارد. دوم، تغییر جملات مجموعه داده به حروف کوچک و بزرگ است که نیازمند نیروی کار و هزینه‌ی مادی است.

---

[1] XLNet
[2] RoBert
[3] Electra
[4] Uncased
[5] Cased

پیوست‌ها

پ-۱ جزئیات اشتباهات مدل

پ-۱-۱ اشتباهات در مجموعه داده‌ی ATIS

show flight and prices kansas city to chicago on next wednesday arriving in chicago by 7 pm	جمله‌ی ورودی	۱
O O O O B-fromloc.city_name I-fromloc.city_name O B-toloc.city_name O B-depart_date.date_relative B-depart_date.day_name O O B-toloc.city_name B-arrive_time.time_relative B-arrive_time.time I-arrive_time.time	برچسب‌های صحیح	
O O O O B-fromloc.city_name I-fromloc.city_name O B-toloc.city_name O B-depart_date.date_relative B-depart_date.day_name O O B-toloc.city_name B-arrive_time.time_relative B-arrive_time.time I-arrive_time.time	برچسب‌های پیش‌بینی شده	
atis_flight#atis_airfare	هدف صحیح	
atis_flight	هدف پیش‌بینی شده	
what day of the week do flights from nashville to tacoma fly on	جمله‌ی ورودی	۲
O O O O O O O O B-fromloc.city_name O B-toloc.city_name O O	برچسب‌های صحیح	
O O O O O O O O B-fromloc.city_name O B-toloc.city_name O O	برچسب‌های پیش‌بینی شده	
atis_day_name	هدف صحیح	
atis_flight	هدف پیش‌بینی شده	
what days of the week do flights from san jose to nashville fly on	جمله‌ی ورودی	۳
O O O O B-fromloc.city_name O B-airline_name I-airline_name O B-class_type I-class_type	برچسب‌های صحیح	
O O O O B-fromloc.city_name O B-airline_name I-airline_name O B-class_type I-class_type	برچسب‌های پیش‌بینی شده	
atis_day_name	هدف صحیح	
atis_flight	هدف پیش‌بینی شده	
what's the fare for a taxi to denver	جمله‌ی ورودی	٤
O O O O O B-transport_type O B-city_name	برچسب‌های صحیح	
O O O O O B-transport_type O B-toloc.city_name	برچسب‌های پیش‌بینی شده	
atis_ground_fare	هدف صحیح	
atis_airfare	هدف پیش‌بینی شده	
to what cities from boston does america west fly first class	جمله‌ی ورودی	۵
O O O O B-fromloc.city_name O B-airline_name I-airline_name O B-class_type I-class_type	برچسب‌های صحیح	
O O O O B-fromloc.city_name O B-airline_name I-airline_name O B-class_type I-class_type	برچسب‌های پیش‌بینی شده	
atis_city	هدف صحیح	
atis_flight	هدف پیش‌بینی شده	

٦	جمله‌ی ورودی	list all sunday flights from cleveland to nashville and their fares
	برچسب‌های صحیح	O O O O O O O O B-fromloc.city_name O B-toloc.city_name O O
	برچسب‌های پیش‌بینی شده	O O O O O O O O B-fromloc.city_name O B-toloc.city_name O O
	هدف صحیح	atis_flight#atis_airfare
	هدف پیش‌بینی شده	atis_flight
۷	جمله‌ی ورودی	list the airfare for american airlines flight 19 from jfk to lax
	برچسب‌های صحیح	O O O O B-airline_name I-airline_name O B-flight_number O B-fromloc.airport_code O B-toloc.airport_code
	برچسب‌های پیش‌بینی شده	O O O O B-airline_name I-airline_name O B-flight_number O B-fromloc.airport_code O B-toloc.city_name
	هدف صحیح	atis_airfare#atis_flight
	هدف پیش‌بینی شده	atis_airfare
۸	جمله‌ی ورودی	i need a round trip flight from san diego to washington dc and the fares
	برچسب‌های صحیح	O O O B-round_trip I-round_trip O O B-fromloc.city_name I-fromloc.city_name O B-toloc.city_name B-toloc.state_code O O O
	برچسب‌های پیش‌بینی شده	O O O B-round_trip I-round_trip O O B-fromloc.city_name I-fromloc.city_name O B-toloc.city_name B-toloc.state_code O O O
	هدف صحیح	atis_flight#atis_airfare
	هدف پیش‌بینی شده	atis_flight
۹	جمله‌ی ورودی	i need a round trip from atlanta to washington dc and the fares leaving in the morning
	برچسب‌های صحیح	O O O B-round_trip I-round_trip O B-fromloc.city_name O B-toloc.city_name B-toloc.state_code O O O O O O B-depart_time.period_of_day
	برچسب‌های پیش‌بینی شده	O O O B-round_trip I-round_trip O B-fromloc.city_name O B-toloc.city_name B-toloc.state_code O O O O O O B-depart_time.period_of_day
	هدف صحیح	atis_flight#atis_airfare
	هدف پیش‌بینی شده	atis_airfare

۱۰	جمله‌ی ورودی	i need flight and airline information for a flight from denver to salt lake city on monday departing after 5 pm
	برچسب‌های صحیح	O O O O O O O O O O B-fromloc.city_name O B-toloc.city_name I-toloc.city_name I-toloc.city_name O B-depart_date.day_name O B-depart_time.time_relative B-depart_time.time I-depart_time.time
	برچسب‌های پیش‌بینی شده	O O O O O O O O O O B-fromloc.city_name O B-toloc.city_name I-toloc.city_name I-toloc.city_name O B-depart_date.day_name O B-depart_time.time_relative B-depart_time.time I-depart_time.time
	هدف صحیح	atis_flight#atis_airline
	هدف پیش‌بینی شده	atis_flight
۱۱	جمله‌ی ورودی	i need flight and fare information for thursday departing prior to 9 am from oakland going to salt lake city
	برچسب‌های صحیح	O O O O O O O B-depart_date.day_name O B-depart_time.time_relative I-depart_time.time_relative B-depart_time.time I-depart_time.time O B-fromloc.city_name O O B-toloc.city_name I-toloc.city_name I-toloc.city_name
	برچسب‌های پیش‌بینی شده	O O O O O O O B-depart_date.day_name O B-depart_time.time_relative O B-depart_time.time I-depart_time.time O B-fromloc.city_name O O B-toloc.city_name I-toloc.city_name I-toloc.city_name
	هدف صحیح	atis_flight#atis_airfare
	هدف پیش‌بینی شده	atis_flight
۱۲	جمله‌ی ورودی	i need flight and fare information departing from oakland to salt lake city on thursday before 8 am
	برچسب‌های صحیح	O O O O O O O O B-fromloc.city_name O B-toloc.city_name I-toloc.city_name I-toloc.city_name O B-depart_date.day_name B-depart_time.time_relative B-depart_time.time I-depart_time.time
	برچسب‌های پیش‌بینی شده	O O O O O O O O B-fromloc.city_name O B-toloc.city_name I-toloc.city_name I-toloc.city_name O B-depart_date.day_name B-depart_time.time_relative B-depart_time.time I-depart_time.time
	هدف صحیح	atis_flight#atis_airfare
	هدف پیش‌بینی شده	atis_flight

i need flight numbers and airlines for flights departing from oakland to salt lake city on thursday departing before 8 am	جمله‌ی ورودی	۱۳
O O O O O O O O O O B-fromloc.city_name O B-toloc.city_name I-toloc.city_name I-toloc.city_name O B-depart_date.day_name O B-depart_time.time_relative B-depart_time.time I-depart_time.time	برچسب‌های صحیح	
O O O O O O O O O O B-fromloc.city_name O B-toloc.city_name I-toloc.city_name I-toloc.city_name O B-depart_date.day_name O B-depart_time.time_relative B-depart_time.time I-depart_time.time	برچسب‌های پیش‌بینی شده	
atis_flight_no#atis_airline	هدف صحیح	
atis_flight_no	هدف پیش‌بینی شده	
list la	جمله‌ی ورودی	۱٤
O B-city_name	برچسب‌های صحیح	
O B-airline_code	برچسب‌های پیش‌بینی شده	
atis_city	هدف صحیح	
atis_abbreviation	هدف پیش‌بینی شده	
how many northwest flights leave st. paul	جمله‌ی ورودی	۱۵
O O B-airline_name O O B-fromloc.city_name I-fromloc.city_name	برچسب‌های صحیح	
O O B-airline_name O O B-fromloc.city_name I-fromloc.city_name	برچسب‌های پیش‌بینی شده	
atis_flight	هدف صحیح	
atis_quantity	هدف پیش‌بینی شده	
what is a d9s	جمله‌ی ورودی	۱٦
O O O B-aircraft_code	برچسب‌های صحیح	
O O O B-aircraft_code	برچسب‌های پیش‌بینی شده	
atis_abbreviation	هدف صحیح	
atis_aircraft	هدف پیش‌بینی شده	
how many flights does alaska airlines have to burbank	جمله‌ی ورودی	۱۷
O O O O B-airline_name I-airline_name O O B-toloc.city_name	برچسب‌های صحیح	
O O O O B-airline_name I-airline_name O O B-toloc.city_name	برچسب‌های پیش‌بینی شده	
atis_flight	هدف صحیح	
atis_quantity	هدف پیش‌بینی شده	

پ-۱-۲ اشتباهات در مجموعه داده‌ی SNIPS

۱	جمله‌ی ورودی	listen to dragon ball: music collection
	برچسب‌های صحیح	O O B-object_name I-object_name I-object_name I-object_name
	برچسب‌های پیش‌بینی شده	O O B-object_name I-object_name I-object_name I-object_name
	هدف صحیح	SearchCreativeWork
	هدف پیش‌بینی شده	PlayMusic
۲	جمله‌ی ورودی	find the panic in needle park
	برچسب‌های صحیح	O B-movie_name I-movie_name I-movie_name I-movie_name I-movie_name
	برچسب‌های پیش‌بینی شده	O B-object_name I-object_name I-object_name I-object_name I-object_name
	هدف صحیح	SearchScreeningEvent
	هدف پیش‌بینی شده	SearchCreativeWork
۳	جمله‌ی ورودی	play the album journeyman
	برچسب‌های صحیح	O O B-object_type B-object_name
	برچسب‌های پیش‌بینی شده	O O B-music_item B-album
	هدف صحیح	SearchCreativeWork
	هدف پیش‌بینی شده	PlayMusic
٤	جمله‌ی ورودی	find now and forever
	برچسب‌های صحیح	O B-movie_name I-movie_name I-movie_name
	برچسب‌های پیش‌بینی شده	O B-object_name I-object_name I-object_name
	هدف صحیح	SearchScreeningEvent
	هدف پیش‌بینی شده	SearchCreativeWork

مراجع

[1] J. Lesser, "From tvs to beertails: How chatbots help brands engage consumers on twitter," *Twitter Marketing*, Dec 2017. [Online]. Available: https://marketing.twitter.com/en/perspectives/from-tvs-to-beertails-how-chatbots-help-brands-engage-consumers-on-twitter

[2] D. Rumelhart, G. Hinton, and R. Williams, "Learning internal representations by error propagation," California Univ San Diego La Jolla Inst for Cognitive Science, Tech. Rep., 1985.

[3] S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber, *Gradient flow in recurrent nets: the difficulty of learning long-term dependencies*. A field guide to dynamical recurrent neural networks. IEEE Press, 2001, pp. 237–243.

[4] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166, 1994.

[5] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[6] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[7] S. Noh, "Analysis of gradient vanishing of rnns and performance comparison," *Information*, vol. 12, no. 11, 2021.

[8] A. Kag, Z. Zhang, and V. Saligrama, "Rnns incrementally evolving on an equilibrium manifold: A panacea for vanishing and exploding gradients?" in *International Conference on Learning Representations*, Addis Ababa, Ethiopia, 2020.

[9] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi *et al.*, "Transformers: State-of-the-art natural language processing," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Punta Cana, Dominican Republic: Association for Computational Linguistics, Oct. 2020, pp. 38–45.

[10] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez *et al.*, "Attention is all you need," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., 2017, pp. 6000–6010.

[11] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, Las Vegas, United States, 2016, pp. 770–778.

[12] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[13] J. Pennington, R. Socher, and C. Manning, "GloVe: Global vectors for word representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1532–1543.

[14] M. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee *et al.*, "Deep contextualized word representations," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 2227–2237.

[15] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, USA: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186.

[16] S. Louvan and B. Magnini, "Exploring named entity recognition as an auxiliary task for slot filling in conversational language understanding," in *Proceedings of the 2018 EMNLP Workshop SCAI: The 2nd International Workshop on Search-Oriented Conversational AI*, Brussels, Belgium, 2018, pp. 74–80.

[17] N. T. Vu, P. Gupta, H. Adel, and H. Schütze, "Bi-directional recurrent neural network with ranking loss for spoken language understanding," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Shanghai, China: IEEE, 2016, pp. 6060–6064.

[18] B. Liu and I. Lane, "Attention-based recurrent neural network models for joint intent detection and slot filling," in *Proceedings of Interspeech 2016*, San Francisco, USA, 2016, pp. 685–689.

[19] Y. Wang, L. Tang, and T. He, "Attention-based cnn-blstm networks for joint intent detection and slot filling," in *Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data*, M. Sun, T. Liu, X. Wang, Z. Liu, and Y. Liu, Eds. Cham, Switzerland: Springer International Publishing, May 2018, pp. 250–261.

[20] C. Goo, G. Gao, Y. Hsu, C. Huo, T. Chen, K. Hsu *et al.*, "Slot-gated modeling for joint slot filling and intent prediction," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. New Orleans, USA: Association for Computational Linguistics, Jun. 2018, pp. 753–757.

[21] Y. Zhang, "Joint models for NLP," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: Tutorial Abstracts*. Melbourne, Australia: Association for Computational Linguistics, Oct.-Nov. 2018.

[22] A. Jaech, L. Heck, and M. Ostendorf, "Domain adaptation of recurrent neural networks for natural language understanding," *arXiv preprint arXiv:1604.00117*, 2016.

[23] P. Wei, B. Zeng, and W. Liao, "Joint intent detection and slot filling with wheel-graph attention networks," *Journal of Intelligent & Fuzzy Systems*, vol. 42, no. 3, pp. 2409–2420, 2022.

[24] S. Varghese, S. Sarang, V. Yadav, B. Karotra, and N. Gandhi, "Bidirectional lstm joint model for intent classification and named entity recognition in natural language

understanding," *International Journal of Hybrid Intelligent Systems*, vol. 16, no. 1, pp. 13–23, 2020.

[25] W. Xu, B. Haider, and S. Mansour, "End-to-end slot alignment and recognition for cross-lingual nlu," *arXiv preprint arXiv:2004.14353*, 2020.

[26] H. E, P. Niu, Z. Chen, and M. Song, "A novel bi-directional interrelated model for joint intent detection and slot filling," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 5467–5471.

[27] M. Firdaus, H. Golchha, A. Ekbal, and P. Bhattacharyya, "A deep multi-task model for dialogue act classification, intent detection and slot filling," *Cognitive Computation*, vol. 13, no. 3, pp. 626–645, May 2021.

[28] C. Hou, J. Li, H. Yu, X. Luo, and S. Xie, *Prior knowledge modeling for joint intent detection and slot filling*. World Scientific, 2023, pp. 3–10.

[29] Q. Chen, Z. Zhuo, and W. Wang, "Bert for joint intent classification and slot filling," *arXiv preprint arXiv:1902.10909*, 2019.

[30] C. Wang, Z. Huang, and M. Hu, "Sasgbc: Improving sequence labeling performance for joint learning of slot filling and intent detection," in *Proceedings of 2020 the 6th International Conference on Computing and Data Engineering*, ser. ICCDE 2020. New York, USA: Association for Computing Machinery, 2020, p. 29–33.

[31] Z. Huang, F. Liu, and Y. Zou, "Federated learning for spoken language understanding," in *Proceedings of the 28th International Conference on Computational Linguistics*. Barcelona, Spain: International Committee on Computational Linguistics, Dec. 2020, pp. 3467–3478.

[32] L. Qin, T. Liu, W. Che, B. Kang, S. Zhao, and T. Liu, "A co-interactive transformer for joint slot filling and intent detection," in *2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Toronto, Canada, 2021, pp. 8193–8197.

[33] P. Yang, D. Ji, C. Ai, and B. Li, "Aise: Attending to intent and slots explicitly for better spoken language understanding," *Knowledge-Based Systems*, vol. 211, p. 106537, 2021.

[34] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. Le, and R. Salakhutdinov, "Transformer-xl: Attentive language models beyond a fixed-length context," *arXiv preprint arXiv:1901.02860*, 2019.

[35] A. Siddhant, A. Goyal, and A. Metallinou, "Unsupervised transfer learning for spoken language understanding in intelligent agents," *Proceedings of the Association for the Advancement of Artificial Intelligence Conference on Artificial Intelligence*, vol. 33, no. 01, pp. 4959–4966, jul 2019.

[36] K. Ethayarajh, "How contextual are contextualized word representations? Comparing the geometry of BERT, ELMo, and GPT-2 embeddings," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and*

the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 55–65.

[37] Y. Wu, M. Schuster, Z. Chen, Q. Le, M. Norouzi, W. Macherey *et al.*, "Google's neural machine translation system: Bridging the gap between human and machine translation," *arXiv preprint arXiv:1609.08144*, 2016.

[38] C. Zhou, C. Sun, Z. Liu, and F. Lau, "A c-lstm neural network for text classification," *arXiv preprint arXiv:1511.08630*, 2015.

[39] C. Suen, "n-gram statistics for natural language understanding and text processing," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-1, no. 2, pp. 164–172, 1979.

[40] G. Tur, D. HakkaniTür, and L. Heck, "What is left to be understood in atis?" in *2010 IEEE Spoken Language Technology Workshop*, Berkeley, USA, 2010, pp. 19–24.

[41] C. Hemphill, J. Godfrey, and G. Doddington, "The ATIS spoken language systems pilot corpus," in *Speech and Natural Language: Proceedings of a Workshop*, Hidden Valley, USA, June 1990.

[42] A. Coucke, A. Saade, A. Ball, T. Bluche, A. Caulier, D. Leroy *et al.*, "Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces," *arXiv preprint arXiv:1805.10190*, 2018.

[43] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2017.

[44] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, jan 2014.

[45] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ser. ICML'13. Microtome Publishing, 2013, pp. 1310–1318.

[46] M. Firdaus, A. Kumar, A. Ekbal, and P. Bhattacharyya, "A multi-task hierarchical approach for intent detection and slot filling," *Knowledge-Based Systems*, vol. 183, 2019.

[47] B. Kane, F. Rossi, O. Guinaudeau, V. Chiesa, I. Quénel, and S. Chau, "Joint intent detection and slot filling via cnn-lstm-crf," in *2020 6th IEEE Congress on Information Science and Technology (CiSt)*, Virtual Event, 2020, pp. 342–347.

[48] Y. Liu, F. Meng, J. Zhang, J. Zhou, Y. Chen, and J. Xu, "CM-net: A novel collaborative memory network for spoken language understanding," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 1051–1060.

[49] H. Tang, D. Ji, and Q. Zhou, “End-to-end masked graph-based crf for joint slot filling and intent detection,” *Neurocomputing*, vol. 413, pp. 348–359, 2020.

[50] P. Nakkiran, G. Kaplun, Y. Bansal, T. Yang, B. Barak, and I. Sutskever, “Deep double descent: Where bigger models and more data hurt,” *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2021, no. 12, p. 124003, 2021.

واژه‌نامه انگلیسی به فارسی

Activation Function	تابع فعال‌ساز	Early Stopping	توقف زودرس
Attention Mechanism	مکانیزم توجه	Embedding	تعبیه
Automation	خودکارسازی	Encoder	رمزنگار
AutoRegressive	خود همبسته	Epoch	دور
Back-Propagation	پس‌انتشار	Explicit	صریح
Batch-Size	اندازه‌ی دسته	Exploding Gradient	گرادیان انفجاری
Bottleneck	گلوگاه	Fake-News	اخبار جعلی
Component	مؤلفه	Feature Map	نقشه‌ی ویژگی
Concatenation	الحاق	Feed-Forward	تغذیه‌به‌جلو
Conditional Random Field	میدان تصادفی شرطی	Finetune	تنظیم دقیق
Context	پیش‌زمینه	Forget Gate	دروازه‌ی فراموشی
Continuous Bag-Of-Words	کیسه‌ی کلمات ممتد	Gesture	ژست
Continuous Skip-Gram	اسکیپ‌گرام ممتد	Goal-Driven	هدف محور
Convolutional Neural Network	شبکه‌ی عصبی کانولوشنی	Gradient Clipping	برش گرادیان
Corpus	مجموعه‌ی متنی	Gradient Descent	نزول گرادیان
Cross-Attention	توجه متقابل	Haptic	لمس
Dataset	مجموعه‌داده	Hate-Speech	کلام نفرت‌افکن
Decoder	رمزگشا	Hidden State	حالت مخفی
Deep Double Descent	فرود عمیق دوگانه	Hyper-Parameter	ابر پارامتر
Diagonal Matrix	ماتریس قطری	Inferrence	استنتاج
Dialogue Act Classification	طبقه بندی قانون گفتگو	Input Gate	دروازه‌ی ورودی
Dialogue Management Unit	واحد مدیریت مکالمه	Intent-Detection	تشخیص هدف
Dialogue System	سیستم گفت‌وگو	Kernel	هسته
Digital Assistant	دستیار دیجیتال	Key	کلید
Dimension	بعد	Language Generation Unit	واحد تولید زبان
Discontinuous	ناپیوسته	Language Model	مدل زبانی
Dot Product	ضرب نقطه‌ای	Linear Transformation	تبدیلات خطی
Dropout	حذف تصادفی	Long Short-Term Memory	حافظه کوتاه‌مدت طولانی

Masked Multi-Headed Attention	توجه چند سر پوشیده	Sequence Labeling	برچسب‌زنی توالی
Memory Cell	سلول حافظه	Sliding Window	پنجره لغزان
Multi-Head attention	توجه چند سر	Slot-Filling	تشخیص جای‌خالی
Named-Entity Recognition	شناسایی موجودیت‌های نام‌دار	Speech	گفتار
Natural Language Understanding	درک زبان طبیعی	Stochastic Gradient Descent	نزول گرادیان تصادفی
Node	گره	Stride	گام
Operational Cost	هزینه‌ی عملیاتی	Sub-Task	زیروظیفه
Optimization Algorithm	الگوریتم بهینه سازی	Supervised	باناظر
Output Gate	دروازه‌ی خروجی	Syntax	نحو
Overall Goal	هدف کلی	Teacher Forcing	اجبار معلم
Overfitting	بیش‌برازش	Thought Vector	بردار تفکر
Part-Of-Speech Tagging	برچسب‌زنی اجزای کلام	Time Series	سری های زمانی
Policy	سیاست	Time-Step	گام زمانی
Positional Encoding	تعبیه‌ی موقعیتی	Token	نشانه
Position-Awared Multi-head Masked Attention	توجه چندسر پوشیده‌ی آگاه از موقعیت	Token ID	شناسه‌ی نشان
		Tokenize	نشان کردن
Pragmatics	عمل‌شناسی	Tokenizer	نشانه ساز
Precision	درستی	Transformer	ترنسفورمر
Pre-trained	پیش‌آموز شده	Transpose	ترانهاده
Query	پرسش	Unbalanced	نامتوازن
Recall	به یاد آوری	Upper Triangular Matrix	ماتریس مثلثی بالایی
Segment Embedding	تعبیه‌ی بخش	Value	مقدار
Self-Attention	توجه به خود	Vanishing Gradient	گرادیان محو شونده
Self-Supervised	خوداناظر	Vector	بردار
Semantic Information	اطلاعات معنایی	WhiteSpace	فضای خالی
Semantic Relations	روابط معنایی	Word Co-occurance Matrix	ماتریس هم‌آیی کلمات
Semantics	معناشناسی		
Sentiment Analysis	تحلیل احساسات		

واژه‌نامه فارسی به انگلیسی

تشخیص جای‌خالی	Slot-Filling	ابر پارامتر	Hyper-Parameter
تشخیص هدف	Intent-Detection	اجبار معلم	Teacher Forcing
تعبیه	Embedding	اخبار جعلی	Fake-News
تعبیه‌ی بخش	Segment Embedding	استنتاج	Inferrence
تعبیه‌ی موقعیتی	Positional Encoding	اسکیپ‌گرام ممتد	Continuous Skip-Gram
تغذیه‌به‌جلو	Feed-Forward	اطلاعات معنایی	Semantic Information
تنظیم دقیق	Finetune	الحاق	Concatenation
توجه به خود	Self-Attention	الگوریتم بهینه سازی	Optimization Algorithm
توجه چند سر	Multi-Head attention	اندازه‌ی دسته	Batch-Size
توجه چند سر پوشیده	Masked Multi-Headed Attention	باناظر	Supervised
توجه چندسر پوشیده‌ی آگاه از موقعیت	Position-Awared Multi-head Masked Attention	برچسب‌زنی اجزای کلام	Part-Of-Speech Tagging
		برچسب‌زنی توالی	Sequence Labeling
توجه متقابل	Cross-Attention	بردار	Vector
توقف زودرس	Early Stopping	بردار تفکر	Thought Vector
حافظه کوتاه‌مدت طولانی	Long Short-Term Memory	برش گرادیان	Gradient Clipping
حالت مخفی	Hidden State	بعد	Dimension
حذف تصادفی	Dropout	به یاد آوری	Recall
خود همبسته	AutoRegressive	بیش‌برازش	Overfitting
خودکارسازی	Automation	پرسش	Query
خودناظر	Self-Supervised	پس‌انتشار	Back-Propagation
درستی	Precision	پنجره لغزان	Sliding Window
درک زبان طبیعی	Natural Language Understanding	پیش‌آموز شده	Pre-trained
دروازه‌ی خروجی	Output Gate	پیش‌زمینه	Context
دروازه‌ی فراموشی	Forget Gate	تابع فعال‌ساز	Activation Function
دروازه‌ی ورودی	Input Gate	تبدیلات خطی	Linear Transformation
دستیار دیجیتال	Digital Assistant	تحلیل احساسات	Sentiment Analysis
دور	Epoch	ترانهاده	Transpose
رمزگشا	Decoder	ترنسفورمر	Transformer

ماتریس قطری	Diagonal Matrix	Encoder
ماتریس مثلثی بالایی	Upper Triangular Matrix	Semantic Relations
ماتریس هم‌آیی کلمات	Word Co-occurance Matrix	Sub-Task
مجموعه‌داده	Dataset	Gesture
مجموعه‌ی متنی	Corpus	Time Series
مدل زبانی	Language Model	Memory Cell
معناشناسی	Semantics	Policy
مقدار	Value	Dialogue System
مکانیزم توجه	Attention Mechanism	Convolutional Neural Network
مؤلفه	Component	Named-Entity Recognition
میدان تصادفی شرطی	Conditional Random Field	Token ID
ناپیوسته	Discontinuous	Explicit
نامتوازن	Unbalanced	Dot Product
نحو	Syntax	Dialogue Act Classification
نزول گرادیان	Gradient Descent	Pragmatics
نزول گرادیان تصادفی	Stochastic Gradient Descent	Deep Double Descent
نشان کردن	Tokenize	WhiteSpace
نشانه	Token	Hate-Speech
نشانه ساز	Tokenizer	Key
نقشه‌ی ویژگی	Feature Map	Continuous Bag-Of-Words
هدف کلی	Overall Goal	Stride
هدف محور	Goal-Driven	Time-Step
هزینه‌ی عملیاتی	Operational Cost	Exploding Gradient
هسته	Kernel	Vanishing Gradient
واحد تولید زبان	Language Generation Unit	Node
واحد مدیریت مکالمه	Dialogue Management Unit	Speech
		Bottleneck
		Haptic

رمزنگار	Encoder
روابط معنایی	Semantic Relations
زیروظیفه	Sub-Task
ژست	Gesture
سری های زمانی	Time Series
سلول حافظه	Memory Cell
سیاست	Policy
سیستم گفت‌وگو	Dialogue System
شبکه‌ی عصبی کانولوشنی	Convolutional Neural Network
شناسایی موجودیت‌های نام‌دار	Named-Entity Recognition
شناسه‌ی نشان	Token ID
صریح	Explicit
ضرب نقطه‌ای	Dot Product
طبقه بندی قانون گفتگو	Dialogue Act Classification
عمل‌شناسی	Pragmatics
فرود عمیق دوگانه	Deep Double Descent
فضای خالی	WhiteSpace
کلام نفرت‌افکن	Hate-Speech
کلید	Key
کیسه‌ی کلمات ممتد	Continuous Bag-Of-Words
گام	Stride
گام زمانی	Time-Step
گرادیان انفجاری	Exploding Gradient
گرادیان محو شونده	Vanishing Gradient
گره	Node
گفتار	Speech
گلوگاه	Bottleneck
لمس	Haptic

Proposing a Model for Natural Language Understanding Using Deep Neural Networks

Mehrdad RafiePour
rafiepour@grad.kashanu.ac.ir

19/2/2023
Department of Electrical and Computer Engineering
University of Kashan, Isfahan 87317-53153, Iran
Degree: Master of Science (MSc)
Language: Farsi

Supervisor: Javad Salimi Sartakhti, Prof., salimi@kashanu.ac.ir.
Advisor: Fereshteh Dehghani, Prof., fdehghani@kashanu.ac.ir.

Abstract
With the increasing popularity of smartphones, adoption of dialogue-system-based tools have seen significant growth. Natural language understanding is a key component of dialogue systems, as it can be a major bottleneck in the dialogue system workflow. Intent-detection and slot-filling are the two main tasks in natural language understanding. Recurrent Neural Networks have been extensively explored to improve these tasks, but they have well-established weaknesses, namely gradient vanishing and high training time. Recently, Transformer was introduced to rectify said flaws. Moreover, we observed that only a few models further encode the pre-trained language model's output. In this thesis, CTran is propose. CTran is a novel encoder-decoder CNN-Transformer-based architecture designed for intent-detection and slot-filling. In the encoder, BERT is utilized as a word embedding. Then, several convolutional layers with different kernel sizes are used, which are then transposed and concatenated. In the last part of the encoder, stacked Transformer encoders are used to provide final output of the encoder. For the intent-detection decoder, self-attention is utilized followed by a linear layer. In the slot-filling decoder, the aligned Transformer decoder is introduced, which utilizes a heuristical diagonal mask. The diagonal mask provides access to encoder positions which correspond to each target token, and hides other positions. Finally, to evaluate the performance of the proposed model, it is applied on ATIS and SNIPS. The results show that CTran achieve better results than the current state-of-the-art in slot-filling on both datasets. Furthermore, two strategies, meaning language model as word embedding, and language model as an encoder, are compared. The results show that language model as word embedding strategy yields a better result.

Keywords
1-Deep Neural Networks, 2-Natural Language Processing, 3- Natural Language Understanding, 4-Intent-Detection, 5-Slot-Filling