ترنسفورمرها

مائده ابراهیمی

عرشیا جمالیان

پرهام مستجیر

استاد بهرام پروین

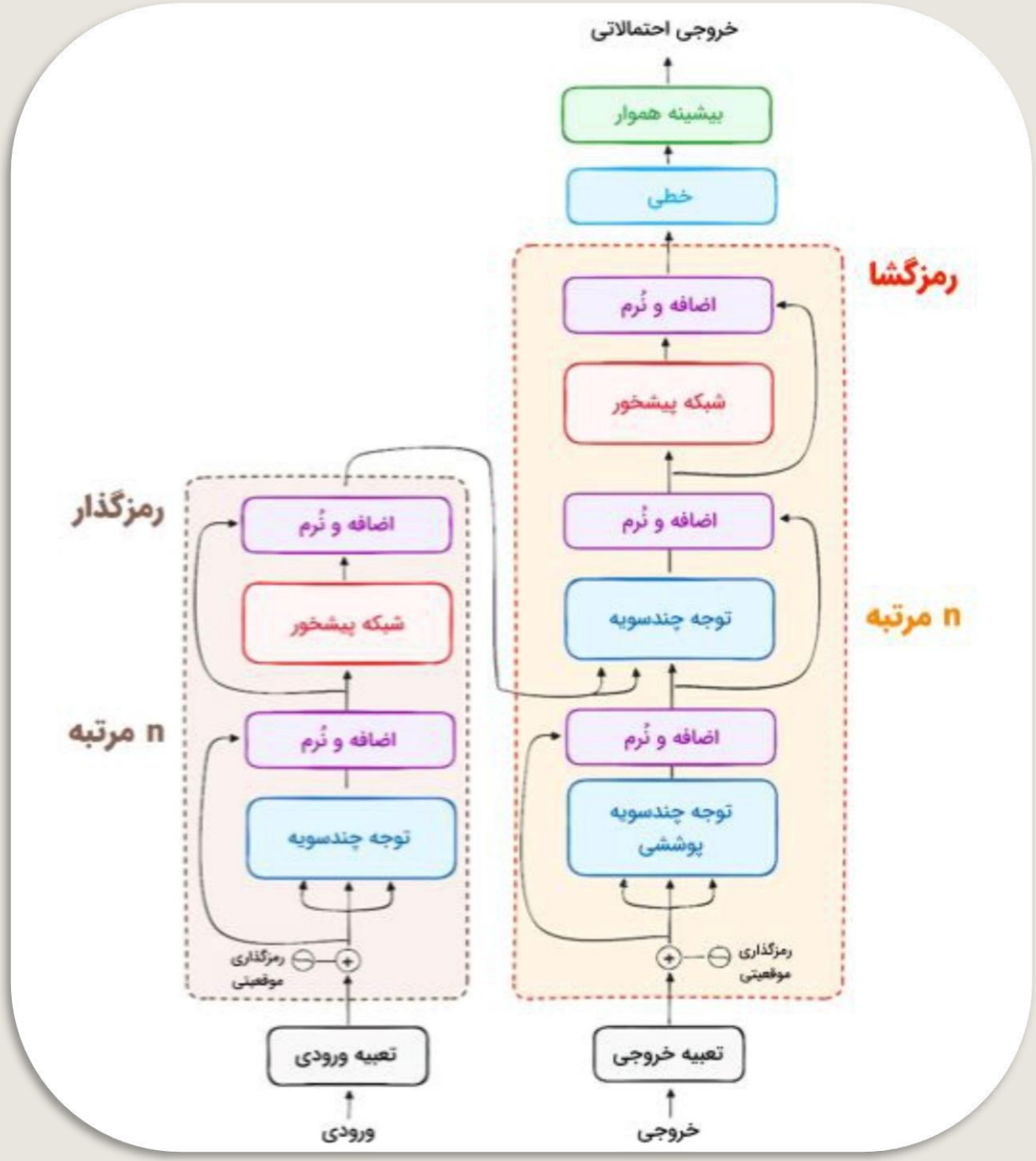
۱. مقدمه

دیپ لرنینگ (Deep Learning)

یادگیری عمیق شاخه‌ای از یادگیری ماشین است که بر استفاده از شبکه‌های عصبی با چندین لایه (معروف به شبکه‌های عمیق) تمرکز دارد. این روش‌ها قادرند ویژگی‌های پیچیده و انتزاعی را از داده‌های خام یاد بگیرند و در بسیاری از کاربردهای هوش مصنوعی عملکردی نزدیک به انسان ارائه دهند. ترجمه ماشینی و درک زبان (NLP) — سیستم‌های پیشنهاددهنده (مثلاً در نتفلیکس یا آمازون) .

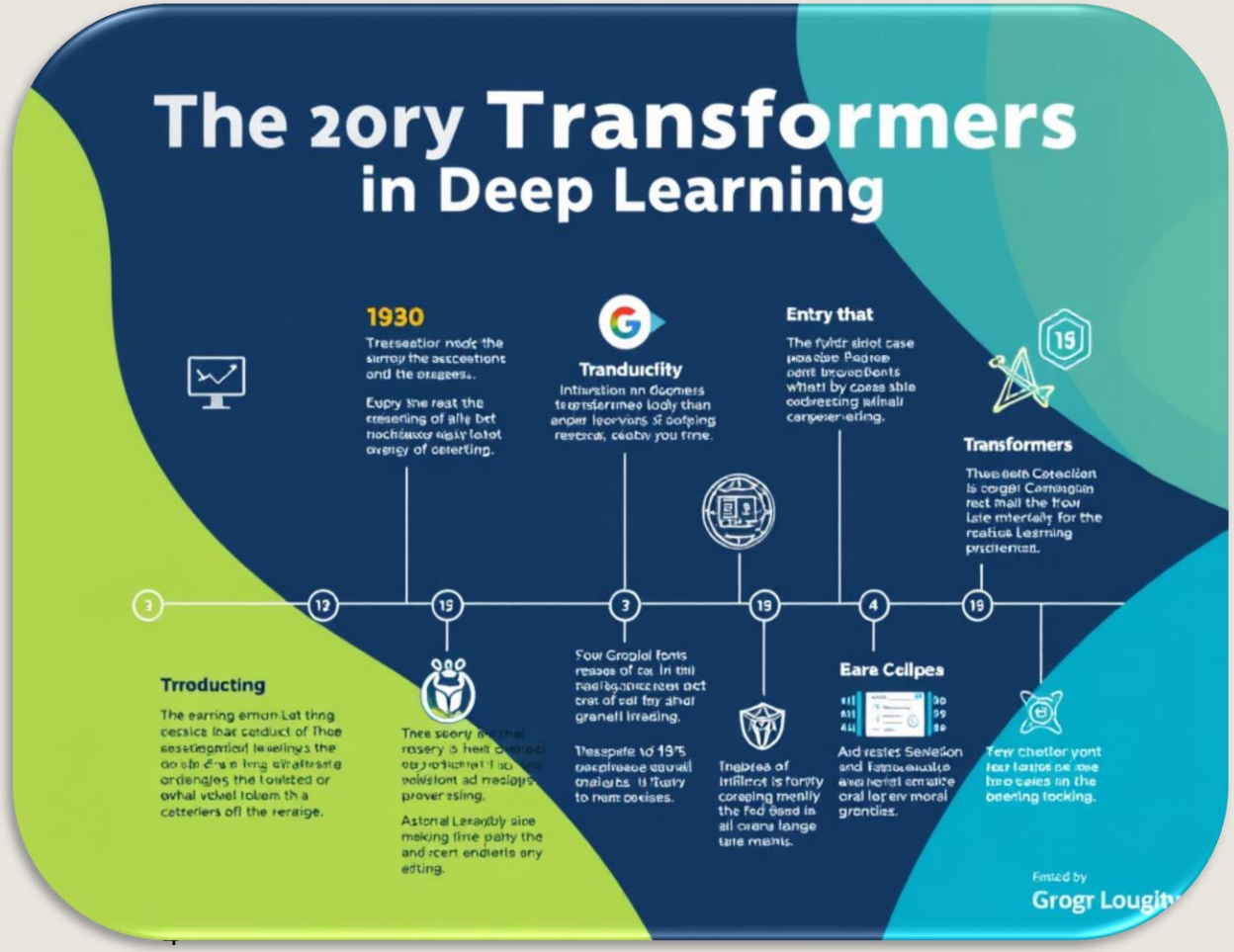
شبکه‌های عصبی (Artificial Neural Networks)

شبکه‌های عصبی مصنوعی از ساختار مغز انسان الهام گرفته‌اند. آن‌ها شامل مجموعه‌ای از "نورون‌ها" هستند که اطلاعات را در لایه‌های مختلف پردازش می‌کنند. در ابتدایی‌ترین شکل، این شبکه‌ها شامل لایه ورودی، چند لایه پنهان، و یک لایه خروجی هستند. پایه‌ای برای توسعه‌ی مدل‌های پیشرفته مثل RNN، CNN، و Transformer — آموزش‌پذیر با استفاده از داده‌های واقعی .



معرفی ترنسفور ها

ترنسفورمرها نوعی معماری شبکه عصبی قدرتمند هستند که انقلاب بزرگی در پردازش زبان طبیعی (NLP) و سایر وظایف ترتیب به ترتیب ایجاد کرده‌اند. این معماری‌ها در کاربردهای متنوعی از جمله ترجمه ماشینی، خلاصه‌سازی متن و مدل‌سازی زبان استفاده می‌شوند.



یکی از نوآوری‌های کلیدی ترنسفورمرها، قابلیت پردازش کل توالی‌های داده به‌صورت موازی است، به‌جای تکیه بر رویکرد تکراری که داده‌ها را مرحله به مرحله پردازش می‌کند. این کار از طریق استفاده از مکانیزم‌های توجه‌به‌خود (self-attention) امکان‌پذیر است، که به مدل اجازه می‌دهد تا به‌طور انتخابی بر روی بخش‌های مختلف توالی ورودی تمرکز کند

۲. تاریخچه

تا پیش از معرفی ترنسفورمرها، مدل‌های یادگیری عمیق برای پردازش زبان طبیعی (NLP) عمدتاً مبتنی بر شبکه‌های بازگشتی (RNN) و به‌خصوص LSTM بودند. این مدل‌ها اگرچه در درک دنباله‌ها عملکرد قابل‌قبولی داشتند، اما با چالش‌هایی چون محدودیت در پردازش موازی، افت کیفیت در دنباله‌های طولانی و هزینه‌های محاسباتی بالا مواجه بودند.

در سال ۲۰۱۷، مقاله‌ای انقلابی با عنوان “Attention is All You Need” توسط تیمی از محققان گوگل منتشر شد. این مقاله برای اولین بار معماری جدیدی را معرفی کرد که به طور کامل از مکانیزم Attention استفاده می‌کرد و هیچ‌گونه ساختار بازگشتی در آن نبود. این معماری جدید با نام ترنسفورمر (Transformer) شناخته شد.

۳. کاربردهای ترنسفورمرها

مدل‌های ترنسفورمر، پس از معرفی اولیه در حوزه‌ی پردازش زبان طبیعی، به سرعت جای خود را در طیف وسیعی از حوزه‌های یادگیری عمیق باز کردند. انعطاف‌پذیری معماری آن‌ها، به‌ویژه قابلیت مدل‌سازی وابستگی‌های بلندمدت در داده‌ها، باعث شده که ترنسفورمرها از ابزارهای محوری در پروژه‌های تحقیقاتی و صنعتی تبدیل شوند. برخی از مهم‌ترین حوزه‌های کاربردی آن‌ها :

الف) پردازش زبان طبیعی (Natural Language Processing - NLP)

بزرگ‌ترین جهش عملکردی ترنسفورمرها در حوزه NLP رقم خورد. با معرفی ترنسفورمرها، مدل‌های مبتنی بر RNN و LSTM به‌سرعت جای خود را به معماری‌های attention-based دادند. مدل‌هایی مانند BERT (Bidirectional Encoder Representations from Transformers) و GPT (Generative Pre-trained Transformer) توانستند در بسیاری از وظایف زبانی به دقتی برسند که پیش از آن بی‌سابقه بود.

در ترجمه ماشینی، مدل‌هایی نظیر mBART و T5 جایگزین سیستم‌های قدیمی مانند Google Neural Machine Translation شدند و به دقت بسیار بالاتری دست یافتند. در پاسخ به سوال، مدل‌های BERT fine-tuned شده به عنوان موتورهای پاسخ‌دهی در موتورهای جستجو و سیستم‌های گفت‌وگومحور (مانند Siri و Google Assistant) استفاده می‌شوند.

همچنین، در حوزه‌ی خلاصه‌سازی متن و تولید زبان طبیعی (NLG)، ترنسفورمرها ساختارهای پیچیده زبان را بهتر درک کرده و خروجی‌هایی با روانی و انسجام بالا تولید می‌کنند. این ویژگی در تولید گزارش خودکار، خلاصه‌سازی متون علمی، و تولید محتوای خودکار در رسانه‌ها کاربرد دارد.

ب) بینایی ماشین (Computer Vision)

با معرفی مدل Vision Transformer (ViT)، استفاده از ترنسفورمرها در بینایی ماشین نیز آغاز شد. برخلاف شبکه‌های کانولوشنی (CNN) که روی پیوستگی فضایی پیکسل‌ها تمرکز می‌کنند، ترنسفورمرها تصویر را به قطعاتی (patches) تقسیم می‌کنند و هر قطعه را مشابه توکن‌های متنی پردازش می‌کنند. این رویکرد باعث می‌شود مدل، دید وسیع‌تری نسبت به ساختار کلی تصویر داشته باشد.

در حوزه‌ی دسته‌بندی تصاویر، مدل ViT توانسته در بسیاری از دیتاست‌های استاندارد (مانند ImageNet) نتایج قابل رقابتی با CNNها و حتی بهتر را ثبت کند. همچنین در وظایفی مانند تشخیص اشیا (Object Detection) و تقسیم‌بندی تصاویر (Semantic Segmentation)، نسخه‌های اصلاح‌شده مانند DETR (Detection Transformer) عملکرد چشمگیری از خود نشان داده‌اند.

علاوه بر تصاویر ایستا، ترنسفورمرها در تحلیل ویدیو نیز با مدل‌هایی مانند TimeSformer و Video Swin Transformer توانسته‌اند روابط زمانی-مکانی پیچیده بین فریم‌های ویدیو را مدل‌سازی کنند. این امر در کاربردهایی نظیر تشخیص فعالیت انسانی، پایش امنیتی، و خلاصه‌سازی ویدیویی اهمیت بالایی دارد.

ج) سایر حوزه‌ها: صدا، سری‌های زمانی و زیست‌شناسی محاسباتی

کاربرد ترنسفورمرها به متن و تصویر محدود نمی‌شود. در پردازش صوت، مدل‌هایی مانند Wav2Vec 2.0، Whisper و HuBERT از ترنسفورمرها برای مدل‌سازی ویژگی‌های پیچیده آوایی استفاده می‌کنند. این مدل‌ها با بهره‌گیری از داده‌های صوتی عظیم و بدون برچسب، توانسته‌اند سیستم‌هایی بسیار دقیق برای تبدیل گفتار به متن یا حتی تشخیص احساسات صوتی ارائه دهند.

در حوزه‌ی مدل‌سازی سری‌های زمانی مانند پیش‌بینی بازار سهام، تحلیل داده‌های حسگر یا پایش سلامت بیماران، ترنسفورمرها با قابلیت یادگیری وابستگی‌های بلندمدت، دقت بالاتری نسبت به مدل‌های کلاسیک ARIMA یا حتی LSTMها ارائه داده‌اند.

در زیست‌شناسی محاسباتی، مدل‌هایی مانند AlphaFold که از معماری‌های مبتنی بر attention استفاده می‌کنند، توانسته‌اند ساختار سه‌بعدی پروتئین‌ها را با دقت بی‌سابقه‌ای پیش‌بینی کنند. این پیشرفت، افق‌های جدیدی در داروسازی و درک عملکرد مولکول‌های زیستی گشوده است.

۴. مزایا و چالش های ترنسفورمرها

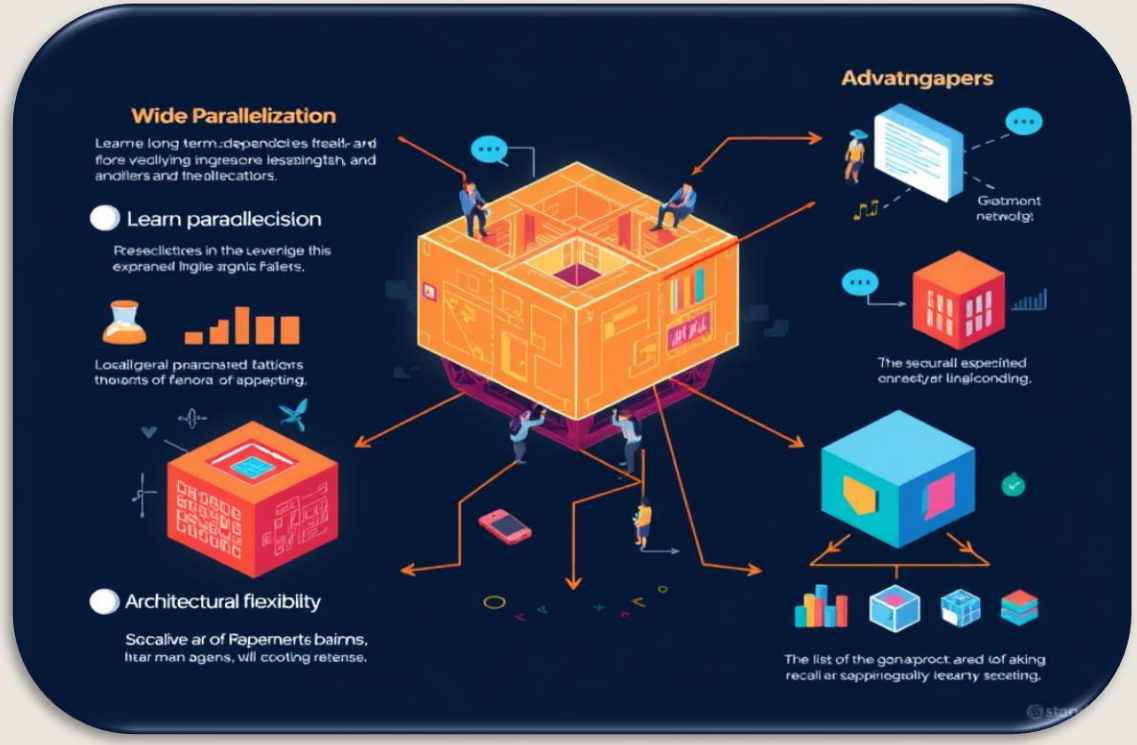
مزایا

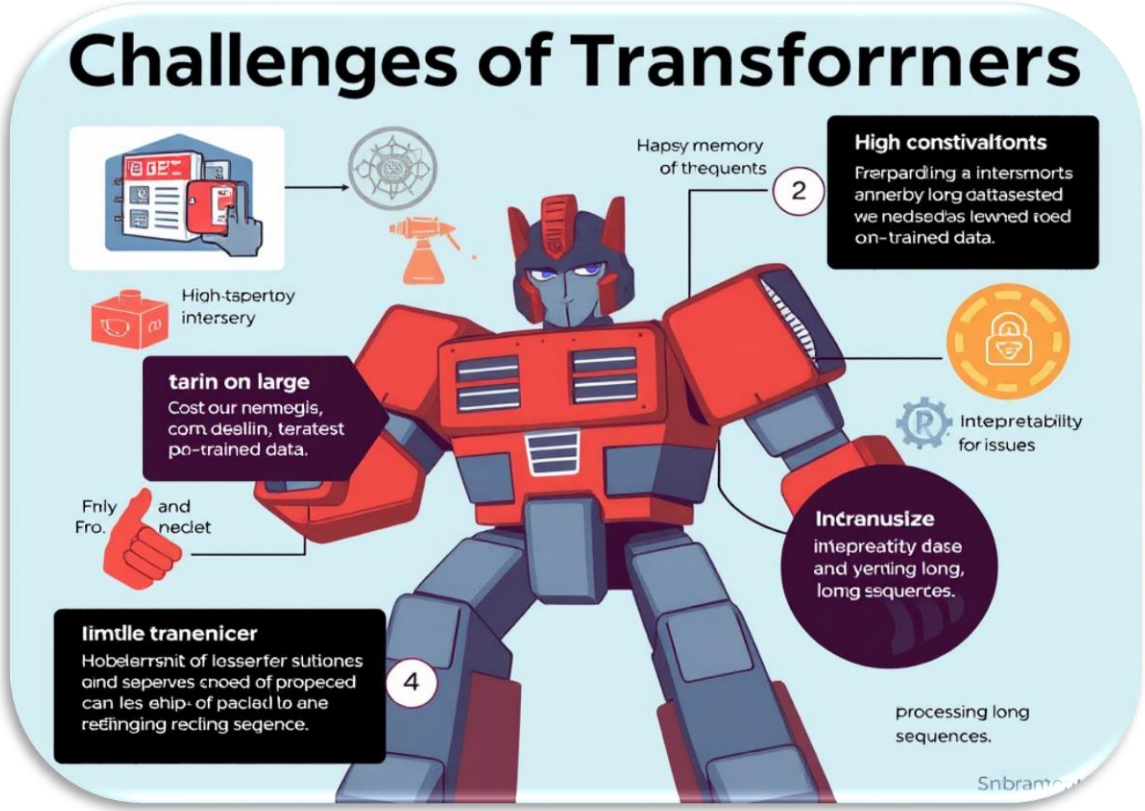
۱. **موازی‌سازی گسترده:** کل توالی/ورودی یکجا پردازش می‌شود که زمان آموزش را به‌طور چشمگیری کاهش می‌دهد.

۲. **یادگیری وابستگی‌های بلندمدت:** مکانیزم attention می‌تواند مستقیماً میان هر دو نقطه از توالی ارتباط برقرار کند.

۳. **انعطاف‌پذیری معماری:** با تغییر ساده توابع attention و feedforward می‌توان مدل را برای متن، تصویر، صوت یا سری‌های زمانی تنظیم کرد.

۴. **مقیاس‌پذیری:** ساختار بلوکی ترنسفورمر به‌سادگی در عمق و عرض افزایش پیدا می‌کند و با داده‌های عظیم همخوانی دارد.





چالش‌ها (معایب)

۱. هزینه‌های محاسباتی و نیاز به حافظه زیاد

ترنسفورمرها به‌ویژه در مقیاس‌های بزرگ، به دلیل پیچیدگی محاسباتی و نیاز به حافظه زیاد در پردازش توالی‌های طولانی، هزینه‌های اجرایی بالایی دارند. در حالی که مدل‌های کوچکتر و ساده‌تر مانند RNN و CNN حافظه کمتری مصرف می‌کنند، ترنسفورمرها نیاز به سخت‌افزارهای قدرتمند دارند که به مراتب گران‌تر و مصرف انرژی بیشتری دارند.

۲. آموزش بر روی داده‌های بزرگ و نیاز به داده‌های پیش‌آموزش‌شده

ترنسفورمرها به داده‌های آموزش وسیع نیاز دارند تا عملکرد مناسبی در وظایف مختلف داشته باشند. به عنوان مثال، مدل‌هایی مانند GPT-3 نیاز به میلیاردها نمونه از داده دارند که جمع‌آوری و برچسب‌گذاری این داده‌ها هزینه‌بر است.

۳. مشکلات تفسیرپذیری و شفافیت مدل

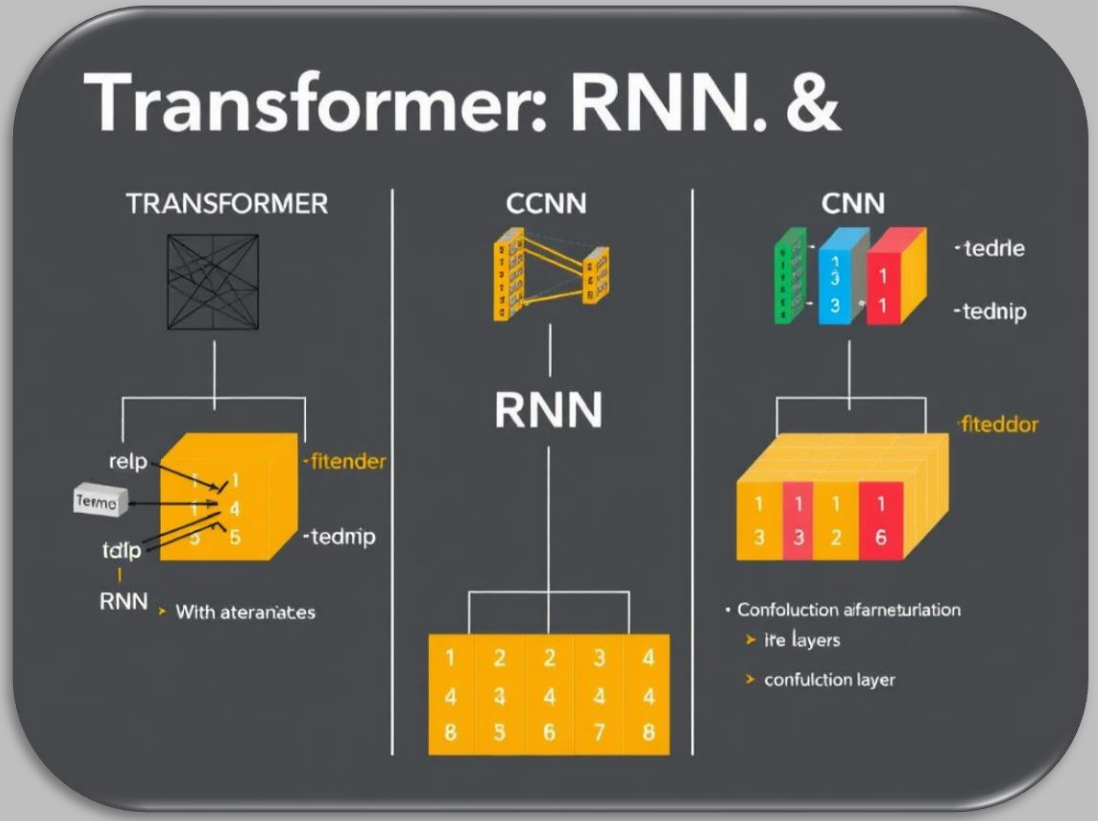
یکی از بزرگ‌ترین چالش‌ها در ترنسفورمرها، کمبود تفسیرپذیری است. این مدل‌ها به دلیل پیچیدگی زیاد و تعداد پارامترهای بالا، شفافیت کافی ندارند، و به‌سختی می‌توان فهمید که چرا مدل به نتیجه خاصی رسیده است.

۴. محدودیت‌های در پردازش توالی‌های طولانی

مدل‌های ترنسفورمر با محدودیت‌هایی در پردازش توالی‌های بسیار طولانی مواجه‌اند، زیرا تعداد محاسبات مورد نیاز برای هر توکن در طول توالی با افزایش طول توالی به‌طور نمایی افزایش می‌یابد.

۵. مقایسه با مدل‌های دیگر

در این بخش، ترنسفورمرها را با دو معماری متداول دیگر—شبکه‌های بازگشتی (RNN) و شبکه‌های کانولوشنی—(CNN) از جنبه‌های مختلف مقایسه کرده و نقاط قوت و ضعف هرکدام را تحلیل می‌کنیم.



مقایسه ترنسفورمر، RNN وCNN

ویژگی / مدل	ترنسفورمر(Transformer)	شبکه بازگشتی (RNN / LSTM / GRU)	شبکه کانولوشنی(CNN)
پردازش توالی	موازی، بدون ترتیب ذاتی	گام‌به‌گام، به ترتیب زمانی	محدود به پنجره محلی، نه به‌صورت صریح توالی‌محور
یادگیری وابستگی بلندمدت	بسیار مؤثر) از طریق attention)	با دشواری و مشکل گرادیان ناپدید شونده	ضعیف، نیازمند لایه‌های زیاد برای درک روابط دور
موازی‌سازی	بسیار بالا	بسیار پایین (وابسته به ترتیب)	بالا (در پردازش بلوکی و فیلترهای موازی)
مقیاس‌پذیری	عالی (افزایش عمق/عرض ساده است)	محدود (افزایش عمق منجر به مشکلات یادگیری می‌شود)	قابل قبول، اما در لایه‌های عمیق پیچیده‌تر می‌شود
استفاده در بینایی ماشین	مدل‌های ViT با نیاز به داده زیاد	کاربرد محدود (اغلب در توالی‌های زمانی)	بسیار مؤثر و رایج در تصویرپردازی
نیاز به داده آموزشی	بسیار زیاد (مخصوصاً مدل‌های بزرگ)	متوسط تا زیاد (بسته به عمق)	کمتر از ترنسفورمرها، در بسیاری از وظایف کفایت می‌کند
تفسیرپذیری	نسبتاً کم	نسبتاً بیشتر (به‌خصوص در مدل‌های ساده)	قابل قبول، بخصوص در لایه‌های اولیه

۶. پیاده‌سازی مدل ترنسفورمر

راهنمای گام‌به‌گام برای پیاده‌سازی مدل ترنسفورمر در پایتون، با استفاده از کتابخانه یادگیری عمیق TensorFlow.

* کتابخانه TensorFlow یکی از محبوب‌ترین و قدرتمندترین ابزارهای یادگیری عمیق(Deep Learning) است که توسط شرکت Google Brain توسعه داده شده است.

TensorFlow یک کتابخانه متن‌باز (Open-source) برای محاسبات عددی و یادگیری ماشین است که در ابتدا در نوامبر سال ۲۰۱۵ توسط گوگل معرفی شد. این کتابخانه عمدتاً برای طراحی، آموزش و اجرای شبکه‌های عصبی مصنوعی به کار می‌رود، اما دامنه استفاده از آن فراتر از یادگیری عمیق است.)

پیش‌پردازش داده‌ها

پیش‌پردازش داده‌ها به مجموعه‌ای از تکنیک‌ها گفته می‌شود که روی داده‌های خام اعمال می‌شوند تا آن‌ها را برای تحلیل یا آموزش مدل‌های یادگیری ماشین آماده کنند. این مرحله یکی از مهم‌ترین مراحل در فرآیند یادگیری ماشین است، چرا که کیفیت داده تأثیر مستقیم بر عملکرد مدل دارد.

بارگذاری داده‌ها

در پردازش زبان طبیعی(NLP) ، بارگذاری داده‌ها اولین مرحله است که شامل وارد کردن داده‌های متنی به محیط برنامه‌نویسی می‌شود. این داده‌ها معمولاً از منابع مختلفی مانند فایل‌های متنی، پایگاه‌های داده یا APIها به دست می‌آیند.

*(API مخفف Application Programming Interface به معنای «رابط برنامه‌نویسی کاربردی» است. API مجموعه‌ای از قوانین و پروتکل‌ها است که به برنامه‌های نرم‌افزاری مختلف اجازه می‌دهد با یکدیگر ارتباط برقرار کنند)

تجسم داده‌ها (Visualization)

تجسم داده‌ها به ما کمک می‌کند تا الگوها و روندهای موجود در داده‌ها را بهتر درک کنیم. این مرحله شامل استفاده از نمودارها و گراف‌ها برای نمایش توزیع کلمات، طول جملات و سایر ویژگی‌های متنی است.

توکن‌سازی

توکن‌سازی **فرآیند تقسیم متن به واحدهای فردی** به نام توکن‌ها است. این توکن‌ها معمولاً کلمات هستند، اما بسته به کاربرد، می‌توانند عبارات، زیرکلمات یا کاراکترها نیز باشند. توکن‌سازی یک مرحله اساسی در بسیاری از وظایفNLP ، مانند مدل‌سازی زبان، ترجمه ماشینی و طبقه‌بندی متن است. پس از توکن‌سازی، متن می‌تواند به یک نمای عددی تبدیل شود که به عنوان ورودی برای مدل یادگیری ماشینی استفاده می‌شود.

پدینگ

در مدل‌های یادگیری ماشینی، معمولاً ورودی‌ها باید دارای طول ثابتی باشند. اگر توالی‌های ورودی طول‌های متفاوتی داشته باشند، نیاز است که با مقداری جایگزین (معمولاً ۰) پدینگ شوند تا همه به یک طول یکسان برسند. این فرآیند پدینگ نامیده می‌شود. پدینگ اطمینان می‌دهد که ورودی‌های مدل از یک اندازه یکسان برخوردارند، که برای آموزش کارآمد مدل ضروری است.

توابع کمکی(Helper Functions)

کدگذاری موقعیتی (Positional Encoding)

کدگذاری موقعیتی (Positional Encoding)یکی از مفاهیم کلیدی در معماری ترنسفورمرهاست که به مدل کمک می‌کند تا ترتیب عناصر موجود در یک توالی را درک کند.

مانی به‌صورت طبیعی حفظ می‌شود، ترنسفورمرها به‌صورت موازی با
ظ اطلاعات مربوط به **ترتیب واژه‌ها یا نشانه‌ها در جمله**، باید موقعیت

. ایده اصلی این است که برای هر موقعیت pos و هر بُعد در بردار

چرا از سینوس و کسینوس استفاده می‌شود؟

- ترتیب کلمات در ترنسفورمر حفظ نمی‌شه، پس باید به مدل بگیم هر کلمه کجاست.
- سینوس و کسینوس الگوهای متفاوت و قابل درکی برای هر موقعیت تولید می‌کنن.
- این توابع باعث می‌شن مدل فاصله بین کلمات رو هم بفهمه.
- چون این کدگذاری ثابت و بدون یادگیریه، ساده و سریع انجام می‌شه.

$$PE_{(pos,2i)} = sin\left(\frac{pos}{10000^{\frac{2i}{d}}}\right) \qquad (1)$$

$$PE_{(pos,2i+1)} = cos\left(\frac{pos}{10000^{\frac{2i}{d}}}\right) \qquad (2)$$

در اینجا:

pos نشان‌دهنده موقعیت (جایگاه) ورودی در دنباله است،

i نمایانگر شماره (اندیس) بُعد در بردار تعبیه‌سازی (embedding) است، d بیانگر تعداد کل ابعاد مدل است.

ترنسفورمر از دو بخش اصلی تشکیل شده است: **رمزگذار (Encoder)** و **رمزگشا (Decoder)**

رمزگذار، ابتدا یک دنباله‌ی ورودی را دریافت می‌کند و آن را به یک سری بردار معنایی یا «نمایش‌های پنهان» تبدیل می‌کند. سپس **رمزگشا** با استفاده از این نمایش‌های پنهان، خروجی موردنظر را به‌صورت گام‌به‌گام تولید می‌کند.

هر کدام از این دو بخش (Encoder و Decoder)، شامل چندین لایه هستند که هر لایه از دو جزء کلیدی تشکیل شده است:

- **توجه چندسری (Multi-head Self-Attention)** برای درک وابستگی بین بخش‌های مختلف دنباله.
- **شبکه عصبی پیش‌خور (Feed-Forward Neural Network)** برای پردازش ویژگی‌های به‌دست‌آمده.

◆ منظور از **نمایش‌های پنهان**، بردارهایی هستند که در هر لایه از مدل ساخته می‌شوند و اطلاعات معنی‌دار ورودی را به‌صورت خلاصه‌شده نگه می‌دارند تا مدل بتواند آن‌ها را بهتر پردازش کند.

Encoder

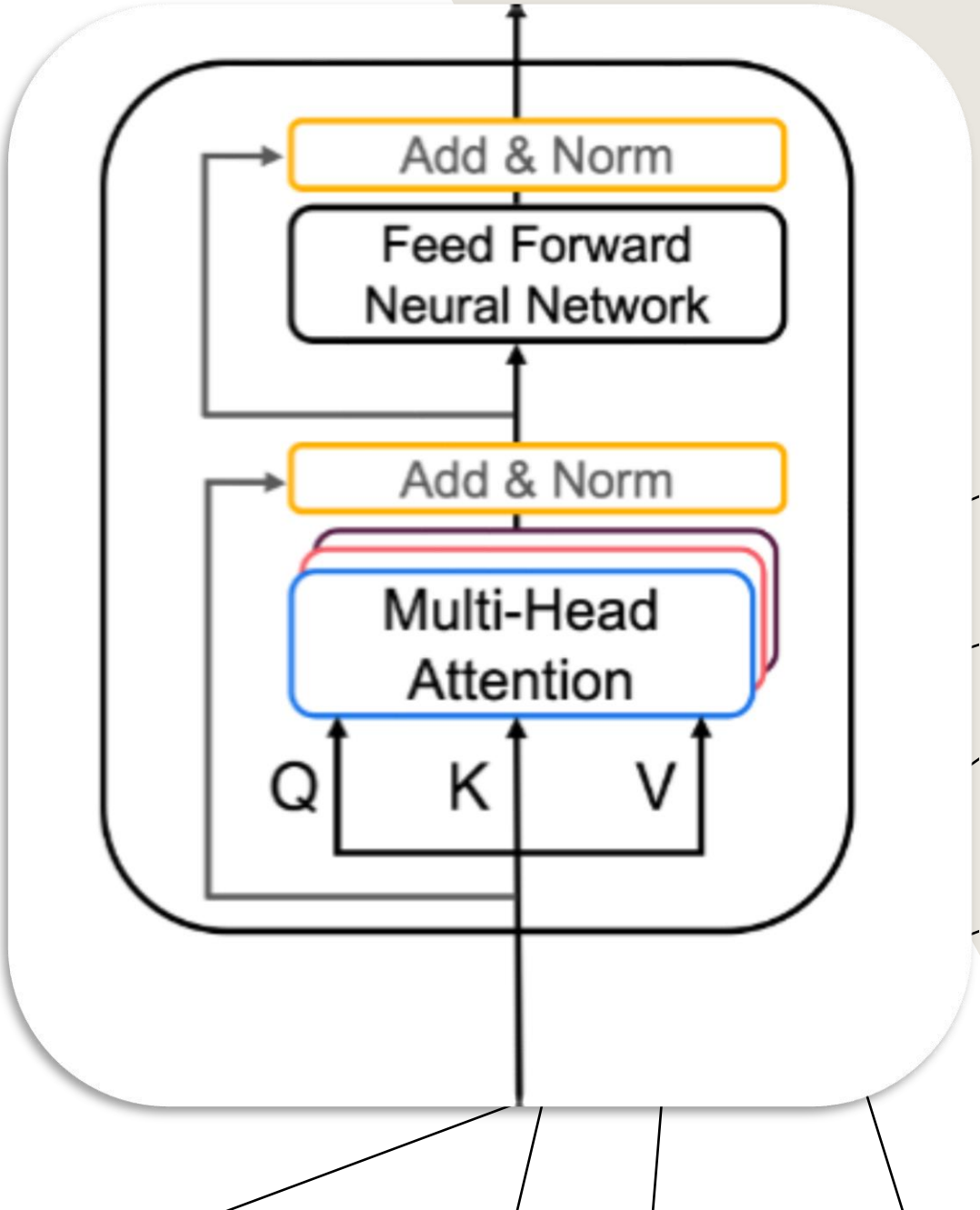
در معماری ترنسفورمر، Encoder وظیفه دارد ورودی‌ها را تحلیل کرده و از آن‌ها ویژگی‌های مهم (نمایش‌های پنهان) استخراج کند.
ابتدا، ورودی‌های متنی به بردارهای عددی (embedding) تبدیل می‌شوند. سپس به این بردارها کدگذاری موقعیتی (Positional Encoding)اضافه می‌شود تا ترتیب واژه‌ها مشخص شود، چون ترنسفورمر به‌تنهایی ترتیب را نمی‌فهمد.
بعد از این مرحله، داده‌ها وارد چند لایه انکودر می‌شوند. هر لایه انکودر شامل دو قسمت اصلی است:

1.توجه چندسر :(Multi-head Attention)
مدل با این بخش می‌تواند هم‌زمان به قسمت‌های مختلف جمله نگاه کند و روابط بین واژه‌ها را پیدا کند. این توجه با استفاده از سه بردار انجام می‌شود:

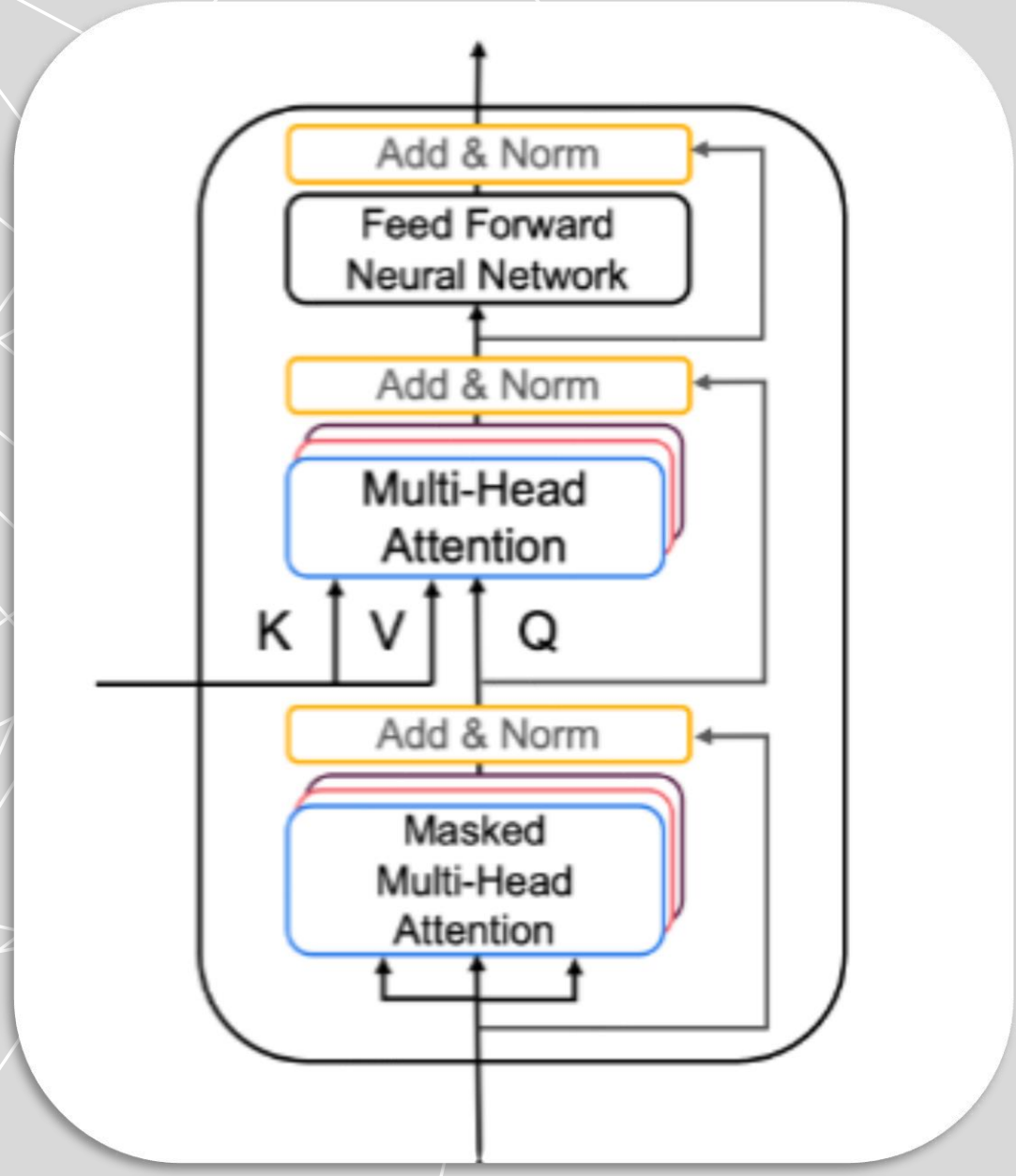
1. Query(پرسش)
2. Key(کلید)
3. Value(مقدار)

2.شبکه‌ی پیش‌خور Feed Forward Network – FFN
یک شبکه عصبی ساده که روی هر موقعیت جداگانه کار می‌کند و ویژگی‌ها را بهتر ترکیب می‌کند.

در پایان هر بخش، از نرمال‌سازی لایه‌ای (Layer Normalization) و اتصال باقیمانده (Residual Connection)استفاده می‌شود تا آموزش بهتر و پایدارتر انجام شود.
این لایه‌ها چند بار تکرار می‌شوند. در نهایت، خروجی انکودر به Decoderفرستاده می‌شود تا خروجی نهایی (مثل ترجمه یا پاسخ) تولید شود.



Decoder



در معماری ترنسفورمر، Decoder بخشی است که وظیفه تولید خروجی نهایی را دارد؛ مثلاً ترجمه یک جمله یا تولید پاسخ.
فرآیند دیکودر به این صورت است:

1. ورودی دیکودر (مثلاً کلمات قبلی خروجی) به بردار عددی (embedding) تبدیل می‌شود. سپس، اطلاعات موقعیت‌یابی (Positional Encoding) به آن اضافه می‌شود تا ترتیب کلمات مشخص باشد.
2. سپس داده‌ها وارد چندین لایه دیکودر می‌شوند. هر لایه شامل سه بخش اصلی است:
   1. توجه چندسر با ماسک (Masked Multi-head Self-Attention)
      این بخش فقط اجازه می‌دهد هر کلمه به کلمات قبلی خودش توجه کند، نه کلمات آینده. این کار برای حفظ ترتیب در تولید خروجی ضروری است.
   2. توجه چندسر به خروجی انکودر (Encoder-Decoder Attention)
      در این مرحله، دیکودر به خروجی‌های انکودر نگاه می‌کند تا تصمیم بگیرد که به کدام قسمت از ورودی اصلی (مثلاً جمله اولیه) توجه کند.
   3. شبکه‌ی پیش‌خور (Feed Forward Network)
      ویژگی‌های هر موقعیت به‌صورت جداگانه پردازش می‌شود.
3. بعد از هر بخش، از نرمال‌سازی لایه‌ای (Layer Normalization) و اتصال باقیمانده (Residual Connection) استفاده می‌شود.

در پایان، خروجی آخرین لایه از طریق یک لایه خطی و سپس تابع Softmax به یک توزیع احتمال روی واژگان تبدیل می‌شود تا کلمه بعدی انتخاب شود.
این فرایند برای هر کلمه تکرار می‌شود تا کل جمله خروجی ساخته شود.

Masking

در پیاده‌سازی ترنسفورمرها، **ماسک‌گذاری (Masking)** نقشی کلیدی در کنترل دسترسی هر توکن به سایر توکن‌ها دارد. دو نوع اصلی ماسک وجود دارد:

1.ماسک پَدینگ:(Padding Mask)

زمانی استفاده می‌شود که جملات ورودی طول‌های متفاوتی دارند. برای یکسان‌سازی طول‌ها، از توکن‌های [PAD] استفاده می‌شود، اما این توکن‌ها اطلاعات مفیدی ندارند. ماسک پَدینگ جلوی توجه مدل به این بخش‌های بی‌معنا را می‌گیرد.

2.ماسک پیش‌نگر:(Look-ahead Mask)

در بخش دیکودر، هنگام آموزش، از این ماسک استفاده می‌شود تا مدل فقط به توکن‌های قبلی یا جاری نگاه کند و نه به توکن‌های بعدی. این کار برای شبیه‌سازی حالت پیش‌بینی واقعی (در زمان استنتاج) انجام می‌شود تا مدل نتواند از آینده تقلب کند. به این ترتیب، **نشت اطلاعات (information leakage)** جلوگیری می‌شود.

این ماسک‌ها معمولاً با قرار دادن مقادیر بسیار منفی (مانند منفی بی‌نهایت) در مکان‌های نامجاز ماتریس توجه اعمال می‌شوند. این مقادیر پس از اعمال تابع softmaxباعث می‌شوند وزن توجه به آن نقاط تقریباً صفر شود.

برای ساخت ماسک پیش‌نگر، از یک **ماتریس مثلثی پایین** استفاده می‌شود که فقط به توکن‌های قبلی اجازه توجه می‌دهد. این ماتریس معمولاً در قالب یک tf.Tensorبازگشت داده می‌شود. این فرآیند به مدل کمک می‌کند در طول آموزش، رفتار مرحله‌به‌مرحله تولید توکن را به‌درستی یاد بگیرد.

تابع $create\_masks$

تابع $create\_masks$ در پیاده‌سازی ترنسفورمر برای ساختن **سه نوع ماسک کلیدی** به کار می‌رود که برای عملکرد صحیح مکانیزم توجه ضروری هستند:

- **ماسک پدینگ انکودر ($Encoder\ Padding\ Mask$)**

مشخص می‌کنه کدوم قسمت‌های ورودی انکودر توکن‌های [PAD] هستند تا مدل اون‌ها رو نادیده بگیره.

- **ماسک پیش‌نگر ($Look-ahead\ Mask$)**

در دیکودر استفاده می‌شه تا از نگاه کردن به توکن‌های آینده جلوگیری کنه. باعث می‌شه مدل فقط از توکن‌های قبلی برای پیش‌بینی استفاده کنه.

- **ماسک پدینگ دیکودر ($Decoder\ Padding\ Mask$)**

درست مثل انکودر، این ماسک کمک می‌کنه مدل توکن‌های [PAD] رو در ورودی دیکودر نادیده بگیره.

- **ماسک ترکیبی ($Combined\ Mask$)**

ترکیبی از ماسک پیش‌نگر و ماسک پدینگ دیکودر هست. کمک می‌کنه مدل هم توکن‌های آینده رو نبینه و هم توکن‌های [PAD] رو نادیده بگیره.

در مجموع، این ماسک‌ها به مدل کمک می‌کنند تا ساختار ترتیبی توالی، محدودیت‌های ورودی، و پردازش تدریجی دیکودر را رعایت کند و عملکرد دقیق‌تری در وظایف زبانی ارائه دهد.

تابع توجه ضرب نقطه‌ای مقیاس شده (Scaled Dot-Product Attention)

تابع **توجه ضرب نقطه‌ای مقیاس‌شده** (Scaled Dot-Product Attention) یکی از اجزای کلیدی در معماری ترنسفورمر (Transformer)است که در مدل‌هایی مانند BERT و GPTبه کار می‌رود. این تابع تعیین می‌کند که یک کلمه (یا عنصر) در دنباله ورودی، هنگام پردازش، باید به کدام کلمات دیگر بیشتر "توجه" کند.

مفهوم پایه‌ای:

در این تابع، ما سه بردار داریم:

- **Query(پرسش)**: نماینده آن چیزی است که به دنبال اطلاعات درباره‌اش هستیم.
- **Key(کلید)**: نماینده ویژگی‌های دیگر کلمات.
- **Value(ارزش)**: همان اطلاعاتی است که در نهایت برای ادامه پردازش استفاده می‌شود.

فرمول ریاضی:

$$\text{softmax}\left(\frac{^{T}QK}{\sqrt{_{k}d}}\right)V = \text{Attention}(Q, K, V)$$

وزن‌های توجه نرمال‌شده یعنی اینکه مدل تصمیم می‌گیره به **کدوم کلمه‌ها بیشتر دقت کنه** و به هر کلمه **چقدر اهمیت بده**.
مدل اول شباهت بین کلمات رو حساب می‌کنه (با ضرب نقطه‌ای پرسش و کلید)، بعد با **تابع Softmax**این شباهت‌ها رو به **درصد اهمیت** تبدیل می‌کنه، طوری که همه وزن‌ها جمع‌شون بشه ۱. این درصدها همون وزن‌های توجه نرمال‌شده هستن.
به زبان ساده:
مدل می‌فهمه از بین همه کلمات، باید بیشتر حواسش به کدوم‌ها باشه.

مراحل اجرای تابع:

- **محاسبه شباهت بین Query و Key**
- برای هر کلمه، بردار پرسش آن با بردار کلید

چقدر آن کلمات به هم مرتبط هستند.

- **مقیاس کردن (Scaling)**
- مقدار حاصل از ضرب نقطه‌ای، ممکن است بزرگ باشد و باعث شود مقدار Softmax خیلی شیب‌دار شود. بنابراین آن را تقسیم بر **جذر ابعاد بردار کلید** می‌کنند.($\sqrt{d_k}$)
- **اعمال Softmax**
- خروجی مرحله قبل از طریق تابع Softmax عبور می‌کند تا **وزن‌های توجه نرمال‌شده** * به دست آید. این وزن‌ها تعیین می‌کنند که کدام کلمات باید بیشتر مورد توجه قرار بگیرند.
- **ترکیب با Value**
- در نهایت، وزن‌های توجه به بردارهای Value ضرب می‌شوند و یک **مجموع وزنی از اطلاعات** تولید می‌شود.

توجه چندسر(MultiHeadAttention)

توجه چندسر (Multi-Head Attention)یکی از اجزای کلیدی در ساختار ترنسفورمرهاست که نقش مهمی در قدرت درک و یادگیری مدل دارد. در این مکانیزم، به جای اجرای تنها یک بار عملیات توجه (Attention)، این فرآیند به‌صورت موازی و چندباره توسط «سرهای مختلف توجه» انجام می‌شود.

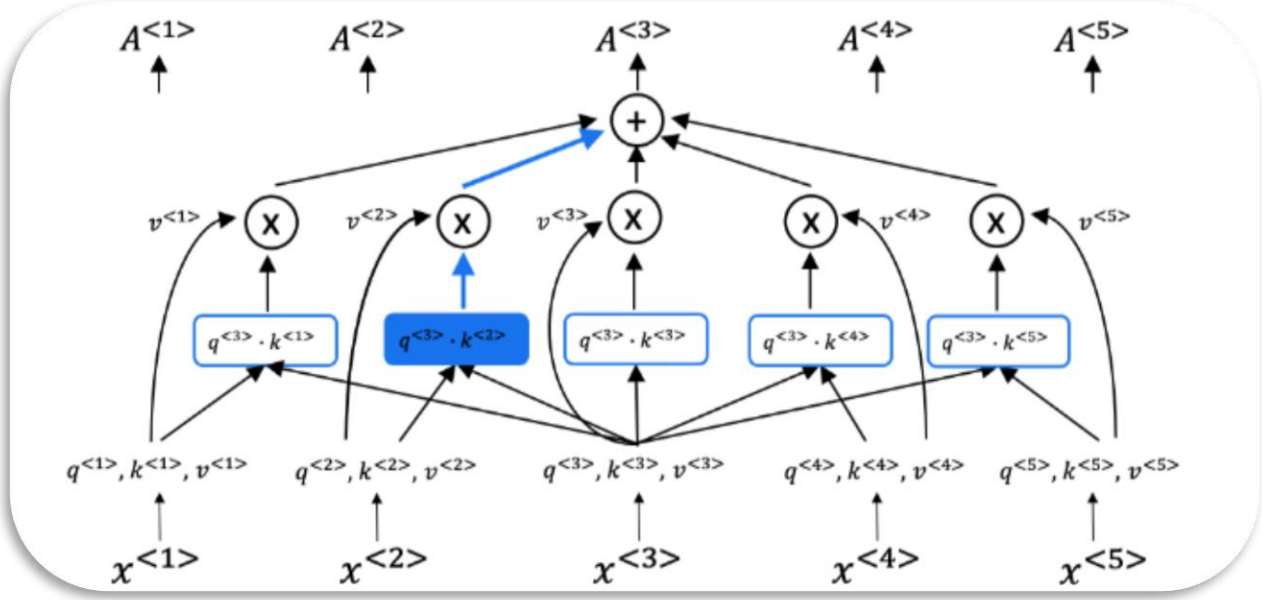
برای هر سر، ابتدا ورودی به سه بردار تبدیل می‌شود:

- **Q (Query)**یا پرسش
- **K (Key)**یا کلید
- **V (Value)**یا مقدار

سپس بر اساس این سه بردار، **توجه ضرب نقطه‌ای مقیاس‌شده (Scaled Dot-Product Attention)** محاسبه می‌شود. هر سر پارامترهای مستقلی دارد، بنابراین هر کدام جنبه‌ای متفاوت از اطلاعات را استخراج می‌کند: مثلاً یک سر ممکن است روی روابط معنایی بین کلمات تمرکز کند و دیگری روی ساختار نحوی جمله.

خروجی هر سر یک بردار ویژگی است. این بردارها پس از محاسبه، با هم **در امتداد بُعد ویژگی ترکیب (Concatenate)** می‌شوند و سپس از طریق یک لایه خطی عبور می‌کنند تا به ابعاد مطلوب مدل تبدیل شوند.

این ساختار به مدل اجازه می‌دهد تا وابستگی‌ها و روابط پیچیده بین اجزای ورودی را بهتر درک کند. توجه چندسر هم در بخش **رمزگذار (Encoder)**و هم در **رمزگشا (Decoder)**استفاده می‌شود و از آن‌جایی که بدون نیاز به RNNیا CNNعمل می‌کند، یکی از عوامل اصلی موفقیت و سرعت بالای ترنسفورمرها محسوب می‌شود.



آموزش

تنظیم‌کننده نرخ یادگیری

در پیاده‌سازی مدل ترنسفورمر در یادگیری عمیق، تنظیم‌کننده نرخ یادگیری (Learning Rate Scheduler) نقش بسیار مهمی در بهبود روند آموزش و رسیدن به همگرایی بهتر ایفا می‌کند. نرخ یادگیری یکی از حساس‌ترین هایپرپارامترهای مدل است که اگر به‌درستی انتخاب نشود، می‌تواند باعث عدم یادگیری مناسب یا نوسان زیاد در به‌روزرسانی وزن‌ها شود. در مقاله اصلی ترنسفورمر با عنوان"Attention is All You Need" ، یک تنظیم‌کننده نرخ یادگیری خاص پیشنهاد شده است که ترکیبی از افزایش اولیه نرخ یادگیری (warm-up) و کاهش تدریجی آن بر اساس معکوس ریشه گام آموزشی است. فرمول این تنظیم‌کننده به صورت زیر است:

$$Ir(step) = d_{model}^{-0.5}.\min(step^{-0.5}, step.warmup\_steps^{-1.5})$$

در این فرمول:

d_{model} بعد بردارهای ورودی/خروجی در مدل ترنسفورمر است.
$step$ شماره گام آموزش (training step) فعلی است.
$warmup_steps$تعداد گام‌هایی است که نرخ یادگیری به‌تدریج افزایش می‌یابد.

این استراتژی باعث می‌شود در ابتدای آموزش، نرخ یادگیری از مقدار بسیار کوچک شروع شده و به‌آرامی افزایش یابد (مرحله warm-up)سپس بعد از رسیدن به نقطه اوج، به‌صورت تدریجی و بر اساس معکوس ریشه شماره گام کاهش یابد. این کار به مدل کمک می‌کند تا در ابتدا وزن‌ها را به آرامی تنظیم کند و از ناپایداری‌های ناشی از نرخ یادگیری زیاد در مراحل اولیه جلوگیری شود، و در مراحل بعدی هم با کاهش نرخ یادگیری، مدل به سمت همگرایی بهتر حرکت کند.
در بسیاری از پیاده‌سازی‌های مدرن ترنسفورمر (مثلاً در PyTorch یا TensorFlow)از چنین تنظیم‌کننده‌هایی با کمی تغییرات استفاده می‌شود تا آموزش مدل‌ها پایدارتر و بهینه‌تر انجام شود.

تابع ضرر و معیارها (Loss Function and Metrics)

در یادگیری عمیق، تابع ضرر یا معیار (Loss Function) نقش بسیار مهمی دارد؛ چون مشخص می‌کند مدل چقدر خوب یاد گرفته و چقدر اشتباه کرده است.

وقتی یه مدل ترنسفورمر آموزش می‌دیم (مثلاً برای ترجمه یا تولید متن)، در هر مرحله، مدل سعی می‌کنه یه کلمه‌ی بعدی رو پیش‌بینی کنه. برای اینکه بفهمیم پیش‌بینی مدل چقدر درسته، از **تابع ضرر** استفاده می‌کنیم. رایج‌ترین تابع ضرری که استفاده می‌شه، **Cross Entropy Loss** هست.

این تابع مقدار شباهت بین خروجی مدل (که به صورت یک احتمال برای هر کلمه است) و پاسخ واقعی (کلمه‌ای که باید تولید می‌کرد) رو اندازه‌گیری می‌کنه. اگه مدل به کلمه درست احتمال بالایی بده، مقدار ضرر کم می‌شه. اگه به کلمه اشتباه احتمال بالا بده، ضرر زیاد می‌شه.

به زبان ساده‌تر:

مدل یه لیست از احتمال‌ها می‌سازه (مثلاً: "کتاب" ۰.۱، "خانه" ۰.۷، "مدرسه" ۰.۲)

جواب درست مثلاً "خانه" هست

Cross Entropy نگاه می‌کنه که مدل چقدر احتمال به "خانه" داده هر چی احتمال بیشتر باشه، ضرر کمتر می‌شه مدل سعی می‌کنه ضرر رو کم کنه تا یاد بگیره بهتر پیش‌بینی کنه

در طول آموزش، این ضرر برای همه‌ی کلمات در جمله محاسبه می‌شه و مدل با استفاده از پس‌انتشار (backpropagation) وزن‌های خودش رو طوری تنظیم می‌کنه که در دفعات بعدی بهتر عمل کنه.

تابع آموزش

در آموزش یک مدل ترنسفورمر، هدف اینه که مدل کم‌کم یاد بگیره چطور داده‌های ورودی (مثلاً یک جمله) رو پردازش کنه و خروجی درست (مثلاً ترجمه‌ی اون جمله) رو تولید کنه. برای این کار، از چیزی به اسم **حلقه آموزش** استفاده می‌کنیم.
حلقه آموزش مثل یک فرآیند تکراریه که در هر دور، مدل یک قدم به یادگیری بهتر نزدیک‌تر می‌شه. مراحل:
۱. داده‌های ورودی و خروجی رو به مدل می‌دیم. مثلاً جمله انگلیسی به‌عنوان ورودی و ترجمه‌ی فرانسوی به‌عنوان خروجی.
۲. مدل با استفاده از وزن‌های فعلی‌اش، سعی می‌کنه خروجی رو پیش‌بینی کنه (همون کلمات ترجمه).
۳. حالا خروجی مدل (پیش‌بینی‌ها) رو با پاسخ واقعی (ترجمه صحیح) مقایسه می‌کنیم. اینجا از تابع ضرر استفاده می‌کنیم (مثل cross-entropy loss) تا بفهمیم مدل چقدر اشتباه کرده.
۴. بعد از محاسبه ضرر، با روش پس‌انتشار خطا(backpropagation) ، مدل یاد می‌گیره که چطوری وزن‌هاش رو تغییر بده تا دفعه بعد بهتر عمل کنه.
۵. با استفاده از یک به‌روزرسانی‌کننده (optimizer) مثلAdam ، وزن‌های مدل تغییر می‌کنن.
۶. این مراحل برای هر دسته از داده‌ها (batch) تکرار می‌شن و کل این فرآیند برای چندین بار روی همه‌ی داده‌ها اجرا می‌شه (تعداد تکرارهاepoch = ها)
در هر دور از آموزش، مدل کم‌کم ضرر کمتری تولید می‌کنه و بهتر یاد می‌گیره که خروجی درست رو تولید کنه.

ارزیابی

- **تابع کمک ترجمه** (Translation Helper Function)

تابع کمک ترجمه (Translation Helper Function) یکی از اجزای کلیدی در پیاده‌سازی مدل‌های ترنسفورمر برای ترجمه ماشینی است. این تابع معمولاً در مرحله‌ی ارزیابی (Inference/Test) مورد استفاده قرار می‌گیرد و مسئول تولید ترجمه به‌صورت خودبازگشتی (Autoregressive) است.

در این فرایند، ابتدا جمله‌ی ورودی به بخش Encoder مدل داده می‌شود تا نمایش برداری آن استخراج گردد. این بردار سپس به‌عنوان ورودی ثابت به Decoderمنتقل می‌شود. در سمتDecoder ، تابع کمک ترجمه کار خود را با یک توکن شروع (مانند <start>) آغاز کرده و در هر مرحله، توکن بعدی را بر اساس خروجی مدل پیش‌بینی می‌کند.

پس از تولید هر توکن، آن را به توالی ترجمه‌شده اضافه کرده و توالی جدید را دوباره به Decoder می‌فرستد تا توکن بعدی را تولید کند. این روند تا رسیدن به توکن پایان (مثلاً <end>) یا رسیدن به **حداکثر طول مجاز جمله** ادامه می‌یابد.

تابع کمک ترجمه درواقع مانند یک حلقه‌ی تکرار (loop) عمل می‌کند که در هر مرحله، خروجی مرحله قبل را به‌عنوان ورودی مرحله‌ی بعدی استفاده می‌کند. این مکانیزم باعث می‌شود مدل بتواند ترجمه‌ای روان، معنادار و مرحله‌به‌مرحله تولید کند. همچنین چون در هر گام فقط از توکن‌های قبلی استفاده می‌شود، رفتار مدل بسیار شبیه به شرایط واقعی ترجمه در دنیای بیرون خواهد بود.

بنابراین، تابع کمک ترجمه یکی از اجزای حیاتی در ساختار مدل‌های ترنسفورمر است و بدون آن، امکان تولید خروجی متنی به شکل پیاپی و منطقی وجود نخواهد داشت.

خلاصه ای از مقاله Quantized Transformer از دانشگاه Stanford

این مقاله با عنوان "ترنسفورمر کمی‌سازی‌شده" به بررسی چالش‌های محاسباتی و حافظه‌ای مدل ترنسفورمر در پردازش زبان طبیعی می‌پردازد و راهکارهایی برای بهینه‌سازی آن ارائه می‌دهد. نویسنده، چائوفی فان از دانشگاه استنفورد، نشان می‌دهد که می‌توان مدل‌های ترنسفورمر را به ۸ بیت کمی‌سازی کرد بدون اینکه عملکرد آن به شدت کاهش یابد. کمی‌سازی به کاهش اندازه مدل کمک می‌کند و همچنین مصرف انرژی را به میزان قابل توجهی کاهش می‌دهد؛ به طوری که یک مدل ۸ بیتی می‌تواند ۱۸ تا ۳۰ برابر کمتر انرژی مصرف کند.

مقاله به کارایی ترنسفورمر کمی‌سازی‌شده در وظایفی نظیر ترجمه ماشینی، طبقه‌بندی جملات و پاسخ‌گویی به سوالات می‌پردازد. نتایج نشان می‌دهد که مدل‌های ۸ بیتی تقریباً به اندازه مدل‌های ۳۲ بیتی عملکرد خوبی دارند و در مواردی که داده‌های آموزشی محدود است، کمی‌سازی می‌تواند به عنوان یک منظم‌کننده عمل کند و حتی عملکرد بهتری را ارائه دهد. همچنین، نویسنده بیان می‌کند که کمی‌سازی وزن‌ها به تنهایی نسبت به کمی‌سازی هم وزن‌ها و هم فعالیت‌ها نتایج بهتری دارد.

در این مقاله، از تکنیک‌های کمی‌سازی مختلف مانند کمی‌سازی خطی و کمی‌سازی باینری استفاده شده و نتایج آزمایشات بر روی مجموعه‌های داده مختلف، از جمله ترجمه از زبان انگلیسی به آلمانی و مجموعه‌های داده BERT، بررسی شده است. در نهایت، مقاله به این نکته اشاره می‌کند که ترنسفورمرهای کمی‌سازی‌شده قابلیت اجرای مؤثر بر روی دستگاه‌های موبایل را دارند و می‌توانند ترجمه آفلاین را ممکن کنند. به طور کلی، نتایج نشان می‌دهند که ترنسفورمر کمی‌سازی‌شده می‌تواند در کاربردهای واقعی مؤثر باشد و با استفاده از تکنیک‌های دیگر در آینده بهبود یابد.

لینک مقاله :

https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1194/reports/custom/15742249.pdf

در صورتی که تمایل دارید با مفاهیم ترنسفورمرها به‌صورت عمیق‌تر و دقیق‌تر آشنا شوید، پیشنهاد می‌شود ویدیوی آموزشی "Overview of Transformers" ارائه‌شده توسط دانشگاه استنفورد را مشاهده نمایید.

لینک ویدیو :

https://youtu.be/JKbtWimlzAE?si=r2aDiCGWmFvGJGbh

THE END