دانشگاه صنعتی امیرکبیر

(پلی تکنیک تهران)

دانشکده مهندسی کامپیوتر

گزارش سمینار

استاد

دکتر مریم امیرمزلقانی

اعضا

مهدی نیک نژاد- ۹۶۳۰۰۳۹

سینا محمودی-۹۶۳۱۷۰۳

رسول خزاعی لکی-۹۶۳۱۷۰۱

بهمن ۱۴۰۰

فهرست مطالب:

فهرست اشکال و جداول:

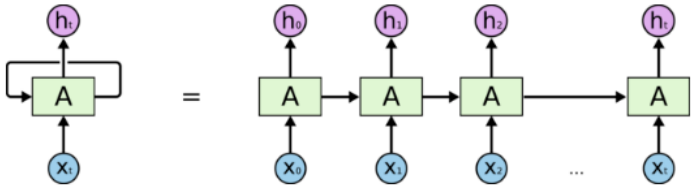
مقاله اول

مقدمه

در این مقاله یک معماری شبکه جدید، به نام ترنسفورمر را پیشنهاد می کنیم، که صرفاً بر اساس مکانیزم توجه است و به طور کامل از بازگشت و کانولوشن چشم پوشی می کند. آزمایش‌ها روی دو کار ترجمه ماشینی نشان می‌دهد که این مدل‌ها از نظر کیفیت برتر هستند، در حالی که موازی‌پذیری بیشتری دارند و به زمان قابل توجه کمتری برای آموزش نیاز دارند.

شبکه‌های عصبی بازگشتی

این شبکه ها یک لایه feedback (بازخورد) دارند که در آن خروجی شبکه و ورودی بعدی به شبکه برگردانده می‌شود. RNN[1] ها به دلیل داشتن همین لایه بازخورد، عملا دارای حافظه داخلی هستند؛ یعنی ورودی های قبلی را به خاطر می‌سپارند و از این حافظه برای پردازش دنباله‌ای از ورودی ها استفاده میکنند. پس RNN ها یک حلقه بازگشتی دارند که باعث می‌شود اطلاعات لحظات قبل از بین نرود و در شبکه باقی بمانند.



شکل ۱ - معماری RNN

کاربرد شبکه‌های بازگشتی در چند مورد است؛ مثلا برای پیش‌بینی سری‌ها (دنباله‌ها) با توجه به کلمات قبلی (اگر متن یک سری از کلمات باشد) یا برای پیش‌بینی قیمت در بورس با توجه به قیمتهای قبلی.

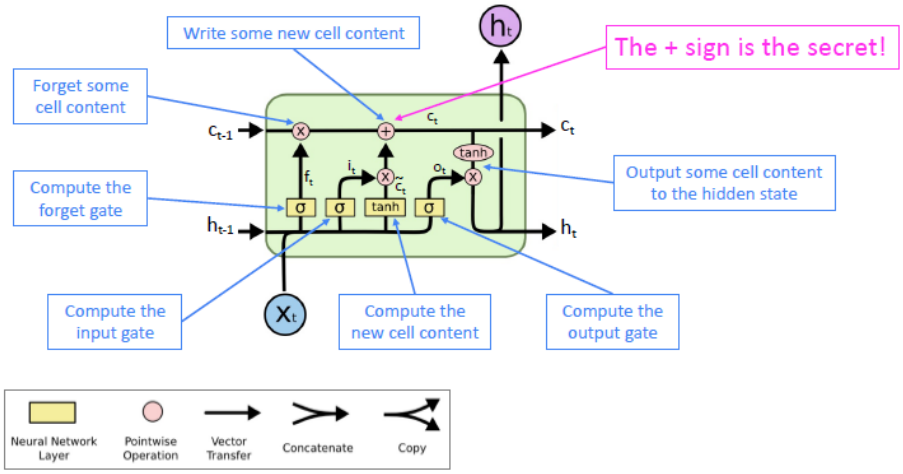
مشکلی که RNN ها داشتند و باعث شد تا رویکرد ترنسفورمرها جایگزین شوند، این بود که اگر سری طولانی شود، RNN نمی‌تواند به‌درستی عمل کند و ۲ مشکل پیش می‌آید: ۱) ناپایداری گرادیانها ۲) فراموش کردن ورودی‌های ابتدایی شبکه

۱. این مشکل به ۲ شکل خود را نشان می‌دهد: محو شدن / انفجار گرادیانها[2]. RNN ها مانند زنجیره هستند و خروجی‌های قبلی را به عنوان ورودی می‌گیرند. اگر مقدار گرادیان بیشتر از یک باشد، گرادیان به مرور بزرگ و

[1] Recurrent Neural Network
[2] Vanishing / Exploding gradients

بزرگتر می‌شود تا به اصطلاح منفجر شود و همچنین اگر مقدار گرادیان کوچکتر از یک باشد، مقدار گرادیان کوچک و کوچکتر می‌شود تا به اصطلاح محو و ناپدید شود.

۲. مشکل دیگر فراموشی این شبکه است. RNN ، یک جمله طولانی را پیمایش میکند و وقتی به آخر جمله می‌رسد، کلمات ابتدایی را فراموش میکند. برای حل این مشکل «حافظه‌ها» مطرح شدند و LSTM[۳] ها و مدل ساده‌تر آنها یعنی GRU پا به میدان گذاشتند. در معماری LSTM، یک سلول داریم که اطلاعات طولانی‌مدت[۴] را نگهداری میکند.



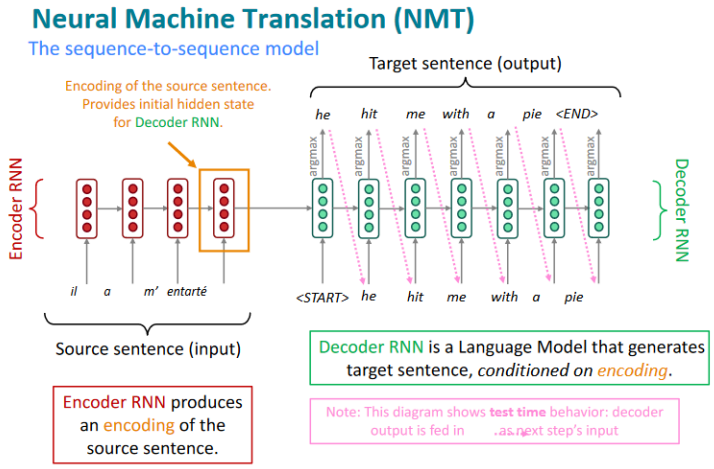
شکل ۲ - معماری LSTM

در تصویر شکل۲، c(t) نشان‌دهنده حافظه طولانی‌مدت است. (cell-state) و h(t) نشان‌دهنده حافظه کوتاه‌مدت است. (hidden state)

در پیش‌بینی کلمه بعدی با توجه به کلمات قبلی، مدل‌زبانی تعریف می‌شود؛ یعنی دنباله‌ای از کلمات را داریم و می‌خواهیم احتمال کلمه بعدی را بدست آوریم. کارهایی که در این حوزه تعریف می‌شوند گوناگون هستند؛ اما برای مثال تسک ترجمه ماشینی[۵] یکی از آنها است که معماری دنباله به دنباله[۶] دارد.

[۳] Long short-term memory
[۴] Long-term information
[۵] Neural Machine Translation Task
[۶] Sequence to sequence (seq۲seq)

معماری seq۲seq ، از ۲ RNN تشکیل شده است و در تسک ترجمه ماشینی، یک جمله از زبان مبدا را به جمله دیگر ترجمه میکنیم. این معماری از ۲ بخش کدگذار[۷] و کدگشا[۸] تشکیل شده است و هر گدام می‌توانند هر نوعی از RNN ها باشند؛ مثلا : Uni-dirctional, Bi-directional, multi-layer, LSTM, ...



شکل ۳ - معماری seq۲seq

همانطور که از شکل۳ مشخص است، کدگذار یک جمله از زبان مبدا را به یک بردار تبدیل میکند و کدگشا هم عملکرد عکس دارد.

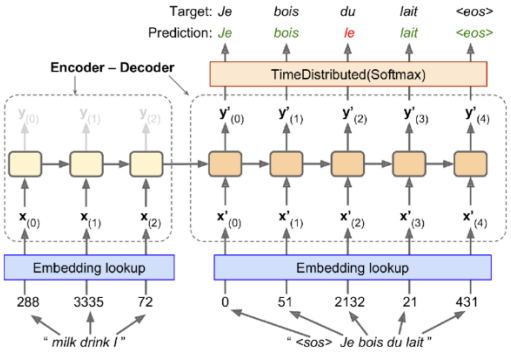
مشکلی که در معماری seq۲seq داریم این است که اطلاعات کلمات ابتدایی جمله به خوبی به کدگشا منتقل نمی‌شود و موقع آموزش مدل، هم گرادیان به خوبی به عقب منتقل نمی‌شود. البته یکسری راه‌حل داده شد که مثلا جمله مبدا را برعکس به کدگذار بدهیم اما دیدیم که با این حرکت، مشکل حل نمی‌شود؛ یعنی دوباره اطلاعات جمله مبدا در یک بردار با طول ثابت ذخیره میشود؛ اینجا بود که مکانیزم توجه[۹] ابداع شد.

ایده اولیه مکانیزم توجه

یکی از مهم‌ترین مفاهیمی است که انقلابی در حوزه پردازش زبان طبیعی به راه انداخته است. آقای باهدانا و همکارانشان در مقاله‌ای فن مقابل را معرفی کردند که در آن کدگشا در هر گام فقط روی کلمات مناسب توجه میکند. این ایده باعث شد که مسیر بین کلمه در زبان مبدا تا ترجمه‌اش خیلی کوتاه‌تر بشود و مشکل حافظه کوتاه‌مدت RNN ها خیلی کمتر مشکل ساز شود. در هر گام در این فن، کدگشا یک جمع وزن‌دار بین خروجی های کدگذار محاسبه میکندو همین وزنها مفهوم توجه را بیان میکنند. یعنی به آنکه وزن بیشتری دارد، توجه

[۷] Encoder
[۸] Decoder
[۹] Attention mechanism

بیشتری می‌شود. در حقیقت این وزنها تابعی از میزان مشابهت حالت نهان[۱۰] فعلی کدگشا با حالت نهان کدگشا است.



شکل ۴ – ایده اولیه مکانیزم توجه

مکانیزم توجه تعمیم‌یافته

فرضا یکسری موجودیت داریم و هر موجودیت هم با ۳ بردار بازنمایی می‌شود: ۱) بردار پرسش Q[۱۱] ۲) بردار کلید K[۱۲] ۳) بردار مقدار V[۱۳]

حالا می‌خواهیم توجه یک موجودیت روی بقیه موجودیت ها را محاسبه کنیم؛ یعنی مثلا برای محاسبه توجه شکل دایره روی بقیه شکلها (مثلث، مربع و پنج‌ضلعی) باید ترکیب خطی از سایر موجودیتها به جز دایره بسازیم.

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

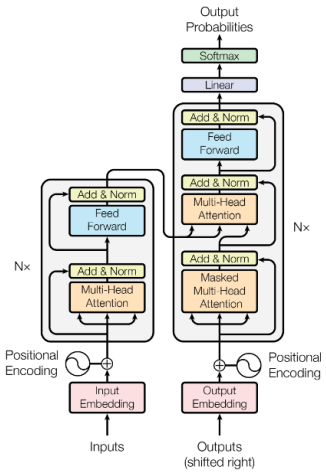
شکل ۵ – فرمول محاسبه توجه تعمیم یافته

مدل ترنسفورمرها

تعریف تعمیم یافته از مکانیزم توجه باعث پیشرفتهای بعدی در پردازش زبان شد.
مشکلی که RNNها داشتند این بود که نمیتوانستیم با آنها پردازش موازی داشته باشیم و میبایست پردازش ترتیبی[۱۴] انجام میشد.
مفهوم ترنسفورمرها اولین بار در مقاله Attention is al you need از طرف Google بیان شد.

[۱۰] Hidden state
[۱۱] Query vector
[۱۲] Key vector
[۱۳] Value vector
[۱۴] Sequential

ترنسفورمرها، ۲ بخش دارند :۱) کدگذار ۲) کدگشا . پس مدل ترنسفورمرها یک مدل seq۲seq است و ساختار کدگذار-کدگشا دارد.
در ترنسفورمرها از مکانیزم بازگشتی استفاده نمیشود و فقط از مکانیزم توجه که پیشتر بیان شد، استفاده میشود.



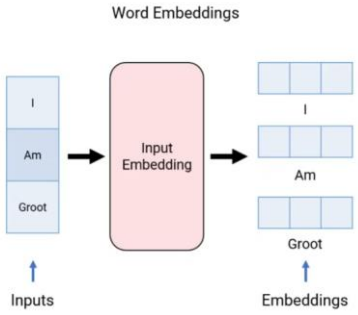
شکل ۶ - معماری ترنسفورمرها

اجزای داخلی کدگذار ترنسفورمرها

Input embedding

ترنسفورمرها به طور کلی برای تسک NMT معرفی شده‌اند؛ لذا ورودی ها دنباله‌ای از کلمات هستند. پس کدگذار یک دنباله متنی را میگیرد و یک بازنمایی از آنرا تولید میکند. (کدگشا هم این بازنمایی حاصل را میگیرد و سعی میکند کلمه بعدی را تخمین بزند)

باید کلمات را به صورت بردار نشان دهیم به‌طوریکه کلمات یکسان، بردارهای یکسان داشته باشند. به این عملیات word-embedding میگوییم.

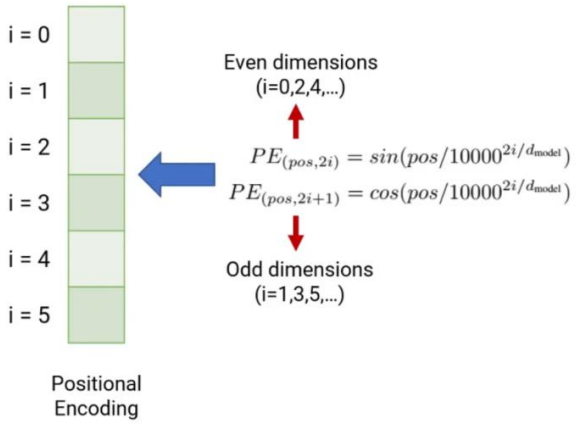


شکل ۷ - word-embedding

Positional encoding

در RNN ها ، توکن‌ها و کلمات به ترتیب می‌آیند و اطلاعات مکانی[۱۵] آنها مشخص است که کدام توکن قبل و کدام توکن بعد است. اما در ترنسفورمرها همه توکن‌ها با همدیگر وارد میشوند و اطلاعات مکانی نداریم؛ پس مکانیزم تعبیه مکانی جهت تزریق اطلاعات به ترنسفورمرها طراحی شده است.

انواع تعبیه مکانی : ۱) sin برای بعدهای زوج ۲) cos برای بعدهای فرد



شکل ۸ - مکانیزم تعبیه مکانی

۲ مورد input embedding و positional encoding ، نوعی پیش‌پردازش برای داده‌ها میباشند.

Multi-head attention

ابتدا self-attention را بیان میکنیم.

از Q,K برای محاسبه score استفاده میکنیم . این امتیاز را بر بعد بردار K تقسیم میکنیم و از حاصل softmax میگیریم و حاصلضرب آنرا با V محاسبه میکنیم و در نهایت، نتیجه تصمیم میگیرد که چه مقدار بقیه به این موجودیت توجه دارند. (در مقاله بعد بردار K یعنی d_k برابر است با ۶۴)

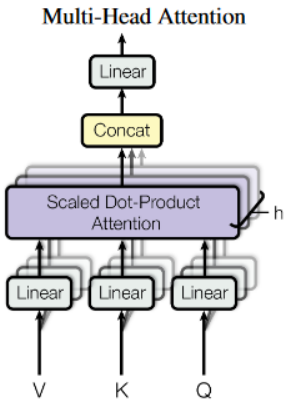
حالا mult-head attention را بیان میکنیم.

به این عملیات که بردارهای بازنمایی با اندازه بزرگ را به چند بردار کوچک بشکانیم و بعد روی تک تک آنها مکانیزم توجه را اعمال کنیم و در نهایت حاصل توجه این بردارهای کوچک را با یکدیگر الحاق کنیم توجه چندسر میگویند.

[۱۵] Spatial information

فایده‌ی آن برابر است با : ۱) توانایی موازی سازی و سرعت بیشتر ۲) تفسیرپذیری[۱۶] مدل

تفسیرپذیری یعنی اینکه هر head به یک چیز توجه میکند مثلا یکی به جنبه معنایی[۱۷] متن و دیگری به جنبه نحوی[۱۸] متن اشاره میکند.



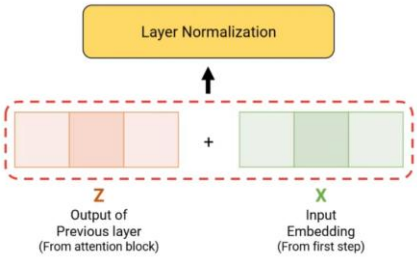
شکل ۹ - multi-head attention

Add & Norm

Layer Normalization برابر است با مجموع خروجی های بلوک attention از مرحله قبل + بردار input embedding از مرحله اول.

فواید نرمالیزیشن : ۱) کاهش زمان آموزش مدل ۲) کاهش bias مدل ۳) جیوگیری از انفجار وزنها

انواع نرمالیزیشنها : ۱) batch : محاسبه میانگین و واریانس فیچرها ۲) layer : محاسبه میانگین و واریانس داده‌ها



شکل ۱۰ - Add & Norm

[۱۶] Interpretability
[۱۷] Semantics
[۱۸] Syntax

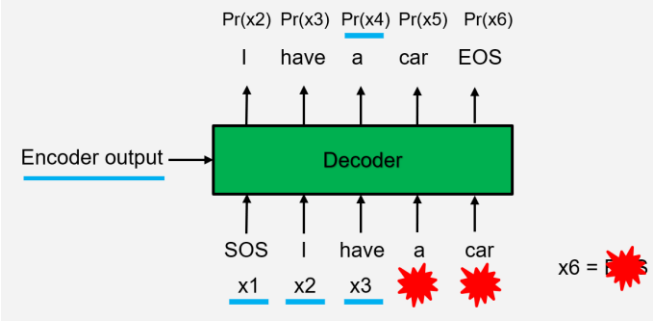
Residual connection

این اتصالها خروجی هر زیرلایه را به لایه Add & Norm لایه بعد می‌فرستند.

اجزای داخلی کدگشا ترنسفورمرها

Masked multi-head attention

در کل جمله ورودی را به کدگذار و جمله خروجی را به کدگشا میدهیم؛ پس در این حالت کدگشا میتواند تقلب کند و این طوری عملا آموزشی صورت نمیگیرد. پس برای جلوگیری از تقلب کدگشا از پنهان کردن[۱۹] ورودی استفاده میکنیم.



شکل ۱۱ - Masked multi-head attention

Multi-head attention

بردارهای K,V از کدگذار و بردار Q از کدگشا می‌آیند.

نتایج مقاله

۲ نوع مدل از ترنسفورمرها در مقاله بیان شده است: ۱) base ۲)big .

در نمودار زیر، مقایسه بین این ۲ نوع مدل از ترنسفورمرها با سایر مدلها را میبینیم. همانطور که مشخص است معیار BLEU score ترنسفورمرها بیشتر از بقیه است. این معیار برای ارزیابی ترجمه می باشد.

[۱۹] mask

Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [18]	23.75			
Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	**$3.3 \cdot 10^{18}$**	
Transformer (big)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

شکل ۱۲ – نتایج مقاله Attention is all you need

خوبی ها و بدی های مدل ترنسفورمر

خوبی‌ها:

- موازی سازی و آموزش سریعتر مدل
- نداشتن vanishing
- وجود مکانیزم attention

بدی‌ها

- نیاز به منابع سخت افزاری زیاد برای کار با ترنسفورمرها

نتیجه‌گیری

در این مقاله معماری ترنسفورمرها معرفی شد که مبتنی بر مکانیزم توجه خالص بود و ساختار کدگذار-کدگشا داشت.

مقاله دوم

مقدمه

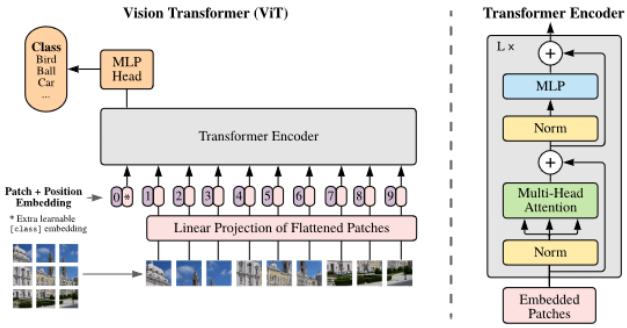
در حالی که معماری ترنسفورمرها به استانداردی برای کارهای پردازش زبان طبیعی تبدیل شده است، کاربردهای آن در بینایی کامپیوتر محدود باقی مانده است .در بینایی، توجه یا در ارتباط با شبکه‌های کانولوشن به کار می‌رود، یا برای جایگزینی اجزای خاصی از شبکه‌های کانولوشنال در حالی که ساختار کلی آنها در جای خود حفظ می‌شود، استفاده می‌شود .ما نشان می‌دهیم که این اتکا به CNN ضروری نیست و یک ترنسفورمر خالص که مستقیماً به دنباله‌ای از پچ‌های تصویر اعمال می‌شود، می‌تواند در کار طبقه‌بندی تصویر بسیار خوب عمل کند.

در این مقاله، مدل ViT معرفی میشود که قصد دارد تا ترنسفورمرها را جایگیزین CNN ها در کارهای پردازش تصویر کند. اما استفاده از ترنسفورمرها در پردازش تصویر سخت است و پیچیدگی زمانی $O(n^2)$ دارد.

راه حل‌هایی که برای این مسأله داریم این است که self-attention را به صورت سراسری اعمال کنیم.

مدل ViT

ایده اصلی : استفاده از مکانیزم توجه در پردازش تصویر؛ یعنی همه قسمتهای تصویر با همدیگر ارتباط داشته باشند.



شکل ۱۳ - مدل ViT

مراحل

۱) Patch embedding: تصویر را به چندین ناحیه با اندازه ثابت و بدون اشتراک تقسیم میکنیم

۲) تبدیل Patch ها به بردار توسط عملیات Linear projection + اضافه کردن توکن cls به ورودی های کدگذار

۳) اضافه کردن Positional Encoding به بردارها

۴) وارد کردن توکن ها به کدگذار ترنسفورمرها

۵) هر کدام از توکنهای حاصل از خروجی کدگذار حاصل توجه یک patch به بقیه patch ها است

۶) انجام عملیات دسته‌بندی

انواع مدلهای موجود در مقاله

۱) ViT-base ۲) ViT-Large ۳) ViT-Huge

نتایج مقاله

در شکل۱۴، TPUv۳-core-days نشان‌دهنده میزان هزینه‌کرد برای آموزش است. و نتیجه به این صورت است که ViT در مقابل بقیه مدلها بسیار کم‌هزینه‌تر است.

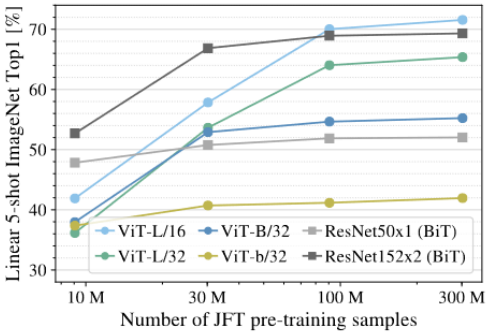
	Ours-JFT (ViT-H/14)	Ours-JFT (ViT-L/16)	Ours-I21k (ViT-L/16)	BiT-L (ResNet152x4)	Noisy Student (EfficientNet-L2)
ImageNet	**88.55** ± 0.04	87.76 ± 0.03	85.30 ± 0.02	87.54 ± 0.02	88.4/88.5*
ImageNet ReaL	**90.72** ± 0.05	90.54 ± 0.03	88.62 ± 0.05	90.54	90.55
CIFAR-10	**99.50** ± 0.06	99.42 ± 0.03	99.15 ± 0.03	99.37 ± 0.06	–
CIFAR-100	**94.55** ± 0.04	93.90 ± 0.05	93.25 ± 0.05	93.51 ± 0.08	–
Oxford-IIIT Pets	**97.56** ± 0.03	97.32 ± 0.11	94.67 ± 0.15	96.62 ± 0.23	–
Oxford Flowers-102	99.68 ± 0.02	**99.74** ± 0.00	99.61 ± 0.02	99.63 ± 0.03	–
VTAB (19 tasks)	**77.63** ± 0.23	76.28 ± 0.46	72.72 ± 0.21	76.29 ± 1.70	–
TPUv3-core-days	2.5k	0.68k	0.23k	9.9k	12.3k

Table 2: Comparison with state of the art on popular image classification benchmarks. We report mean and standard deviation of the accuracies, averaged over three fine-tuning runs. Vision Transformer models pre-trained on the JFT-300M dataset outperform ResNet-based baselines on all datasets, while taking substantially less computational resources to pre-train. ViT pre-trained on the smaller public ImageNet-21k dataset performs well too. *Slightly improved 88.5% result reported in Touvron et al. (2020).

شکل ۱۴ – نتایج مقاله ViT

در کل نتایج نشان می‌دهد که ترنسفورمرها از مدلهای بر پایه CNN بهتر هستند، اما آیا این بدان معنا است که باید برای کارهای پردازش تصویر خود از ترنسفورمرها استفاده کنیم؟

اگر تعداد داده‌هایی که داریم کم باشد، CNN ها بهتر میباشند اما اگر تعداد داده هایی که در اختیار داریم زیاد است، بهتر است از همان ترنسفورمرها استفاده کنیم. این موضوع را از روی نمودار زیر از مقاله بیان میکنیم.



شکل ۱۵ - مقایسه بین مدلهای ViT و CNN

نتیجه‌گیری

در این مقاله ابتدا ویژگی گرسنه محاسباتی بودن مکانیزم توجه مورد بررسی قرار گرفت و در ادامه راه‌حلی که نویسندگان مقاله ارائه داده بودند بیان شد و آن ایده تقسیم تصویر به چندین patch بود.

در انتها دیدیم که ViT بهتر از CNN-based ها عمل میکند اما به شرطی که تعداد داده نمونه زیادی داشته باشیم.

مقاله سوم

مقدمه

اخیرا پیشرفت هایی که در پردازش زبان طبیعی صورت گرفته ناشی از این موضوع است که هم ساختار مدل و هم آموزش آن تغییر و بهبود یافته اند. ساختار های ترنسفورمری این امکان را فراهم ساخته اند که مدل های با ظرفیت بیشتر بسازیم و آموزش نیز این امکان را فراهم آورده که بتوانیم از این ظرفیت بیشتر در فرآیند های با گستره ی بیشتری استفاده کنیم.

ترنسفورمر ها و ساختار های ترنسفورمری یک کتابخانه متن باز است که هدف آن این است که این پیشرفت ها را به جامعه بزرگتری از یادگیری ماشین برساند. این کتابخانه از ساختار های ترنسفورمری که به دقت مهندسی شده اند و از یک API مشخص استفاده می کنند. پشتیبان این کتابخانه از مجموعه ای از مدل های پیش آموزش داده شده که توسط جامعه ساخته شده، تشکیل شده است و برای آن در دسترس است. ترنسفورمر ها به صورتی طراحی شده اند که به راحتی توسط پژوهش گران قابل توسعه باشند و برای استفاده کنندگان آسان باشند و در استفاده های صنعتی از آنها بتوان به سرعت از آنها استفاده کرد.

ساختار ترنسفورمری به سرعت به ساختار و معماری غالب برای پردازش زبان طبیعی تبدیل شده است و از مدل های عصبی جایگزین مانند شبکه های عصبی کانولوشنی و مکرر هم در بحث درک زبان طبیعی و هم در تولید آن از نظر عملکرد پیشی گرفته است. این معماری بر اساس داده های آموزشی و سایز مدل تغییر می کند و آموزش موازی کارآمد را راحت تر می کند و ویژگی های توالی دور برد را نیز به مدل ما اضافه می کند. پیش آموزش مدل به مدل ها این قابلیت را می دهد که روی بدنه های عمومی آموزش داده شوند و پس از آن بتوانند با وظایف خاص خود با عملکردی بسیار خوب سازگار شوند.

معماری ترنسفورمر مخصوصاً برای پیش آموزش روی مجموعه های متنی بزرگ مفید است که باعث می شود که دقت مدل ما در حوزه هایی مانند دسته بندی متنی، درک زبان، ترجمه ماشینی، وضوح مرجع و خلاصه سازی بسیار افزایش یابد. این پیشرفت ها باعث می شود که چالش های عملی ای برای ما به وجود بیاید که این مدل ها به خوبی و به صورت وسیع بتوانند استفاده شوند. استفاده همه جانبه از ترنسفورمر سیستم هایی را برای آموزش، تجزیه و تحلیل، مقیاس بندی و تقویت مدل بر روی انواع پلتفرم ها می طلبد. این معماری به عنوان یک بلوک سازنده برای طراحی توسعه ها و آزمایش های دقیق که روز به روز پیچیده تر می شوند به کار می روند. استفاده فراگیر از روش های پیش آموزشی، نیاز هایی را به وجود آورده است که از جمله آنها می

توان به توزیع کردن، تنظیم دقیق، پیاده سازی و فشرده سازی هسته های مدل پیش آموزشی مورد استفاده جامعه هدف اشاره کرد.

ترنسفورمرها یک کتابخانه مستقل هستند که برای پشتیبانی از معماری ها و ساختار های مبتنی بر ترنسفورمر هستند که استفاده از مدل های از پیش آموزش دیده را راحت تر می کنند. در هسته این کتابخانه یک پیاده سازی از ترنسفورمر وجود دارد که هم برای تحقیق و هم برای تولید به کار می رود. فلسفه آن این است که بتواند از قدرت صنعتی پیاده سازی مدل های محبوب که به راحتی قابل خواندن، توسعه و پیاده سازی هستند، استفاده کند. بر این اساس این کتابخانه از توزیع و استفاده از گستره وسیعی از مدل های از پیش آموزش داده شده در یک مدل مرکزی پشتیبانی می کند. این مدل این ویژگی را دارد که کاربران مدل های متنوعی را به وسیله API مینیمال با یکدیگر مقایسه کنند و با استفاده از مدل های به اشتراک گذاشته شده روی گستره وسیعی از کارها آزمایش های خود را انجام دهند. ترنسفورمرها یک تلاش مداوم اند که توسط تیم مهندسان و محققان HuggingFace با حمایت یک جامعه پر جنب و جوش متشکل از ۴۰۰ مشارکت کننده خارجی بو وجود می آیند. این کتابخانه تحت مجوز Apache ۲،۰ منتشر شده است و در GitHub در دسترس است.

کارهای مرتبط انجام شده

NLP و ML یک فرهنگ قوی در مورد تولید ابزار تحقیقاتی متن باز دارند. ساختار ترنسفورمرها از کتابخانه tensor۲tensor و کدهای BERT که هر دو از تحقیقات گوگل هستند، الهام گرفته است. بحث برآورده کردن کش کردن راحت برای مدل های از پیش آموزش داده شده از AllenNLP گرفته شده است. این کتابخانه همچنین ارتباط نزدیکی با ترجمه عصبی و سیستم های مدل زبانی مانند Fairseq، OpenNMT، Texar و Megatron-LM دارد و ساخته شدن بر اساس این المان ها این امکان را به ترنسفورمرها می دهد که ویژگی های در معرض دید کاربر داشته باشد که این اجازه را می دهد که مدل ها را به راحتی دانلود و کش کرده و به صورت دقیق تنظیم کند و همچنین به صورت یکپارچه به تولید این مدل ها بپردازد. ترنسفورمرها قابلیت سازگاری با برخی از این کتابخانه ها را در خود دارند که اکثرا شامل یک ابزار برای اجرای یک نتیجه با استفاده از مدل های Marian NMT و Google BERT است.

برای حوزه general-purpose NLP دو کتابخانه اصلی وجود دارد: NLTK و Stanford CoreNLP. این کتابخانه ها رویکردهای مختلف NLP را در یک پکیج ارائه می دهند. اخیرا کتابخانه های عمومی و متن باز به صورت عمده روی یادگیری ماشین برای کارهای NLP تمرکز کرده اند که از این کتابخانه ها می توان به Spacy، AllenNLP، Flair و Stanza اشاره کرد. ترنسفورمرها عملکردی مشابه این کتابخانه

ها دارند. هر کدام از این کتابخانه ها امروزه از ترنسفورمرها و مدل مرکزی آن به عنوان یک فریم ورک سطح پایین استفاده می کنند.

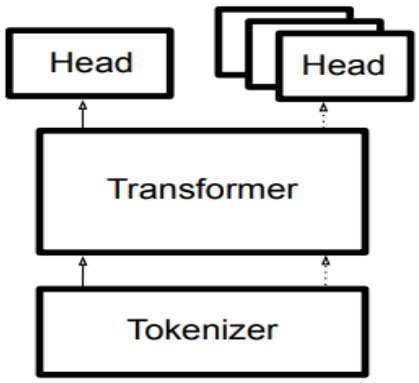
با توجه به اینکه ترنسفورمرها از یک هاب برای مدل های NLP استفاده می کنند با مدل های هاب پر طرفداری همچون Torch Hub و TensorFlow که پارامتر های فریم ورکی را جمع آوری می کنند ارتباط دارد. برخلاف این هاب ها، ترنسفورمرها مختص دامنه هستند که به سیستم اجازه می‌دهد تا پشتیبانی خودکار برای تحلیل مدل، استفاده، استقرار و محک زدن را ارائه دهد.

طراحی کتابخانه

ترنسفورمرها طراحی شده اند تا pipeline استاندارد در مدل های NLP را به نمایش بگذارند که شامل ۳ مرحله زیر است:

۱) پردازش داده
۲) اعمال یک مدل
۳) پیش بینی

اگرچه کتابخانه شامل ابزارهایی است که آموزش و توسعه را تسهیل می کند، در این گزارش فنی ما بر روی مشخصات اصلی مدل سازی تمرکز می کنیم. هر مدل در کتابخانه به طور کامل توسط سه بلوک ساختمانی که در نمودار زیر نشان داده شده است تعریف شده می شوند: (الف) توکنایزر، که متن خام را به کدهای شاخص پراکنده تبدیل می کند، (ب) یک ترنسفورمر، که شاخص های پراکنده را به جاسازی های متنی تبدیل می کند، (ج) یک سر، که از بستر های متنی برای پیش بینی یک کار خاص استفاده می کند. اکثر نیازهای کاربران را می توان با این سه مؤلفه برطرف کرد که در شکل ۱ ساختار آن مشاهده می شود.



شکل ۱۶ – ساختار کلی کتابخانه ترنسفورمر

ترنسفورمرها

در مرکز کتابخانه، پیاده سازی های به دقت آزمایش شده انواع معماری ترنسفورمر هستند که به طور گسترده در NLP استفاده می شوند. در حالی که هر یک از این معماری ها از هد های یکسانی استفاده می کنند، تفاوت های قابل توجهی بین آنها از جمله نمایش های موقعیتی، پوشاندن، لایه بندی و استفاده از طراحی دنباله به دنباله وجود دارد. علاوه بر این، مدل های مختلفی برای هدف قرار دادن کاربردهای مختلف NLP مانند درک، تولید و تولید مشروط، به علاوه موارد استفاده تخصصی مانند استنتاج سریع یا برنامه‌های چند زبانه ساخته شده‌اند.

در عمل، همه مدل ها از سلسله مراتب انتزاع یکسانی پیروی می‌کنند: یک کلاس پایه، نمودار محاسباتی مدل را از یک کد گذاری (نمایش روی ماتریس جاسازی) از طریق یک سری لایه های self attention تا حالت های پنهان رمز گذار نهایی پیاده سازی می‌کند. کلاس پایه برای هر مدل خاص است و از پیاده سازی اصلی مدل پیروی می‌کند که به کاربران این انعطاف پذیری را می‌دهد تا به راحتی کارهای درونی هر معماری جداگانه را تشریح کنند. در بیشتر موارد، هر مدل در یک فایل پیاده سازی می شود تا توسعه پذیری آسانی داشته باشد.

تا جایی که ممکن است، معماری های مختلف از یک API یکسان پیروی می کنند و به کاربران اجازه می دهند به راحتی بین مدل های مختلف جا به جا شوند. مجموعه ای از کلاس‌های خودکار، یک API یکپارچه را فراهم می‌کند که جابجایی بسیار سریع بین مدل ها و حتی بین چارچوب ها را امکان پذیر می‌کند. این کلاس ها به طور خودکار با پیکر بندی مشخص شده توسط مدل از پیش آموزش دیده مشخص شده توسط کاربر نمونه سازی می شوند. در شکل ۲ می توانیم لیست ترنسفورمرهای پیاده سازی شده را مشاهده کنیم:

Transformers	
Masked $[x_{1:N\setminus n} \Rightarrow x_n]$	
BERT	(Devlin et al., 2018)
RoBERTa	(Liu et al., 2019a)
Autoregressive $[x_{1:n-1} \Rightarrow x_n]$	
GPT / GPT-2	(Radford et al., 2019)
Trans-XL	(Dai et al., 2019)
XLNet	(Yang et al., 2019)
Seq-to-Seq $[\sim x_{1:N} \Rightarrow x_{1:N}]$	
BART	(Lewis et al., 2019)
T5	(Raffel et al., 2019)
MarianMT	(J.-Dowmunt et al., 2018)
Specialty: Multimodal	
MMBT	(Kiela et al., 2019)
Specialty: Long-Distance	
Reformer	(Kitaev et al., 2020)
Longformer	(Beltagy et al., 2020)
Specialty: Efficient	
ALBERT	(Lan et al., 2019)
Electra	(Clark et al., 2020)
DistilBERT	(Sanh et al., 2019)
Specialty: Multilingual	
XLM/RoBERTa	(Lample and Conneau, 2019b)

شکل ۱۷ – لیست ترنسفورمرهای مورد استفاده در کتابخانه

توکنایزرها

یکی از جنبه های حیاتی کتابخانه NLP، پیاده سازی توکنایزرهای ضروری برای استفاده از هر مدل است. کلاس‌های توکنایزر (که هر کدام از یک کلاس پایه مشترک به ارث می برند) می‌توانند از یک مدل از پیش آموزش دیده مربوط، نمونه سازی شوند یا به صورت دستی پیکر بندی شوند. این کلاس ها مجموعه توکن به واژگان خود را برای مدل مرتبط خود متناظر کرده و ذخیره می‌کنند و کد گذاری و کد گشایی دنباله های ورودی را بر اساس فرآیند توکن سازی خاص مدل انجام می‌دهند. کاربران می‌توانند به راحتی توکنایزر را با رابط ها تغییر دهند تا نقشه های توکن اضافی، توکن های خاص (مانند نشانه های طبقه بندی یا جداسازی) را اضافه کنند یا اندازه واژگان را تغییر دهند.

توکنایزر ها همچنین می توانند ویژگی های مفید اضافی را برای کاربران پیاده سازی کنند. گستره این ویژگی ها از نوع شاخص ها در دسته بندی توالی تا حداکثر طول کوتاه کردن دنباله با در نظر گرفتن توکن های ویژه مدل اضافه شده (اکثر مدل های ترنسفورمر از قبل آموزش دیده دارای حداکثر طول توالی هستند) را شامل می شوند.

برای آموزش مجموعه داده های بسیار بزرگ، توکن سازی مبتنی بر پایتون اغلب به طور نا مطلوبی کند است. در جدیدترین نسخه، ترنسفورمرها پیاده سازی خود را به طور پیش فرض به استفاده از یک کتابخانه توکنیزاسیون بسیار بهینه سازی شده تغییر داده اند. در شکل ۳ می توان لیست توکنایزر های مورد استفاده در این کتابخانه ها را مشاهده کرد:

Tokenizers	
Name	Ex. Uses
Character-Level BPE	NMT, GPT
Byte-Level BPE	GPT-2
WordPiece	BERT
SentencePiece	XLNet
Unigram	LM
Character	Reformer
Custom	Bio-Chem

شکل ۱۸ – لیست توکنایزر های مورد استفاده در کتابخانه

هد ها

هر ترنسفورمر را می توان با یکی از چندین هد آماده و پیاده سازی شده با خروجی های مناسب برای انواع معمول کار جفت کرد. این هد ها به عنوان کلاس‌های بسته بندی اضافی در بالای کلاس پایه و برای اضافه کردن یک لایه خروجی خاص و تابع از دست دادن اختیاری در بالای جاسازهای متنی ترنسفورمر پیاده سازی می شوند.

این کلاس ها از یک الگوی نام گذاری مشابه پیروی می‌کنند:
XXXForSequenceClassification که در آن XXX نام مدل است و می‌تواند برای تطبیق (تنظیم دقیق) یا پیش آموزش استفاده شود. برخی از هد ها، مانند تولید شرطی، از عملکردهای اضافی مانند نمونه برداری و جستجوی پرتوی پشتیبانی می کنند.

برای مدل های از قبل آموزش دیده، هد هایی را که برای پیش آموزش خود مدل استفاده می شوند، رها می‌کنیم. به عنوان مثال، برای BERT ما مدل سازی زبان و هد های پیش بینی جمله بعدی را منتشر می‌کنیم که امکان تطبیق آسان با استفاده از اهداف پیش آموزشی را فراهم می‌کند. همچنین استفاده از پارامترهای اصلی ترنسفورمر را با انواع هد های دیگر برای تنظیم دقیق برای کاربران آسان می کنیم. در حالی که هر هد را می توان به طور کلی استفاده کرد، کتابخانه همچنین شامل مجموعه ای از نمونه هایی است که عملکرد هر هد را در رویارویی با مشکلات واقعی نشان می دهد. این مثال ها نشان می‌دهند که چگونه یک مدل از پیش آموزش دیده شده را می‌توان با یک هد مشخص برای دستیابی به نتایج پیشرفته در طیف وسیعی از کارهای NLP تطبیق داد. در شکل ۴ لیست هد های لیست هد های پیاده سازی شده برای این کتابخانه را مشاهده می کنیم.

Heads				
Name	Input	Output	Tasks	Ex. Datasets
Language Modeling	$x_{1:n-1}$	$x_n \in \mathcal{V}$	Generation	WikiText-103
Sequence Classification	$x_{1:N}$	$y \in \mathcal{C}$	Classification, Sentiment Analysis	GLUE, SST, MNLI
Question Answering	$x_{1:M}$,$x_{M:N}$	y span $[1:N]$	QA, Reading Comprehension	SQuAD, Natural Questions
Token Classification	$x_{1:N}$	$y_{1:N} \in \mathcal{C}^N$	NER, Tagging	OntoNotes, WNUT
Multiple Choice	$x_{1:N}, \mathcal{X}$	$y \in \mathcal{X}$	Text Selection	SWAG, ARC
Masked LM	$x_{1:N \setminus n}$	$x_n \in \mathcal{V}$	Pretraining	Wikitext, C4
Conditional Generation	$x_{1:N}$	$y_{1:M} \in \mathcal{V}^M$	Translation, Summarization	WMT, IWSLT, CNN/DM, XSum

شکل ۱۹ – لیست هد های مورد استفاده در کتابخانه

مدل هاب

هدف ترنسفورمرها تسهیل استفاده و توزیع آسان مدل های از پیش آموزش دیده است. یک مرحله پیش آموزش، تنظیم دقیق بسیاری از وظایف خاص را تسهیل می کند. مدل هاب دسترسی هر کاربر نهایی به مدل را برای استفاده با داده های خود آسان می کند. این مرکز اکنون شامل ۲۰۹۷ کاربر از سراسر جامعه است که از قبل آموزش دیده و تنظیم شده اند.

در حالی که مدل های اصلی مانند BERT و GPT-۲ همچنان محبوب هستند، مدل های تخصصی دیگر از جمله DistilBERT که برای کتابخانه توسعه داده شده است، اکنون به طور گسترده توسط جامعه دانلود می شوند. رابط کاربری مدل هاب به گونه ای طراحی شده است که ساده و در اختیار عموم جامعه باشد. برای آپلود یک مدل، هر کاربری می تواند یک حساب باز کند و از یک CMI برای تولید آرشیو متشکل از یک توکنایزر، ترنسفورمر و هد استفاده کند. این بسته ممکن است مدلی باشد که از طریق کتابخانه آموزش داده شده یا از سایر ابزارهای آموزشی گرفته شده باشد.

سپس این مدل ها ذخیره می شوند و یک نام متعارف به آنها داده می‌شود که کاربر می‌تواند از آن برای دانلود و کش کردن و اجرای مدل برای تنظیم دقیق یا استنتاج در دو خط کد استفاده کند. هنگامی که یک مدل در مدل هاب آپلود می شود، به طور خودکار یک صفحه فرود به آن داده می شود که ویژگی های اصلی، معماری و موارد استفاده آن را توصیف می کند. داده های خاص مدل اضافی را می توان از طریق یک کارت مدل ارائه کرد که ویژگی های آموزش، استناد به کار، مجموعه داده‌های مورد استفاده در طول پیش آموزش، و هرگونه هشدار در مورد سو گیری های شناخته شده در مدل و پیش بینی های آن را توضیح می‌دهد.

از آنجایی که مدل هاب مختص مدل های مبتنی بر ترنسفورمر است، می توانیم موارد استفاده ای را که برای مجموعه های مدل عمومی تر دشوار است، هدف قرار دهیم. به عنوان مثال، از آنجایی که هر مدل آپلود شده شامل داده های مربوط به ساختار خود است، صفحه مدل می‌تواند شامل استنتاج زنده باشد که به کاربران اجازه می‌دهد خروجی مدل ها را روی یک داده واقعی آزمایش کنند. علاوه بر این، صفحات مدل شامل پیوندهایی به سایر ابزارهای خاص مدل مانند علامت گذاری و مصور سازی است.

مطالعات جامعه ای

مدل هاب نحوه استفاده از ترنسفورمرها توسط ذینفعان مختلف جامعه را برجسته می کند. سه مورد استفاده خاص مشاهده شده را در عمل بیان می کنیم. سیستم های خاصی را که توسط کاربران با اهداف مختلف توسعه یافته اند، به دنبال تمایز معمار، آموزش دهنده و کاربر نهایی بررسی می‌کنیم.

مورد ۱: معماران مدل AllenAI، یک آزمایشگاه تحقیقاتی بزرگ NLP، یک مدل از پیش آموزش دیده جدید برای استخراج بهبود یافته از متون زیست پزشکی به نام SciBERT ایجاد کردند. آنها توانستند این مدل را با استفاده از داده‌های PubMed برای تولید یک مدل زبانی با نتایج پیشرفته در متن هدف آموزش دهند. آنها سپس از مدل هاب برای توزیع مدل و تبلیغ آن به عنوان بخشی از چالش CORD - COVID-۱۹ خود استفاده کردند.

مورد ۲: محققان در دانشگاه نیویورک علاقه مند به ایجاد یک بستر آزمایشی برای عملکرد ترنسفورمرها در انواع وظایف مختلف تشخیص معنایی بودند. چارچوب آن‌ها Jiant، به آنها اجازه می‌دهد تا با روش های مختلف پیش آموزش مدل ها و مقایسه خروجی هایشان آزمایش هایشان را انجام دهند. آن‌ها از API ترنسفورمرها به عنوان یک پیش فرض عمومی و با تنظیم دقیق در انواع مدل های مختلف استفاده کردند، که منجر به تحقیق در مورد ساختار BERT شد.

مورد ۳: کاربران Plot.ly، شرکتی که بر روی داشبورد و تجزیه و تحلیل کاربران تمرکز دارد، در به کارگیری مدلی برای خلاصه سازی خودکار اسناد علاقه مند بودند. آنها رویکردی را می خواستند که به خوبی مقیاس پذیر باشد و به کارگیری آن ساده باشد اما نیازی به آموزش یا تنظیم دقیق مدل نداشته باشد. آنها توانستند مدل هاب را جستجو کنند و DistilBART را بیابند، یک مدل خلاصه سازی از پیش آموزش دیده و تنظیم شده که برای استنتاج دقیق و سریع طراحی شده است. آنها قادر به اجرا و استقرار مدل به طور مستقیم از هاب بدون نیاز به تحقیق یا تخصص ML بودند.

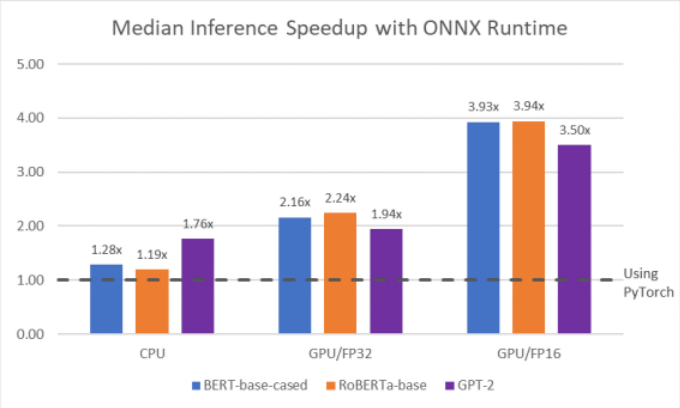
پیاده سازی

یکی از اهداف مهم ترنسفورمرها این است که بکارگیری کارآمد مدل در تولید را آسان کنند. کاربران مختلف نیازهای تولیدی متفاوتی دارند و پیاده سازی اغلب به حل چالش‌های بسیار متفاوتی نسبت به آموزش نیاز دارد. بنابراین کتابخانه چندین استراتژی مختلف برای پیاده سازی را امکان پذیر می کند.

یکی از ویژگی های اصلی این کتابخانه این است که مدل ها هم در PyTorch و هم در TensorFlow در دسترس هستند و بین هر دو فریم ورک قابلیت همکاری وجود دارد. یک مدل از پیش آموزش دیده در یکی از فریم ورک ها را می‌توان از طریق سریال سازی استاندارد ذخیره کرد و از فایل های ذخیره شده در چارچوب دیگر به طور یکپارچه مجددا بارگذاری کرد. این امر به ویژه تغییر از یک چارچوب به چارچوب دیگر را در طول عمر مدل (آموزش، خدمت، و غیره) آسان می کند.

هر فریم ورک دارای توصیه هایی برای پیاده سازی است. به عنوان مثال، در PyTorch، مدل ها با TorchScript، یک نمایش متوسط از یک مدل PyTorch که می‌تواند در پایتون به روشی کارآمدتر یا در یک محیط با کارایی بالا مانند C++ اجرا شود، سازگار هستند. بنابراین، مدل های تنظیم شده را می‌توان به محیط سازگار با پیاده سازی برد و از طریق TorchServing اجرا کرد. TensorFlow شامل چندین گزینه سرویس دهی در اکوسیستم خود است و می توان از آنها به طور مستقیم استفاده کرد.

ترنسفورمرها همچنین می توانند مدل ها را به قالب های شبکه عصبی میانی برای پیاده سازی بیشتر ارسال کنند. از تبدیل مدل ها از فرمت تبادل شبکه عصبی باز (ONNX) برای پیاده سازی پشتیبانی می کنند. این عمل نه تنها به مدل اجازه می دهد تا در قالبی استاندارد شده قابل اجرا شود، بلکه منجر به افزایش سرعت به صورت قابل توجهی نیز می شود. در شکل ۵ می توانیم عملکرد روش های مختلف را مشاهده کنیم.



شکل ۲۰ – مقایسه عملکرد روش های مختلف در فرمت های متمایز

با استفاده از این فرمت میانی، ONNX توانست نزدیک به ۴ برابر سرعت در این مدل به دست آورد. این تیم همچنین در حال آزمایش برروی دیگر فرمت ها مانند JAX/XLA و TVM است. در نهایت، از آنجایی که ترنسفورمرها در همه برنامه‌های NLP به طور گسترده تری مورد استفاده قرار می گیرند، استقرار در دستگاه های لبه ای مانند تلفن ها یا وسایل الکترونیکی خانگی اهمیت فزاینده ای دارد. مدل ها می‌توانند از آداپتور هایی برای تبدیل مدل ها به وزن های CoreML استفاده کنند که می‌توانند در یک برنامه iOS جاسازی شوند تا یادگیری ماشینی روی لبه را فعال کنند. کد نیز در دسترس است. روش های مشابهی را می توان برای دستگاه های اندرویدی استفاده کرد.

مقاله چهارم

مقدمه

در این گزارش یک مدل بازنمایی زبان جدید به نام BERT را معرفی می کنیم که مخفف Bidirectional Encoder Representations from Transformers است. برخلاف مدل های نمایش زبان اخیر، BERT برای پیش آموزش نمایش های دو سویه عمیق از متن بدون برچسب با شرطی کردن مشترک در زمینه چپ و راست در همه لایه ها طراحی شده است. در نتیجه، مدل BERT از پیش آموزش دیده را می توان تنها با یک لایه خروجی اضافی تنظیم کرد تا مدل های پیشرفته ای را برای طیف وسیعی از وظایف ایجاد کند مانند پاسخ به سؤال و استنتاج زبان، بدون تغییرات اساسی در معماری.

نشان داده شده است که پیش آموزش مدل زبان برای بهبود بسیاری از وظایف پردازش زبان طبیعی مؤثر است. اینها شامل وظایف سطح جمله ای مانند استنتاج و بازنویسی زبان طبیعی است که هدف آنها پیش بینی روابط بین جملات با تجزیه و تحلیل و همچنین وظایف سطح نشانه مانند تشخیص موجودیت نامگذاری شده و پاسخ به سؤال است.

دو استراتژی موجود برای اعمال بازنمایی های زبانی از پیش آموزش دیده برای کارها وجود دارد: ۱) مبتنی بر ویژگی ها ۲) تنظیم دقیق. رویکرد مبتنی بر ویژگی ها، مانند ELMo، از معماری های خاص کار استفاده می‌کند که شامل نمایش های از پیش آموزش دیده شده به عنوان ویژگی های اضافی است. رویکرد تنظیم دقیق، مانند ترانسفورمرها از ترانسفورمر پیش آموزش دیده (OpenAI GPT)، حداقل پارامترهای مربوط به کار را به دست می آورد و در کارها به سادگی با تنظیم دقیق همه پارامترهای از پیش آموزش دیده آموزش داده می شود. این دو رویکرد در طول پیش آموزش، عملکرد و هدف یکسانی دارند، جایی که از مدل های زبانی یک طرفه برای یادگیری بازنمایی های زبان عمومی استفاده می‌کنند.

اثبات می کنیم که تکنیک های فعلی، قدرت نمایش های از پیش آموزش دیده را محدود می کنند، به ویژه برای رویکردهای تنظیم دقیق. محدودیت اصلی این است که مدل های زبان استاندارد یک جهته هستند و این انتخاب معماری هایی را که می‌توان در طول دوره پیش آموزش استفاده کرد، محدود می‌کند. چنین محدودیت هایی برای وظایف سطح جمله اصلا بهینه نیستند و می‌توانند هنگام اعمال رویکردهای مبتنی بر تنظیم دقیق برای وظایف سطح نشانه مانند پاسخ گویی به سؤال، که در آن اضافه کردن محتوا از هر دو جهت بسیار مهم است، بسیار مضر باشد.

BERT با استفاده از یک هدف پیش آموزشی «مدل زبان ماسک‌شده» (MLM) و با الهام از کار کلوز، محدودیت های یکپارچه سازی را که قبلاً ذکر شد، کاهش می‌دهد. مدل زبان ماسک دار به طور تصادفی برخی از نشانه ها را از ورودی پنهان می‌کند و هدف آن پیش بینی واژگان اصلی کلمه پوشانده شده تنها بر اساس محتوای آن است. برخلاف پیش آموزش مدل زبان از چپ به راست، هدف MLM نمایش را قادر می سازد تا محتوای چپ و راست را با هم ترکیب کند، که به ما امکان می‌دهد یک ترانسفورمر دو طرفه عمیق را از پیش آموزش دهیم. علاوه بر مدل زبان پوشانده شده، ما همچنین از یک کار «پیش بینی جمله بعدی» استفاده می‌کنیم که به طور مشترک بازنمایی های جفت متن را آموزش می‌دهد. کار های انجام شده به شرح زیر است:

۱) اهمیت پیش آموزش دو طرفه را برای بازنمایی زبان نشان می‌دهیم. برخلاف روش Radfor که از مدل های زبانی یک طرفه برای پیش آموزش استفاده می‌کند، BERT از مدل های زبان ماسک دار برای فعال کردن نمایش های عمیق دوطرفه از پیش آموزش دیده استفاده می‌کند که برخلاف روش پیترز است که از الحاق دو مدل کم عمق از چپ به راست و از راست به چپ که به طور مستقل آموزش دیده اند، استفاده می کند.
۲) نشان می‌دهیم که بازنمایی های از پیش آموزش دیده شده نیاز به بسیاری از معماری های ویژه وظایف مهندسی شده را کاهش می‌دهند. BERT اولین مدل نمایش مبتنی بر تنظیم دقیق است که عملکردی پیشرفته را در مجموعه بزرگی از وظایف در سطح جمله و سطح نشانه دارد و بر بسیاری از معماری های خاص برتری دارد.
۳) BERT یازده وظیفه NLP را بهبود می بخشد.

کارهای مرتبط

رویکردهای مبتنی بر ویژگی بدون نظارت

یادگیری بازنمایی های قابل کاربرد واژه ها برای دهه ها یک حوزه فعال تحقیقاتی از جمله روش های غیر عصبی و عصبی بوده است. جاسازی های کلمه از پیش آموزش دیده بخشی جدایی ناپذیر از سیستم های NLP مدرن هستند که نسبت به جاسازی هایی که از ابتدا آموخته اند، پیشرفت های قابل توجهی دارند. برای پیش آموزش بردار های جاسازی کلمه، اهداف مدل سازی زبان از چپ به راست و همچنین اهدافی برای تمییز کلمات صحیح از نادرست در محتوای چپ و راست استفاده شده است. این رویکرد ها به جزئیات کلان تر، مانند جاسازی جمله یا جاسازی پاراگراف تعمیم داده شده اند.

برای آموزش نمایش جملات، کار قبلی از اهداف برای رتبه بندی جملات بعدی کاندید و تولید از چپ به راست کلمات جمله بعدی با ارائه نمایشی از جمله قبلی یا حذف نویز از اهداف مشتق شده از رمز گذار

خودکار استفاده کرده است. ELMo جستجوی مجدد جاسازی کلمه سنتی را در ابعاد متفاوتی تعمیم می دهد و ویژگی های حساس به محتوا را برای یک مدل زبان از چپ به راست و راست به چپ استخراج می کند. بازنمایی متنی هر نشانه، الحاق بازنمایی های چپ به راست و راست به چپ است. هنگام ادغام تعبیه های کلمه متنی با معماری های خاص موجود، ELMo وضعیت را برای چندین معیار اصلی NLP از جمله پاسخ به سؤال، تجزیه و تحلیل احساسات و شناسایی موجودیت نام گذاری شده ارتقا می‌دهد.

رویکردهای مبتنی بر تنظیم دقیق بدون نظارت

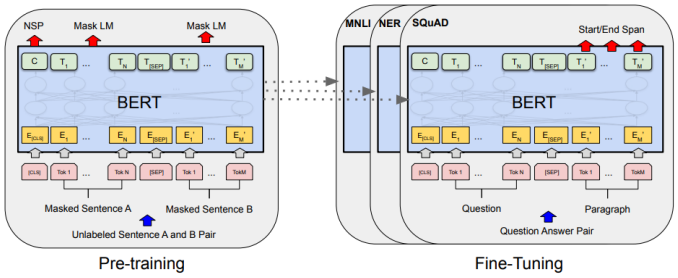
همانند رویکردهای مبتنی بر ویژگی، اولین مورد در این جهت فقط پارامترهای از پیش آموزش داده شده واژه را از متن بدون برچسب استخراج می کند. اخیراً، رمز گذار های جملات یا اسنادی که بازنمایی نشانه های متنی را تولید می کنند، از پیش آموزش داده شده اند و برای یک کار نظارت شده، به خوبی تنظیم شده اند. مزیت این رویکرد ها این است که پارامترهای کمی را باید از ابتدا یاد گرفت. حداقل تا حدی به دلیل این مزیت، OpenAI GPT به نتایج بسیار پیشرفته ای در بسیاری از وظایف سطح جمله در معیار GLUE دست یافت. مدل سازی زبان از چپ به راست و اهداف رمز گذار خودکار برای پیش آموزش این مدل ها استفاده شده است.

انتقال یادگیری از داده های نظارت شده

کارهایی وجود دارد که انتقال مؤثر از وظایف نظارت شده با مجموعه داده های بزرگ، مانند استنتاج زبان طبیعی و ترجمه ماشینی را نشان می دهد. تحقیقات بینایی کامپیوتری همچنین اهمیت انتقال یادگیری از مدل های بزرگ از پیش آموزش دیده را نشان می دهد. جایی که یک راه کار موثر، تنظیم دقیق مدل های از پیش آموزش دیده با ImageNet است.

BERT

در این چارچوب دو مرحله وجود دارد: پیش آموزش و تنظیم دقیق. در طول دوره پیش آموزش، مدل بر روی داده های بدون برچسب بر اساس کارهای مختلف آموزش داده می شود. برای تنظیم دقیق، ابتدا مدل BERT با پارامترهای از پیش آموزش داده شده مقدار دهی اولیه می شود و تمام پارامتر ها با استفاده از داده های برچسب گذاری شده از کارهای پایین دست تنظیم می شوند. هر کار پایین دستی دارای مدل های جداگانه ای است که به خوبی تنظیم شده‌اند، حتی اگر آن‌ها با همان پارامترهای از پیش آموزش دیده شده آغاز شوند. نمونه ای از این روش را در شکل ۱ مشاهده می کنیم.



شکل ۲۱ – مثال عملی BERT

ویژگی متمایز BERT معماری یکپارچه آن در کارهای مختلف است. حداقل تفاوت بین معماری از پیش آموزش دیده و معماری نهایی پایین دستی وجود دارد.

معماری مدل

معماری مدل BERT یک ترانسفورمر و کد گذار چند لایه دو جهته است که در کتابخانه tensor۲tensor منتشر شده است. در این کار، تعداد لایه ها (یعنی بلوک های ترانسفورمر) را L، اندازه پنهان را H و تعداد هد ها را با A نشان می‌دهیم. ما در درجه اول نتایج را در دو اندازه مدل گزارش می‌کنیم: $BERT_{BASE}$ (L=۱۲، H = ۷۶۸، A = ۱۲، مجموع پارامتر ها = ۱۱۰ M) و $BERT_{LARGE}$ (L=۲۴، H = ۱۰۲۴، A = ۱۶، مجموع پارامترهای = ۳۴۰ M).

اندازه $BERT_{BASE}$ برای مقایسه با اندازه مدل OpenAI GPT یکسان انتخاب شد. با این حال، ترانسفورمر BERT از توجه دو طرفه استفاده می کند، در حالی که GPT از توجه محدود استفاده می کند که در آن هر توکن فقط می تواند به محتوای سمت چپ خود توجه کند.

نمایش ورودی و خروجی

برای اینکه BERT انواع وظایف پایین دستی را انجام دهد، نمایش ورودی ما می‌تواند به طور واضح یک جمله و یک جفت جمله را در یک دنباله توکنی نمایش دهد. یک جمله می‌تواند به جای یک جمله زبانی واقعی، یک گستره دلخواه از متن پیوسته باشد. یک دنباله به دنباله توکن قرار داده شده همراه BERT اشاره دارد که ممکن است یک جمله تنها یا دو جمله در کنار هم باشد. از WordPiece با تعداد واژگان ۳۰۰۰۰ تایی استفاده می‌کنیم. اولین نشانه هر دنباله همیشه یک نشانه طبقه بندی ویژه ([CLS]) است. حالت پنهان نهایی مربوط به این نشانه به عنوان نمایش توالی کل برای کارهای طبقه بندی استفاده می شود. جفت جملات در کنار هم در یک دنباله واحد قرار می گیرند. جملات را به دو صورت متمایز می کنیم. ابتدا آنها را با یک نشانه مخصوص ([SEP]) جدا می کنیم. در ادامه یک جاسازی یاد گرفته شده را به هر نشانه اضافه می کنیم که

نشان می دهد به جمله A یا جمله B تعلق دارد. ورودی را به صورت E، بردار پنهان نهایی نشانه [CLS] ویژه را با $C \in R^H$ و بردار پنهان نهایی را برای کد ورودی i ام به صورت $T_i \in R^H$ نشان می دهیم. برای یک نشانه داده شده، نمایش ورودی آن با جمع کردن توکن، بخش و جاسازی موقعیت مربوطه ساخته می‌شود.

پیش آموزش BERT

BERT را با استفاده از دو وظیفه بدون نظارت که در این بخش توضیح داده شده است، پیش آموزش می دهیم. این مرحله در قسمت سمت چپ شکل ۱ نشان داده شده است.

وظیفه شماره ۱- *LM* ماسک شده

به طور شهودی، منطقی است که باور کنیم یک مدل دو سویه عمیق به شدت قدرتمندتر از مدل چپ به راست یا یک الحاق کم عمق یک مدل چپ به راست و راست به چپ است. متأسفانه، مدل های زبان شرطی استاندارد را فقط می‌توان از چپ به راست یا از راست به چپ آموزش داد، زیرا شرطی سازی دو سویه به هر کلمه اجازه می‌دهد مستقیماً خود را ببیند و مدل می‌تواند کلمه هدف را در یک محتوای چند لایه ای بی اهمیت پیش بینی کند. به منظور آموزش یک نمایش دو طرفه عمیق، ما به سادگی درصدی از نشانه های ورودی را به صورت تصادفی پنهان می کنیم و سپس آن نشانه های ماسک شده را پیش بینی می کنیم. ما از این روش به عنوان LM ماسک شده (MLM) یاد می کنیم، اگرچه در ادبیات اغلب به عنوان عملیات کلوز از آن یاد می شود.

بردار های پنهان نهایی مربوط به توکن های ماسک مانند یک LM استاندارد به یک softmax خروجی بر روی واژگان وارد می شوند. در تمام آزمایش هایمان، ۱۵ درصد از کل WordPiece را در هر دنباله به‌طور تصادفی می پوشانیم. برخلاف حذف نویز رمز گذار های خودکار، ما فقط کلمات پوشانده شده را پیش بینی می کنیم تا اینکه کل ورودی را بازسازی کنیم. اگرچه این به ما امکان می دهد یک مدل از پیش آموزش داده شده دوطرفه به دست آوریم، اما یک نقطه ضعف این است که ما در حال ایجاد عدم تطابق بین پیش آموزش و تنظیم دقیق هستیم، زیرا نشانه [MASK] در تنظیم دقیق ظاهر نمی شود. برای کاهش این موضوع، ما همیشه کلمات "ماسک شده" را با نشانه واقعی [MASK] جایگزین نمی کنیم. مولد داده های آموزشی ۱۵ درصد از موقعیت های نشانه را به صورت تصادفی برای پیش بینی انتخاب می کند. اگر علامت i انتخاب شود، علامت i را با (۱) علامت [MASK] در ۸۰٪ مواقع (۲) یک نشانه تصادفی در ۱۰٪ موارد (۳) علامت i-امین بدون تغییر در ۱۰ درصد مواقع جایگزین می کنیم. سپس T_i برای پیش بینی نشانه اصلی با از خطای آنتروپی متقاطع استفاده می‌شود.

وظیفه شماره ۲ - پیش بینی جمله بعدی (*NSP*)

بسیاری از وظایف مهم پایین دستی مانند پاسخ به پرسش و استنتاج زبان طبیعی بر اساس درک رابطه بین دو جمله است که مستقیماً توسط مدل سازی زبان مشخص نمی شود. به منظور آموزش مدلی که روابط جملات را درک می کند، برای یک کار باینری شده پیش بینی جمله قبلی مدل را پیش آموزش می دهیم که به سادگی می تواند از هر پیکره تک زبانی نیز تولید شود. مخصوصاً هنگام انتخاب جملات A و B. برای مثال در پیش آموزش، ۵۰٪ مواقع B جمله واقعی بعدی است که به دنبال A (با برچسب IsNext) قرار دارد و ۵۰٪ از مواقع یک جمله تصادفی از مجموعه (با برچسب NotNext) است. همانطور که در شکل ۱ نشان داده شد، C برای پیش بینی جمله بعدی (NSP) استفاده می شود.

پیش آموزش داده

پیش آموزش تا حد زیادی از ادبیات موجود در مورد پیش آموزش مدل زبان پیروی می‌کند. برای مجموعه پیش آموزشی از BooksCorpus (۸۰۰ میلیون کلمه) و ویکی‌پدیای انگلیسی (۲۵۰۰ میلیون کلمه) استفاده می‌کنیم. برای ویکی‌پدیا ما فقط متن ها را استخراج می‌کنیم و فهرست ها، جداول و سرصفحه ها را نادیده می گیریم. برای استخراج دنباله های طولانی به هم پیوسته، استفاده از پیکره در سطح سند به جای پیکره درهم آمیخته در سطح جمله بسیار مهم است.

تنظیم دقیق BERT

تنظیم دقیق ساده است زیرا مکانیسم توجه به خود در ترنسفورمر به BERT اجازه می دهد تا بسیاری از وظایف پایین دستی را (خواه شامل متن واحد باشد یا جفت متن) با تعویض ورودی ها و خروجی های مناسب مدل سازی کند. برای کاربردهایی که شامل جفت های متنی می شوند، یک الگوی رایج رمز گذاری مستقل جفت های متنی قبل از اعمال توجه متقاطع دو طرفه است. BERT در عوض از مکانیسم توجه به خود برای یکسان کردن این دو مرحله استفاده می کند، زیرا رمز گذاری یک جفت متن به هم پیوسته با توجه به خود به طور موثر شامل توجه متقاطع واکنشی بین دو جمله می شود. برای هر کار، ما به سادگی ورودی ها و خروجی های مربوط به کار را به BERT وصل می کنیم و تمام پارامتر ها را به طور کامل تنظیم می کنیم. در ابتدا، جمله A و جمله B از قبل از آموزش مشابه هستند:

۱) جفت جملات به صورت نقل قول

۲) جفت فرضیه- مقدمه در مستلزم

۳) جفت سؤال - گذر در پاسخ به سؤال

۴) انحطاط جفت متن-Ø در طبقه بندی متن یا برچسب گذاری توالی.

در خروجی، بازنمایی های نماد به یک لایه خروجی برای وظایف سطح نشانه، مانند برچسب گذاری دنباله ای یا پاسخگویی به سؤال، و نمایش [CLS] برای طبقه بندی به لایه خروجی وارد می‌شود. در مقایسه با پیش آموزش، تنظیم دقیق نسبتاً ارزان است.

مطالعات فرسایشی

تأثیر عملیات پیش آموزشی

با ارزیابی دو هدف پیش آموزش با استفاده از داده‌های پیش آموزش، طرح تنظیم دقیق، و پارامتر ها مانند $BERT_{BASE}$، اهمیت واقعیت دوگانه عمیق BERT را نشان می‌دهیم:

بدون NSP: یک مدل دو جهته که با استفاده از LM ماسک دار (MLM) اما بدون پیش بینی جمله بعدی (NSP) آموزش داده می‌شود.

LTR و بدون NSP: یک مدل فقط با محتوای چپ که با استفاده از LM استاندارد از چپ به راست (LTR) به جای MLM آموزش داده شده است. محدودیت فقط سمت چپ نیز در تنظیم دقیق اعمال شد، زیرا حذف آن باعث ایجاد یک عدم تطابق قبل از آموزش/تنظیم دقیق شد که عملکرد پایین دست را کاهش داد. علاوه بر این، این مدل بدون وظیفه NSP از قبل آموزش داده شد. این به طور مستقیم با OpenAI GPT قابل مقایسه است، اما با استفاده از مجموعه داده های آموزشی بزرگتر، نمایش ورودی و طرح تنظیم دقیق ما.

ابتدا تأثیرات ناشی از وظیفه NSP را بررسی می کنیم. در جدول ۱، نشان می دهیم که حذف NSP به طور قابل توجهی به عملکرد QNLI، MNLI و SQuAD۱.۱ لطمه می زند. در مرحله بعد، ما تأثیر آموزش دو طرفه را با مقایسه No NSP با LTR & No NSP ارزیابی می‌کنیم. مدل LTR در همه وظایف بدتر از مدل MLM عمل می کند و با افت زیادی در MRPC و SQuAD همراه است.

جدول ۱ – مقایسه عملکرد کار های مختلف در دیتاست های متمایز

	Dev Set				
Tasks	MNLI-m (Acc)	QNLI (Acc)	MRPC (Acc)	SST-2 (Acc)	SQuAD (F1)
$BERT_{BASE}$	84.4	88.4	86.7	92.7	88.5
No NSP	83.9	84.9	86.5	92.6	87.9
LTR & No NSP	82.1	84.3	77.5	92.1	77.8
+ BiLSTM	82.1	84.1	75.7	91.6	84.9

برای SQuAD به طور شهودی واضح است که یک مدل LTR در پیش بینی نشانه ها ضعیف عمل می‌کند، زیرا حالت های پنهان سطح توکن هیچ محتوا سمت راستی ندارند. به منظور ایجاد حسن نیت در تقویت سیستم LTR، یک BiLSTM به صورت تصادفی در بالا اضافه کردیم. این به طور قابل توجهی نتایج را در SQuAD بهبود می بخشد، اما نتایج هنوز به مراتب بدتر از مدل های دو طرفه از قبل آموزش دیده هستند.

می دانیم که آموزش مدل های LTR و RTL جداگانه و نمایش هر توکن به عنوان الحاق دو مدل همانطور که ELMo انجام می دهد، ممکن است. با این حال:

۱) این دو برابر گرانتر از یک مدل دو طرفه است

۲) این برای کارهایی مانند QA غیر شهودی است، زیرا مدل RTL نمی تواند پاسخ را به سؤال مشروط کند.

۳) این به شدت کمتر از یک مدل دو جهته عمیق است، زیرا می تواند از بافت چپ و راست در هر لایه استفاده کند.

اثر اندازه مدل

تعدادی مدل BERT را با تعداد متفاوتی از لایه‌ها، واحدهای پنهان و هد های توجه آموزش دادیم و سایر موارد مانند حالت قبلی بود. در جدول ۲، میانگین دقت Dev Set را از ۵ راه اندازی مجدد تصادفی تنظیم دقیق گزارش می کنیم. ما می توانیم ببینیم که مدل های بزرگتر منجر به بهبود دقت دقیق در هر چهار مجموعه داده می شوند، حتی برای MRPC که فقط ۳۶۰۰ نمونه آموزش برچسب گذاری شده دارد و تفاوت اساسی با وظایف قبل از آموزش دارد. همچنین شاید تعجب آور باشد که ما قادر به دستیابی به چنین پیشرفت های قابل توجهی در بالای مدل هایی هستیم که نسبت به ادبیات موجود کاملاً آماده هستند.

مدت هاست که مشخص شده است که افزایش اندازه مدل منجر به بهبود مستمر در کارهای مقیاس بزرگ مانند ترجمه ماشینی و مدل سازی زبان می‌شود، که با سردرگمی LM داده‌های آموزشی نگه داشته شده در جدول ۲ نشان داده شده است.

جدول ۲ – اثر اندازه مدل بر پارامترهای عملکردی

Hyperparams				Dev Set Accuracy		
#L	#H	#A	LM (ppl)	MNLI-m	MRPC	SST-2
3	768	12	5.84	77.9	79.8	88.4
6	768	3	5.24	80.6	82.2	90.7
6	768	12	4.68	81.9	84.8	91.3
12	768	12	3.99	84.4	86.7	92.9
12	1024	16	3.54	85.7	86.9	93.3
24	1024	16	3.23	86.6	87.8	93.7

با این حال، معتقدیم که این اولین کاری است که به طور قانع کننده نشان می‌دهد که مقیاس بندی اندازه های مدل شدید منجر به پیشرفت های بزرگ در کارهای مقیاس بسیار کوچک می‌شود، مشروط بر اینکه مدل به اندازه کافی از پیش آموزش داده شده باشد.

رویکرد مبتنی بر ویژگی با BERT

تمام نتایج BERT ارائه شده تا کنون از رویکرد تنظیم دقیق استفاده کرده اند، که در آن یک لایه طبقه بندی ساده به مدل از پیش آموزش دیده شده اضافه می‌شود و همه پارامتر ها به طور مشترک در یک کار پایین دستی تنظیم دقیق می شوند. با این حال، رویکرد مبتنی بر ویژگی، که در آن ویژگی های ثابت از مدل از پیش آموزش دیده استخراج می شوند، دارای مزایای خاصی است. اول، همه وظایف را نمی توان به راحتی با معماری رمز گذار ترانس نشان داد، و بنابراین نیاز به یک معماری مدل خاص کار اضافه می‌شود. دوم، مزایای محاسباتی عمده ای برای پیش محاسبه یک نمایش گران از داده های آموزشی یک بار و سپس اجرای آزمایش های زیادی با مدل های ارزان تر در بالای این نمایش وجود دارد.

در این بخش، این دو رویکرد را با اعمال BERT در عملیات CoNLL-۲۰۰۳ مقایسه می‌کنیم. در ورودی BERT، از یک مدل WordPiece با حفظ حروف استفاده می‌کنیم و حداکثر زمینه سند ارائه شده توسط داده‌ها را درج می‌کنیم. با پیروی از تمرین استاندارد، ما این را به عنوان یک کار برچسب گذاری فرموله می کنیم، اما از یک لایه CRF در خروجی استفاده نمی کنیم. ما از نمایش اولین زیر نشانه به عنوان ورودی طبقه بندی کننده سطح نشانه روی مجموعه برچسب NER استفاده می‌کنیم.

برای حذف رویکرد تنظیم دقیق، رویکرد مبتنی بر ویژگی را با استخراج فعال سازی ها از یک یا چند لایه بدون تنظیم دقیق هیچ پارامتر BERT اعمال می‌کنیم. این بستر های EM متنی به عنوان ورودی به یک BiLSTM دو لایه ۷۶۸ بعدی به طور تصادفی اولیه قبل از لایه طبقه بندی استفاده می شود. نتایج در جدول ۳ ارائه شده است. $\text{BERT}_{\text{LARGE}}$ به صورت رقابتی با روش های پیشرفته عمل می کند. بهترین روش عملکرد، نمایش های نشانه ای را از چهار لایه مخفی بالای ترانسفورماتور از پیش آموزش دیده به هم پیوند می‌دهد، که تنها ۰.۳F۱ از تنظیم دقیق کل مدل عقب است. این نشان می دهد که BERT برای هر دو روش تنظیم دقیق و رویکردهای مبتنی بر ویژگی موثر است.

جدول ۳ - نتایج شناسایی موجودیت نامگذاری شده CoNLL-۲۰۰۳

System	Dev F1	Test F1
ELMo (Peters et al., 2018a)	95.7	92.2
CVT (Clark et al., 2018)	-	92.6
CSE (Akbik et al., 2018)	-	**93.1**
Fine-tuning approach		
$\text{BERT}_{\text{LARGE}}$	96.6	92.8
$\text{BERT}_{\text{BASE}}$	96.4	92.4
Feature-based approach ($\text{BERT}_{\text{BASE}}$)		
Embeddings	91.0	-
Second-to-Last Hidden	95.6	-
Last Hidden	94.9	-
Weighted Sum Last Four Hidden	95.9	-
Concat Last Four Hidden	96.1	-
Weighted Sum All 12 Layers	95.5	-