$\boxed{1.1}$

هر دو گزینه می توانند انتخاب های مناسبی باشند. به طور کلی LSTM از GRU قوی تر است و به نتایج برابر یا بهتری می رسد
اما مزیت GRU ساده تر بودن آن است. GRU پارامتر های کمتری دارد و سریع تر است و هزینه محاسباتی کمتری دارد.
همچنین می توان این طور در نظر گرفت که LSTM برای sequence های طولانی تر مقداری بهتر است و long term dependency ها را بهتر حفظ می کند.
بنابراین بسته به مسئله می توانند بررسی کنند که آیا GRU به نتیجه دلخواهشان می رسد یا نه و اگر نتایج بهتری خواستند و توان پردازشی آن را داشتند از LSTM استفاده کنند.

$\boxed{1.2}$

خیر خرید کارت گرافیک را پیشنهاد نمی دهم. به احتمال بالا دلیل اصلی کندی محاسبات، ذات sequential معماری های LSTM و GRU است.
با توجه به اینکه تسک این محقق sentiment analysis است، می دانیم که تمام توکن های هر sequence را به صورت یکجا در اختیار داریم.
اما با این دو معماری به صورت ترتیبی آن ها را پردازش می کنیم.
به عنوان روش جایگزین می توان از attention و معماری transformer استفاده کرد که انکودر آن تمام توکن های ورودی را به صورت موازی پردازش می کند

$\boxed{1.3.a}$

$*$ ← ضرب element wise
$@$ ← ضرب ماتریسی

$$c^t = c^{t-1} * \Gamma_f^t + \Gamma_u^t * \tilde{c}^t$$

$$a^t = \tanh(c^t) * \Gamma_o^t$$

$$\hat{y}^t = \mathrm{softmax}(W_y @ a^t + b_y)$$

$$\tilde{c}^t = \tanh\Big(\overbrace{\begin{bmatrix} W_{ca} & W_{cx} \end{bmatrix}}^{W_c}_{K\times(K+d)} @ \begin{bmatrix} a^{t-1} \\ x^t \end{bmatrix}_{(K+d)\times 1} + b_c\Big)$$

$$\Gamma_o^t = \sigma\Big(\begin{bmatrix} W_{oa} & W_{ox} \end{bmatrix} @ \begin{bmatrix} a^{t-1} \\ x^t \end{bmatrix} + b_o\Big)$$

$\Gamma_u^t =$ مشابه بالا

$\Gamma_f^t =$ مشابه بالا

$\boxed{1.3.b}$

فرض می‌کنیم بردار های a^t , c^t دارای طول k باشند و بردار x^t دارای طول d و $\hat{y}^t$ دارای طول l

$$\frac{\partial L}{\partial a^t} = \left[W_y^T \right]_{k\times l} @ \left[\hat{y}^t - y^t \right]_{l\times 1} + \left[da_{next} \right]_{k\times 1}$$

$$\frac{\partial L}{\partial c^t} = \left[\frac{\partial L}{\partial a^t} \right]_{k\times 1} * \left[\Gamma_o^t \right]_{k\times 1} * \left[1 - \tanh^2(c^t) \right]_{k\times 1} + \left[dc_{next} \right]_{k\times 1}$$

$$\frac{\partial L}{\partial W_o} = \underbrace{\left(\left[\frac{\partial L}{\partial a^t} \right]_{k\times 1} * \left[\tanh(c^t) \right]_{k\times 1} * \left[\Gamma_o^t * (1 - \Gamma_o^t) \right]_{k\times 1} \right)}_{\frac{\partial L}{\partial \gamma_o^t}} @ \left[\begin{bmatrix} a^{t-1} \\ x^t \end{bmatrix}^T \right]_{1\times (k+d)}$$

$$\frac{\partial L}{\partial b_o} = \left[\frac{\partial L}{\partial \gamma_o^t} \right]_{k\times 1}$$

$$\frac{\partial L}{\partial W_c} = \underbrace{\left(\left[\frac{\partial L}{\partial c^t} \right]_{k\times 1} * \left[\Gamma_u^t \right]_{k\times 1} * \left[1 - \underbrace{\tanh^2(p\tilde{c}^t)}_{(\tilde{c}^t)^2} \right]_{k\times 1} \right)}_{\frac{\partial L}{\partial p\tilde{c}^t}} @ \left[\begin{bmatrix} a^{t-1} \\ x^t \end{bmatrix}^T \right]_{1\times (k+d)}$$

$$\frac{\partial L}{\partial b_c} = \left[\frac{\partial L}{\partial p\tilde{c}^t} \right]_{k\times 1}$$

$$\frac{\partial L}{\partial W_u} = \underbrace{\left(\left[\frac{\partial L}{\partial c^t} \right]_{k\times 1} * \left[\tilde{c}^t \right]_{k\times 1} * \left[\Gamma_u^t * (1 - \Gamma_u^t) \right]_{k\times 1} \right)}_{\frac{\partial L}{\partial \gamma_u^t}} @ \left[\begin{bmatrix} a^{t-1} \\ x^t \end{bmatrix}^T \right]_{1\times (k+d)}$$

$$\frac{\partial L}{\partial b_u} = \left[\frac{\partial L}{\partial \gamma_u^t} \right]_{k\times 1}$$

$\boxed{1.3.b}$ ادامه

$$\frac{\partial L}{\partial W_f} = \underbrace{\left(\left[\frac{\partial L}{\partial c^t}\right]_{k\times 1} * \left[c^{t-1}\right]_{k\times 1} * \left[\Gamma_f^t * (1-\Gamma_f^t)\right]_{k\times 1} \right)}_{\frac{\partial L}{\partial \gamma_f^t}} @ \left[\begin{bmatrix} a^{t-1} \\ x^t \end{bmatrix}^T\right]_{1\times(k+d)}$$

$$\frac{\partial L}{\partial b_f} = \left[\frac{\partial L}{\partial \gamma_f^t}\right]_{k\times 1}$$

$$\frac{\partial L}{\partial W_y} = \left[\hat{y}^t - y^t\right]_{l\times 1} @ \left[\left[a^t\right]^T\right]_{1\times k}$$

$$\frac{\partial L}{\partial b_y} = \left[\hat{y}^t - y^t\right]_{l\times 1}$$

همچنین برای upstream هایی که به عقب فرستاده می‌شود داریم :

$$\frac{\partial L}{\partial c^{t-1}} = \left[\frac{\partial L}{\partial c^t}\right]_{k\times 1} * \left[\Gamma_f^t\right]_{k\times 1}$$

$$\frac{\partial L}{\partial a^{t-1}} = \left[W_{oa}^T\right]_{k\times k} @ \left[\frac{\partial L}{\partial \gamma_o^t}\right]_{k\times 1} + \left[W_{ca}^T\right]_{k\times k} @ \left[\frac{\partial L}{\partial p\tilde{c}^t}\right]_{k\times 1} + \left[W_{ua}^T\right]_{k\times k} @ \left[\frac{\partial L}{\partial \gamma_u^t}\right]_{k\times 1} + \left[W_{fa}^T\right]_{k\times k} @ \left[\frac{\partial L}{\partial \gamma_f^t}\right]_{k\times 1}$$

2.1

$l = len(dict) = 30,000$

$N = 2048$ طول بیشترین دنباله ورودی

$h = 4$ تعداد head ها

$d_k = \frac{768}{h} = 192$

$B = 32$

$d_{model} = 1024$

* همان طور که در paper آن ذکر شده است، با توجه به اینکه در معماری transformer از residual connection استفاده شده است و همچنین اینکه می‌دانیم layer norm ابعاد را تغییر نمی‌دهد، بنابراین تمامی ورودی‌ها و خروجی‌ها دارای بعد $(B \times N \times d_{model})$ هستند به غیر از ورودی embedding در انکودر و دیکودر که هنوز متن تبدیل به embedding ها نشده است (پس از اینکه هر توکن با استفاده از dictionary به یک index تبدیل شد، این لایه آن را به embedding تبدیل می‌کند پس شاید بتوان ورودی لایه امبدینگ را $(B \times N \times \underbrace{1}_{index})$ در نظر گرفت)

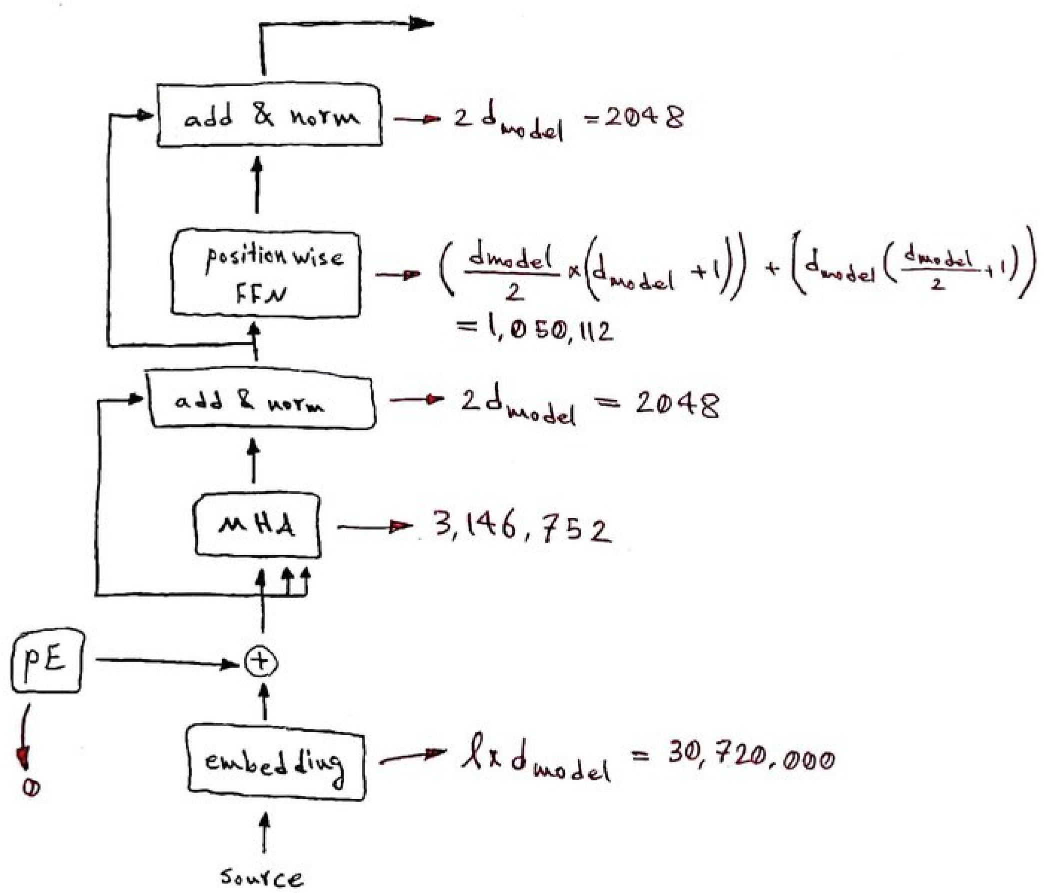
همچنین خروجی نهایی دیکودر نیز دارای بعد نهایی 30,000 تایی به جای d_{model} است

بنابراین ابعاد خواسته شده در صورت سوال برابر است با:

$$(B \times N \times d_{model}) = (32 \times 2048 \times 1024)$$

2.2

encoder



multi head attention params:

$$h\,(d_{model} \times d_k \times 3) \; \begin{Bmatrix} Q \\ K \\ V \end{Bmatrix}$$

$+$

$$d_{model}\,(d_k \times h) \quad \text{linear}$$

$+$

$$d_{model} \quad \text{linear bias}$$

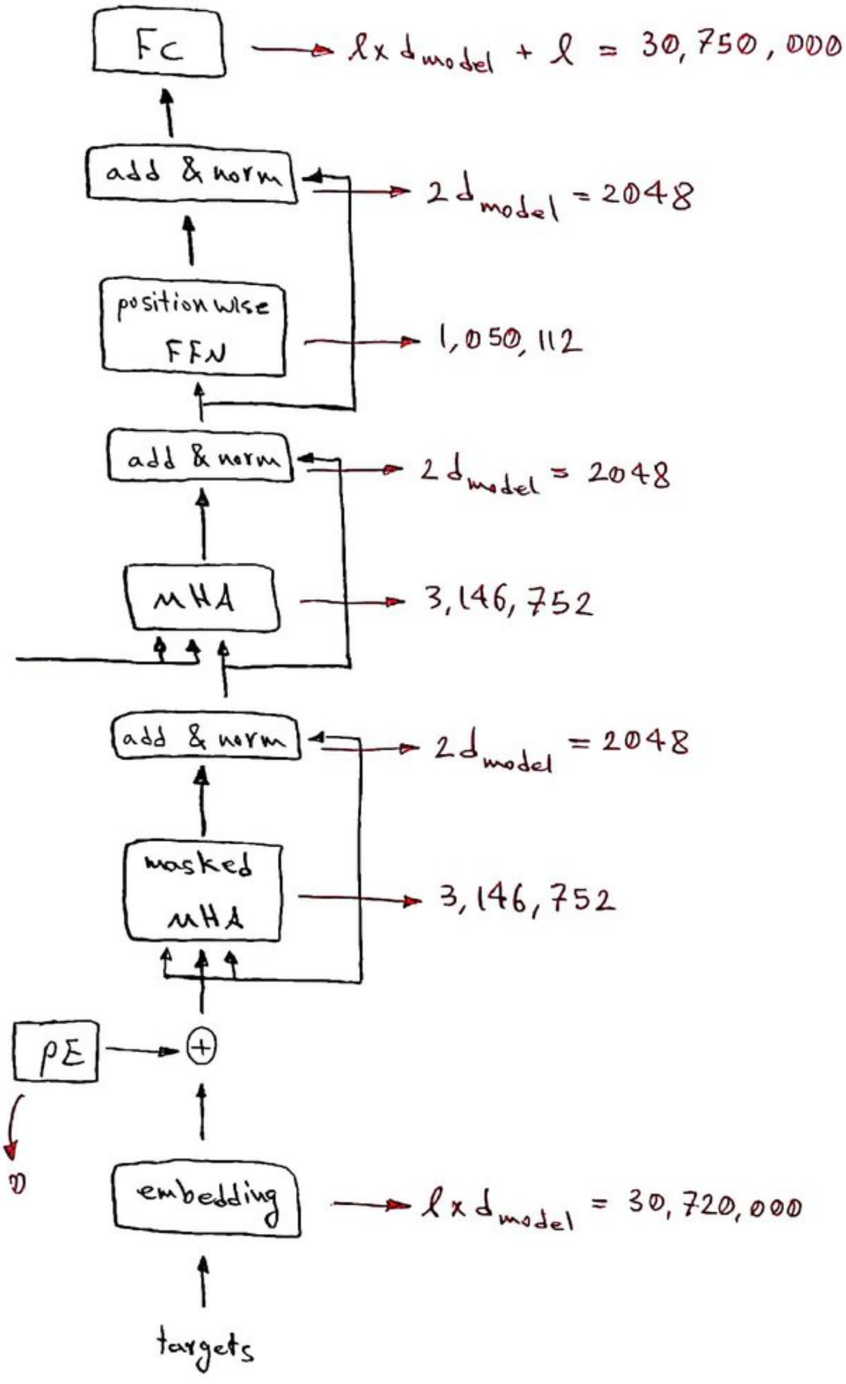
$$= 3,146,752$$

encoder params $= 30,720,000 + 12\,(4,200,960) = 81,131,520$

$\boxed{2.2}$ ادامه

مشابه بخش encoder داریم :

decoder



$$\text{decoder params} = 30,720,000 + 30,750,000 + 8\,(7,349,760) = 120,268,080$$

$$\text{encoder params} + \text{decoder params} = 201,399,600$$

$\boxed{3.1}$

اگر تمام بردار های ورودی برهم عمود باشند داریم:

$$i \neq j \quad a_{ij} = \frac{1}{\underbrace{1+\dots+1}_{N-1} + \|x_i\|^2} \simeq 0$$

$$a_{ii} = \frac{\|x_i\|^2}{1+\dots+1+\|x_i\|^2} \simeq 1$$

$$\longrightarrow A \simeq I_N \longrightarrow y_n = \sum_{m=1}^{N} a_{nm} x_m$$

$$= a_{nn} x_n \simeq x_n$$

بنابراین بردار های خروجی (y_n) تقریباً همان بردارهای ورودی (x_n) می شوند

$\boxed{3.2}$

المان های بردار a, b را با a_i ، b_i نشان می دهیم.

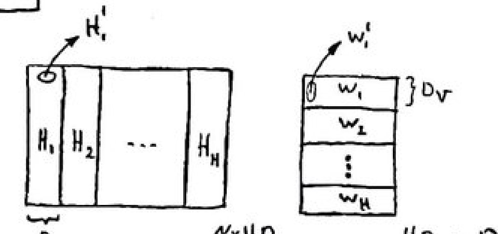
$$a_i \,,\, b_i \sim \mathcal{N}(0,1)$$

$$\mathbb{E}\left[(a^T b)^2\right] = \mathbb{E}\left[(a_1 b_1 + \dots + a_D b_D)^2\right] \overset{\mathrm{I}}{=\!=} \mathrm{Var}\left[a_1 b_1 + \dots + a_D b_D\right] \overset{\text{استقلال}}{=} \underbrace{\mathrm{Var}\left[a_1 b_1\right]}_{1\ \mathrm{II}} + \dots + \underbrace{\mathrm{Var}\left[a_D b_D\right]}_{1\ \mathrm{II}} = D$$

$$\mathbb{E}\left[a_1 b_1 + \dots + a_D b_D\right] = \sum_{i=1}^{D} \mathbb{E}[a_i b_i] \overset{a_i \perp\!\!\!\perp b_i}{=\!=\!=} \sum_{i=1}^{D} \underbrace{\mathbb{E}[a_i]}_{0} \underbrace{\mathbb{E}[b_i]}_{0} = 0 \quad \mathrm{I}$$

$$\mathrm{Var}\left[a_i b_i\right] = \mathbb{E}\left[a_i^2 b_i^2\right] - \underbrace{\mathbb{E}\left[a_i b_i\right]^2}_{0} \overset{a_i \perp\!\!\!\perp b_i}{=\!=\!=} \mathbb{E}\left[a_i^2\right]\mathbb{E}\left[b_i^2\right] = \mathrm{Var}\left[a_i\right]\mathrm{Var}\left[b_i\right] = 1 \quad \mathrm{II}$$

$\boxed{3.3}$



⊛ ماتریس $W^{(0)}$ در صورت سوال را W می نامیم تا فرمول ها خلوت تر شوند!

سطر i ام ماتریس H_j را H_j^i می نامیم که دارای بُعد $1 \times D_v$ است

ستون i ام ماتریس W_j را W_j^i می نامیم که دارای بُعد $D_v \times 1$ است

ضرب 2 ماتریس بالا به صورت بلاکی

$$= \begin{bmatrix} \sum_{i=1}^{H} H_i^1 W_i^1 & \cdots & \sum_{i=1}^{H} H_i^1 W_i^D \\ \vdots & & \vdots \\ \sum_{i=1}^{H} H_i^N W_i^1 & \cdots & \sum_{i=1}^{H} H_i^N W_i^D \end{bmatrix}_{N \times D} = \sum_{i=1}^{H} \underbrace{\begin{bmatrix} H_i^1 W_i^1 & \cdots & H_i^1 W_i^D \\ \vdots & & \vdots \\ H_i^N W_i^1 & \cdots & H_i^N W_i^D \end{bmatrix}_{N \times D}}$$

$$\underbrace{\begin{bmatrix} H_i^1 \\ \vdots \\ H_i^N \end{bmatrix}_{N \times D_v}}_{H_i} \underbrace{\begin{bmatrix} W_i^1 & \cdots & W_i^D \end{bmatrix}_{D_v \times D}}_{W_i}$$

$\boxed{3.3}$ ادامه

بنابراین نشان دادیم : $Y(X) = \sum_{i=1}^{H} H_i W_i = \sum_{i=1}^{H} \text{softmax}\left(\frac{Q_i K_i^T}{\sqrt{D_{K_i}}}\right) X \underbrace{W_i^{(V)} W_i}_{W^{(i)}}$

4.1

learnable vs fix

در حالت learnable جاسازی موقعیت دارای پارامتر هایی است که مانند سایر پارامتر های مدل learn می شوند و بنابراین این روش انعطاف بیشتری در برابر داده های مختلف و شرایط مختلف دارد اما باید حواسمان باشد که پارامتر بیشتر تا حدی به معنای d_{VC} بیشتر است و بنابراین باید مقدار داده ها نیز به اندازه کافی باشد وگرنه گرفتار overfitting می شویم.

در حالت fix جاسازی موقعیت فاقد پارامتر بوده و یک تابع fix است که از روی فرمولی از پیش تعریف شده، خروجی تولید می کند. به عنوان مثال در معماری ترنسفورمر از جاسازی موقعیت ثابت استفاده شده است. این روش انعطاف کمتری نسبت به شرایط و مسائل مختلف دارد زیرا learn نمی شود اما چون ثابت و از پیش تعریف شده است، مدل را ساده تر می کند و سریع تر است. البته باید در نظر داشت که توابع ثابت نیز می توانند inductive bias های مناسبی (مطابق کاربرد) در مدل تزریق کنند که کمک کننده باشند

absolute vs relative

در حالت absolute برداری که برای جاسازی موقعیت به دست می آید تنها بر اساس موقعیت مکانی یا همان ایندکس توکن در sequence است. با اینکه به نظر می رسد همین موقعیت مکانی هر توکن به تنهایی کافی باشد اما مشاهده می شود که در این روش موقعیت نسبی توکن ها نسبت به هم، آن طور که باید، capture نمی شود. به عنوان مثال انتظار داریم که همواره فاصله positional emb. دو توکن که از هم دور تر هستند بیشتر از فاصله positional emb. دو توکن که به هم نزدیک هستند باشد. اما این حالت همیشه رخ نمی دهد (به خصوص در حالت learnable که inductive bias ای برای این قضیه نداریم)

در حالت relative به نسبت بین توکن ها بیشتر پرداخته می شود تا مشکل ذکر شده حل شود. به عنوان مثال اگر یک sequence چهار کلمه ای داشته باشیم و یک کلمه به آن اضافه کنیم، positional emb. هر 4 کلمه قبلی عوض می شود. در این روش چون تمام pair های sequence نسبت به هم سنجیده می شوند محاسبات سنگین تری داریم. هم در زمان آموزش و هم در زمان تست فرایند کند تری خواهیم داشت.

در پیاده سازی حالت relative نیز برخلاف absolute که با embedding اولیه جمع می شود، در attention score ها اعمال می شود.

4.2

در روش RoPE سعی شده است مزیت های هردو روش absolute و relative را ترکیب کنیم. هم از ایندکس توکن ها در sequence تاثیر بپذیریم و هم موقعیت ها به صورت نسبی capture شوند و هم محاسبات efficient ای داشته باشیم!

اعمال این positional emb. از طریق ضرب یک ماتریس rotation در embedding ها صورت می‌پذیرد که زاویه چرخش $m\theta_i$ است. m بخشی است که مشابه حالت absolute است و همان ایندکس توکن ها (صرف نظر از نسبت ها) است.

از طرفی چون کلمات پایانی نسبت به کلمات ابتدایی m بزرگتری دارند، زاویه چرخش بیشتری هم دارند و بنابراین نسبت ها نیز تاثیرگذار خواهند بود.

همچنین محاسبات نیز دارای پیچیدگی کمتری است و به خصوص در زمان تست سرباری نداریم. یعنی هر 3 مشکل ذکر شده تا حد مناسبی بهبود یافتند.

4.3

اندازه بردار معادل embedding اولیه است که هنوز positional emb. به آن اضافه نشده است

زاویه بردار معادل embedding ای است که positional emb. به آن اضافه شده است

بنابراین روش RoPE به جای جمع کردن بردار positional emb. با امبدینگ قبلی، اطلاعات را در دو بعد اندازه و زاویه باهم ترکیب می‌کند به صورتی که ماتریس rotation اندازه را تغییری نمی‌دهد

4.4

روش سینوسی fix است و پارامتری ندارد اما روش RoPE یک روش learnable است.

به طور کلی طبق مزیت های ذکر شده روش RoPE اطلاعات relative و absolute موقعیت های مکانی را بهتر capture می‌کند و به دلیل learnable بودن انعطاف پذیری بیشتری هم دارد. همچنین روش پیاده سازی این دو روش نیز فرق دارد. در حالت سینوسی اطلاعات positional emb. با امبدینگ اولیه جمع می‌شود اما در روش RoPE ماتریس های rotation در Q, K ضرب می‌شوند.

برای حالت سینوسی داریم:

$$PE(pos, 2i) = \sin\left(pos / 10000^{\frac{2i}{d_{model}}}\right)$$

$$PE(pos, 2i+1) = \cos\left(pos / 10000^{\frac{2i}{d_{model}}}\right)$$

برای مثال d_{model} را برابر 4 می‌گیریم:

$$\underbrace{\begin{bmatrix} \sin 0 \\ \cos 0 \\ \sin 0 \\ \cos 0 \end{bmatrix}}_{\text{"دانشگاه"}} \quad \underbrace{\begin{bmatrix} \sin 1 \\ \cos 1 \\ \sin \frac{1}{10000^{\frac{2}{4}}} \\ \cos \frac{1}{10000^{\frac{2}{4}}} \end{bmatrix}}_{\text{"صنعتی"}} \quad \underbrace{\begin{bmatrix} \sin 2 \\ \cos 2 \\ \sin \frac{2}{10000^{\frac{2}{4}}} \\ \cos \frac{2}{10000^{\frac{2}{4}}} \end{bmatrix}}_{\text{"شریف"}}$$

⊛ امبدینگ های به دست آمده با امبدینگ های اولیه جمع می‌شوند به صورت element wise

$\boxed{4.4}$ ادامه

برای حالت RoPE داریم: (فرض می‌کنیم $d=4$)

$$\theta_i = 10000^{-\frac{2(i-1)}{d}}$$

$$\rightarrow \theta_1 = 1$$

$$\theta_2 = 10000^{-\frac{2}{4}} = \frac{1}{100}$$

⊗ ماتریس های به دست آمده از سمت راست
در ماتریس های Q , K ضرب ماتریسی می‌شوند

$$\begin{bmatrix} \cos 0 & -\sin 0 & 0 & 0 \\ \sin 0 & \cos 0 & 0 & 0 \\ 0 & 0 & \cos 0 & -\sin 0 \\ 0 & 0 & \sin 0 & \cos 0 \end{bmatrix} = I$$

"دانشگاه"

$$\begin{bmatrix} \cos 1 & -\sin 1 & 0 & 0 \\ \sin 1 & \cos 1 & 0 & 0 \\ 0 & 0 & \cos \frac{1}{100} & -\sin \frac{1}{100} \\ 0 & 0 & \sin \frac{1}{100} & \cos \frac{1}{100} \end{bmatrix}$$

"صنعتی"

$$\begin{bmatrix} \cos 2 & -\sin 2 & 0 & 0 \\ \sin 2 & \cos 2 & 0 & 0 \\ 0 & 0 & \cos \frac{2}{100} & -\sin \frac{2}{100} \\ 0 & 0 & \sin \frac{2}{100} & \cos \frac{2}{100} \end{bmatrix}$$

"شریف"

4.5

در نرم ماتریسی برای حالت auto regressive داریم :

$$\text{softmax}\left(\begin{bmatrix} q_1 \cdot k_1 & & & \\ q_2 \cdot k_1 & q_2 \cdot k_2 & & \\ & & & \\ q_N \cdot k_1 & q_N \cdot k_2 & \cdots & q_N \cdot k_N \end{bmatrix}_{N \times N} + m \begin{bmatrix} 0 & & & \\ -1 & 0 & & \\ \vdots & & & \\ -(N-1) & -(N-2) & \cdots & 0 \end{bmatrix}_{N \times N} \right)$$

روش عملکرد ALiBi به این صورت است که یک ماتریس شبیه بایاس به QK^T اضافه می کند که بر اساس فاصله هر جفت توکن از یکدیگر به شباهت آن ها پنالتی می دهد. یعنی اگر 2 کلمه شباهت زیادی داشته باشند، ماتریس اول مقدار ضرب داخلی بیشتری خواهد داشت اما اگر آن دو کلمه از هم دور باشند بخش بایاس به اندازه فاصله آن دو کلمه از مقدار شباهت آن ها کم می کند که باعث می شود attention score کمتری نسبت به هم داشته باشند.

به این صورت توسط یک روش غیر learnable توانستیم نسبیت بین توکن ها را نیز در معادله دخیل کنیم.

توسط m می توان میزان تأثیر نسبیت را کنترل کرد. هر چه m بیشتر باشد، نزدیکتر بودن کلمات به هم مهم تر از شباهت امبدینگ آنها می شود.

⊛ m به ازای هر head جدا تعیین می شود که قابلیت کنترل بیشتری می دهد

5.1

طبق توضیحات اضافه در کوئرا ، X را به صورت یک بردار $ND\times 1$ در نظر می‌گیریم و برای عبارت داخل softmax داریم:

$$\frac{QK^T}{\sqrt{D}} = \frac{1}{\sqrt{D}} \times \underset{ND\times ND}{[W_Q]} @ \underset{ND\times 1}{[X]} @ \underset{1\times ND}{[X^T]} @ \underset{ND\times ND}{[W_K^T]}$$

حال XX^T را ورودی در نظر می‌گیریم. باتوجه به اینکه ماتریس W_K^T مربعی است می‌دانیم که می‌توان به ازای آن، ماتریس $\underset{ND\times ND}{[W'_K]}$ را یافت که ضرب کردن W'_K از چپ در عبارت فوق معادل ضرب کردن W_K^T از راست در آن عبارت شود، یعنی:

$$\frac{QK^T}{\sqrt{D}} = \underbrace{\frac{1}{\sqrt{D}} W'_K W_Q}_{W} \underbrace{XX^T}_{\text{ورودی}}$$

بنابراین نشان دادیم که عبارت داخل softmax معادل یک ضرب ماتریسی است.

همچنین softmax را نیز می‌توان معادل activation function ای در نظر گرفت که روی این لایه خطی اعمال می‌شود.

بعد نهایی خروجی نیز معادل بعد ورودی است زیرا:

$$Y = \text{softmax}\left(\underset{ND\times ND}{[W]} \underset{ND\times ND}{[XX^T]}\right) @ \underset{ND\times 1}{[V]}$$

پس Y نیز مانند X دارای بعد $ND\times 1$ است.

5.2

طبق بخش قبل ماتریس W دارای بعد $ND\times ND$ است و بنابراین تعداد پارامترها $O(N^2D^2)$ می‌شود

5.4

تفاوت multi head attention با حالتی که یک head داشته باشیم در طول امبدینگ کلمات (در حالت میانی) است و بنابراین در این سوال که با تغییر ترتیب کلمات کار دارد، تفاوتی ایجاد نمی‌کند. در حالت کلی طبق فرمول attention داریم:

$$QK^T = \underset{N\times D_K}{\begin{bmatrix} x_1^T W_Q \\ \vdots \\ x_N^T W_Q \end{bmatrix}} @ \underset{D_K\times N}{\begin{bmatrix} | & & | \\ W_K^T x_1 & \cdots & W_K^T x_N \\ | & & | \end{bmatrix}}$$

واضح است که ضرب ماتریسی بالا نسبت به جابجا کردن x_i ها (ترتیب کلمات) equivarient است زیرا شباهت هر کلمه نسبت به تمام کلمات سنجیده می‌شود و به همان ترتیب کلمات ورودی در ماتریس $N\times N$ حاصل شده قرار می‌گیرد. حال اگر ترتیب ورودی تغییر کند، طبق ترتیب جدید، تمام شباهت‌ها به دست آمده و در ماتریس $N\times N$ حاصل شده قرار می‌گیرد.

اعمال softmax که اثری در این قضیه ندارد و ضرب در ماتریس $\underset{N\times D_K}{\begin{bmatrix} x_1^T W_V \\ \vdots \\ x_N^T W_V \end{bmatrix}}$ نیز دقیقا طبق استدلال ذکر شده نسبت به جابجا کردن x_i ها equivarient است.

5.4 ادامه

بنابراین همان طور که ذکر شد multi head کردن و سپس concat کردن در بُعد طول امبدینگ هر کلمه است نه ترتیب کلمات و هیچ اثری در این جابجایی ندارد و تمام ضرب های ماتریسی صورت گرفته نیز نسبت به جابجایی ورودی ها equivarient هستند. بنابراین هر ترتیبی که در X داشته باشیم، همان ترتیب سطر ها را نیز در Y داریم و ترتیب توکن های sequence عملاً لحاظ نمی‌شود.

برای همین است که از positional encoding استفاده می‌شود.

⊛ در برخی سوال ها با علی بابایک همفکری داشتم