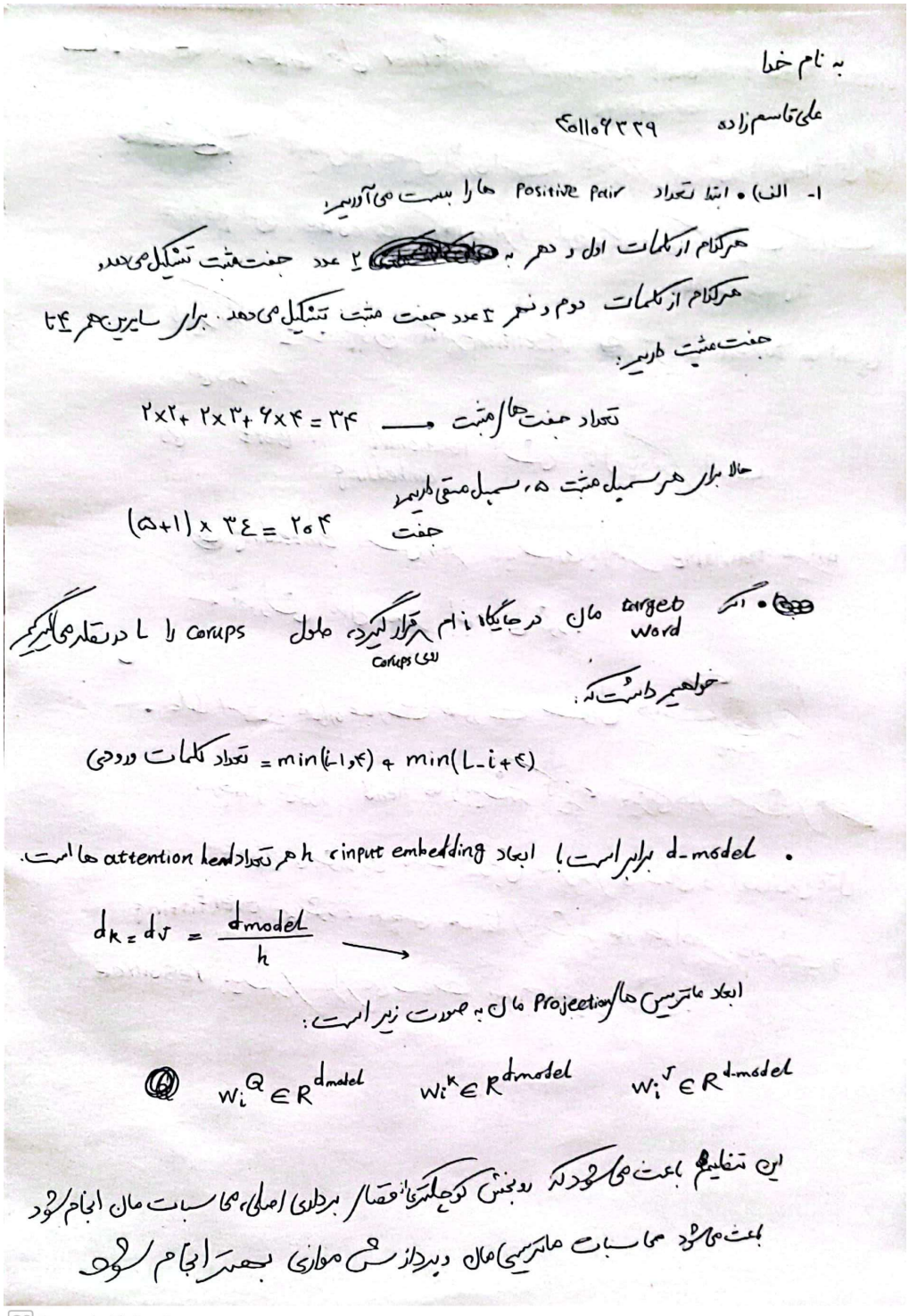
به نام خدا

علی قاسم زاده 401106339



به نام خدا

علی قاسم زاده ۴۰۱۱۰۶۳۳۹

۱- الف) • ابتدا تعداد Positive Pair ها را بست می آوریم:

هرکلام از کلمات اول و دهم به ۲ عدد جفت مثبت تشکیل می دهد.

هرکلام از کلمات دوم و نهم ۳ عدد جفت مثبت تشکیل می دهد. برای سایرین هم ۴ تا جفت مثبت داریم:

تعداد جفت های مثبت ← $۲\times۲+۲\times۳+۶\times۴=۳۴$

حالا برای هر سمپل مثبت ۵ سمپل منفی داریم

جفت $(۵+۱)\times۳۴=۲۰۴$

• اگر target word مان در جایگاه i ام قرار گیرد، طول Corups را L در نظر می گیریم

طول Corups ای که خواهیم داشت که:

تعداد کلمات ورودی = $\min(i-1,۴)+\min(L-i,۴)$

• d_model برابر است با ابعاد input embedding و h تعداد attention head ها است.

$$d_k=d_v=\frac{d_{model}}{h}$$

ابعاد ماتریس های Projection مان به صورت زیر است:

$$w_i^Q\in R^{d_{model}} \quad w_i^K\in R^{d_{model}} \quad w_i^V\in R^{d_{model}}$$

این تفاوت باعث می شود که روبنش کوچکتر از قضا بر طبق اصلی، محاسبات مان انجام شود

باعث می شود محاسبات ماتریسی مان دیدار شش موازی بهتر انجام شود

ب). درست : در معماری اصلی transformer از fixed sinusoidal positional encodings استفاده شده است.

از پیش هر تعیین شده است و یادگیری نمی شود (train نمی شود)

در واقع این encode کردن برای این به کار می رود که ترتیب کلمات اطلاعات حاصل از را نیز نگه داریم. چون مکانیزم attention به ترتیب token ها اهمیتی نمی دهد.

ولی Bert ، positional embedding آن قابل یادگیری است.

embedding آن را هم با یه سری پارامتر trainable میزاره ← آموزش مدل بهتر انجام می شود.

• غلط است، همواره عبارت صورت سوال درست نیست زیرا :

هرچقدر تعداد head ها را بیشتر کنیم، ویژگی ها بیشتر استخراج می شوند ولی باعث افزایش هزینه های محاسباتی و احتمالی مثل overfitting می شود، همچنین ما قطعا بیشتر نیاز خواهیم داشت و resource بیشتری برای محاسبات نیاز خواهیم داشت.

۴- الف)

$$-\frac{\partial \log P(w_o|w_t)}{\partial v_{w_o}} = -\frac{\partial \log P(w_o|w_t)}{\partial P(w_o|w_t)} \cdot \frac{\partial P(w_o|w_t)}{\partial v_{w_o}} =$$

$$-\frac{1}{P(w_o|w_t)} \cdot \frac{\partial \left(\frac{\exp(u_{w_t}^T v_{w_o})}{\sum_{k \in V} \exp(u_{w_t}^T v_k)} \right)}{\partial v_{w_o}} = \quad \rightarrow A$$

$$-\frac{1}{P(w_o|w_t)} \cdot \frac{u_{w_t} \exp(u_{w_t}^T v_{w_o}) A - u_{w_t} \exp(u_{w_t}^T v_{w_o})^2}{A^2}$$

$P(w_o|w_t)$ $\qquad$ $P(w_o|w_t)^2$

$$= -\frac{1}{P(w_o|w_t)} \cdot u_{w_t} \left(P(w_o|w_t) - P(w_o|w_t)^2 \right)$$

$$= u_{w_t} \left(P(w_o|w_t) - 1 \right)$$

ب)

c=1 : مدلسول متعاقبه همسایه‌ها خیلی تردیکش وابسته هستند ← این ساختار برای داده ساختار هایی که داده‌های محلی دارند خوب عمل می‌کند، ارتباط‌های طولانی مدت را هم نادیده می‌گیرد. می‌سبات هم کمتر و آسان‌تر خواهد بود چرا که تعداد neg pair , pos pair کمتر خواهند بود.

c=5 : مدلسول به یه تعداد همسایه تردیک بستگی دارد و مدلسول skip-gram می‌تونه ارتباطات بین کلمات اطرافی را خوب یاد بگیرد، معمولاً هم برای داده‌های عمومی خوب عمل می‌کند. هزینه محاسباتیش هم بین اون دو مدل است می‌تونیم متوسط در نظر بگیریم

c=10 : مدلسول به تعداد همسایه‌های زیادی و روابط طولانی بین کلمات توجه می‌کند و ارتباط‌های طولانی‌تر را در نظر می‌گیرد، از طرفی زمان زیادی هم طول می‌کشد تا مدل ترین شود و تعداد sample بیشتر برای یادگیری نیاز خواهیم داشت.

3. 1. اگر دنباله ی ورودی بردارهای کلمه به شکل X = {x_1, x_2, …., x_n} باشد آنگاه داریم :

Attention(Q, K, V) = softmax(QK^T/sqrt(d_k))V

حال میخواهیم عملیات را به صورت یک لایه ی fc در نظر بگیریم :

بردار های x_i را در کنار هم قرار می دهیم تا ماتریس Y را بسازیم، داریم که :

$$Q=YW_Q, \quad K=YW_K, \quad V=YW_V$$

ماتریس A را به این صورت می سازیم که :A = YY^T

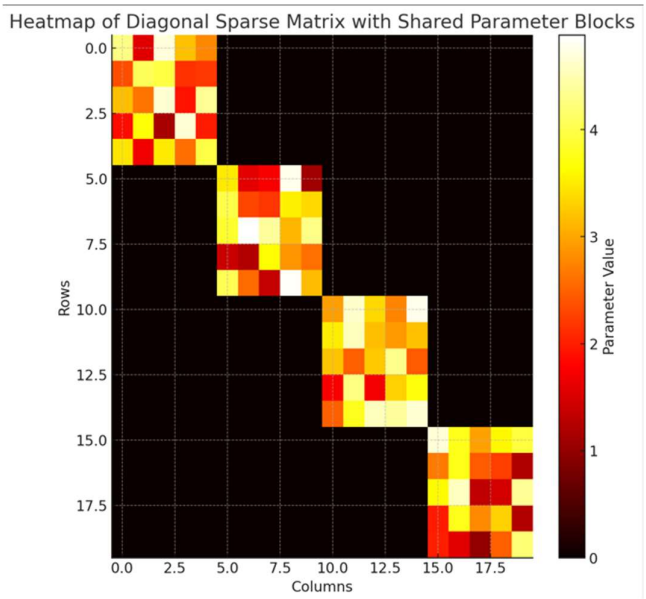
این ضرب هم روابط بین بردار های ورودی را نشان می دهد. حالا برای مپ کردن ورودی به یک بردار خروجی، یک ضرب بین این ماتریس و ماتریس وزن های W انجام می دهیم که W قابل یادگیری است.

Softmax(AW)

این عبارت هم معادل یک FC با تابع activation = softmax است.

2.3. در شبکه قبلی داریم که W ماتریسی N*N است و در نتیجه O(N^2) تا پارامتر دارد. حالا می خواهیم N*N را به D*D ببریم پس به O(N^2D^2) پارامتر نیاز داریم.

3.3. در مکانیزم attention به هرکلمه در دنباله ی ورودی توجه می شود ولی میزان توجه مان به یک اندازه نیست و وزن های متفاوتی داریم همچنین چون از softmax استفاده می کنیم توزیع توجه مان پراکنده خواهد شد و تعداد زیادی از وزن ها نزدیک صفر خواهند ماند در نتیجه ماتریسی sparse ایجاد می شود که به کلمات مهمتر بیشتر وزن برای توجه می دهد.



4. مکانیزم attention نسبت به ترتیب ورودی ها اهمیتی نمی دهد، یعنی اگر جابه جا کنیم ترتیب ورودی ها را باز هم همان خروجی را می دهد برای همین نیاز داریم تا اطلاعات مکانی کلمات را در جایی ذخیره کنیم برای همین از positional encoding استفاده می کنیم.

اکر ماتریس X را با ابعاد N*D در نظر بگیریم، حالا جایگشت p را در نظر می گیریم که ترتیب سطر ها را عوض می کند. خواهیم داشت که X_p(i)

حالا داریم که :

Q' = YW_Q, K' = YW_K, V' = YW_V

Q_p(i) = X_p(i)W_Q, K = X_p(i)W_K, V = X_p(i)W_V

حالا باید داشته باشیم که :

(QK^T)_ij = (Q'K'^T)_p(i)p(j)

حالا softmax را اعمال می کنیم، خواهیم داشت که :

Softmax(Q'K'^T/sqrt(d_k))_p(i)p(j) = Spftmax(QK^T/sqrt(d_k))_ij

حالا وزن های بدست آمده برای هر کلمه در این softmax دقیقا متناظر با همان وزن هایی هستند که بدون جایگشت داشتند. در نتیجه اگر جایگشت بدهیم تغییر در وزن خروجی i ام ایجاد نمی شود

O_i = O_p(i)

در نتیجه حکم اثبات می شود .

۴- ①

encoder Inp : [۳۲, ۲۰۴۸, ۱۰۲۴]

per-head self att : [۳۲, ۲۰۴۸, ۱۹۲]

all heads : [۳۲, ۲۰۴۸, ۱۰۲۴]

Forward 1 : [۳۲, ۲۰۴۸, ۵۱۲]

Forward 2 : [۳۲, ۲۰۴۸, ۱۰۲۴]

Encoder out : [۳۲, ۲۰۴۸, ۱۰۲۴] → encoder

decoder inp : [۳۲, ۲۰۴۸, ۱۰۲۴]

Per-head self att : [۳۲, ۲۰۴۸, ۱۹۲]

all heads : [۳۲, ۲۰۴۸, ۱۰۲۴]

Per-head cross-att : [۳۲, ۲۰۴۸, ۱۹۲]

all-cross-att : [۳۲, ۲۰۴۸, ۱۰۲۴]

Forward 1 : [۳۲, ۲۰۴۸, ۵۱۲]

Forward 2 : [۳۲, ۲۰۴۸, ۱۰۲۴]

out : [۳۲, ۲۰۴۸, ۳۰۰۰۰]

② → embedding vector

$d_k = d_v = \frac{۷۶۸}{۴} = ۱۹۲$

- تعداد پارامترهای self_attention

$*param(W_Q) = *param(W_K) = *param(W_V) = ۴ \times ۱۰۲۴ \times ۱۹۲ = ۷۸۶۴۳۲$

$*param(W_O) = ۷۶۸ \times ۱۰۲۴ = ۷۸۶۴۳۲$

- تعداد پارامترهای Multi-head attention

$M \rightarrow$ self $= ۴ \times (۴ \times ۱۰۲۴ \times ۱۹۲) + (W_O)\ ۷۶۸ \times ۱۰۲۴ = ۳۱۴۵۷۲۸$

- تعداد پارامترهای Feed Forward

$F_1 \rightarrow *param(W_1) = ۱۰۲۴ \times ۵۱۲ + ۵۱۲ = ۵۲۴۸۰۰$

$F_2 \rightarrow *param(W_2) = ۱۰۲۴ \times ۵۱۲ + ۱۰۲۴ = ۵۲۴۸۳۲$

- تعداد پارامترهای هر بلوک

$B = M + F_1 + F_2 = ۴۱۹۵۸۴۰$

- تعداد پارامترهای encoder

$E = ۸ \times B = ۳۳۵۶۶۷۲۰$

حالا تعداد پارامترهای decoder را حساب می‌کنیم:

دیکودر ما یک لایه cross attn نیز دارد که تعداد پارامترهای آن برابر است با تعداد پارامترهای self-att

$12 \times (4195\,8680 + 3145\,728) = 88\,098\,816$

تعداد پارامترهای embedding برای لایه آخر:

$1024 \times 30000 + 30000 = 30\,750\,000$

تعداد کل پارامترها عبارت است از:

$33\,544\,720 + 88\,098\,816 + 30\,750\,000 = 152\,394\,536$

3. "جمله علی به مدرسه می‌رود" را در نظر بگیرید، خواهیم دانست که:

ابتدا هر کلمه در این جمله به بردارهای با سایز 1024 مپ می‌شود که اطلاعات آن مثل اطلاعات معنایی، دستوری و ... را encode می‌کند.

در واقع جمله ما به تعداد token تقسیم می‌شود و سپس هر توکن به یک بردار 1024 تایی مپ می‌شود

در مرحله tokenize کردن، اطلاعات Positional نیز به بردارها اضافه می‌شوند تا اطلاعات ترتیب کلمات حفظ شود

در مرحله encoder،

برای هر توکن ورودی ماتریس های

K ⟶ key
V ⟶ value
Q ⟶ query

محاسبه می‌شوند.

$$Attention(Q,K,V) = Softmax\left(\frac{QK^T}{\sqrt{d_K}}\right)V$$

attention از رابطه‌ی بالا استفاده می‌شود. برای هر Head مقدار Attention محاسبه می‌شود

و در نهایت تمامی مقادیر ادغام می‌شوند و در ماتریس W_o ضرب می‌شوند

$W_o \in R^{V4A \times 1024}$.

نتایج همراز دولایه‌ی Forward عبور می‌کنند (با تابع فعالسازی σ)

$$FFN(x) = \sigma\left(\sigma(XW_1+b_1)W_2+b_2\right)$$

[redacted] Forward خروجی encoder تایل خروجی دومین

سپس دیکودر self attention را مشابه Encoder برای توکن های هدف حساب

می‌کند سپس برای ارتباط بین ورودی دیکودر [redacted] و خروجی انکودر از

cross-attentic استفاده می‌شود

$$Att(Q,K,V) = \left(\frac{QK^T}{\sqrt{d_K}}\right)V.$$

نتایج در نهایت از دولایه Forward عبور می‌کند و $\hat{y}$ = بردار خروجی محاسبه می‌کنیم.

سپس با استفاده از تابع Softmax احتمال‌ها روی کلمات حساب می‌شود

در نهایت کلمات به این صورت پیش بینی می‌شوند که:

کلمه اول پیش بینی شده سپس کلمه بعدی به شرط آن

و الی آخر. (هر کس به شرط تمام قبلی‌هایش)

اینقدر کلمه تولید می‌شود تا به پایان جمله یا حداکثر طول کلمات برسیم.

5. الف) مدل bert از NSP در زمان آموزش خود استفاده می کند، این کمک می کند تا ارتباط بین جملات متوالی را بتواند بهتر درک کند به صورت عمل می کند که با تعدادی pair sentence مدلمان train می شود و بررسی میکند که جمله ی دوم چقدر به اولی وابسته است یا نه این کمک می کند تا ارتباط بین جملات و و ترتیب آنها را بیاموزد.

برای تسک های پایین دستی که نیاز به درک معنای کلی از متن دارند نیازی نداریم که ارتباط بین جملات را بررسی کنیم و با حذف nsp بیشتر می توانیم روی درک مفهوم کلی جمله تمرکز کنیم و در نتیجه عملکردمان بهبود می یابد.

ب) **تنظیم دقیق (Fine-tuning) مدل‌هایی مانند BERT** برای تسک‌های پایین‌دستی (downstream tasks)، زمانی که نیاز به درک کلی جمله و معنای آن در سطحی وسیع‌تر داریم، چالش‌های خاص خود را به همراه دارد. این مسئله به ویژه در شرایطی مطرح می‌شود که مدل فقط برای تسک Masked Language Modeling (MLM)آموزش دیده باشد و فاقد قابلیت‌های پیش‌فرض برای درک روابط کلی جمله باشد (مانند مدل RoBERTa که NSP را حذف کرده است.)

استفاده از لایه‌های Pooling برای استخراج نمای کلی جمله:

- **ایده اصلی :**در این روش، به جای استفاده از بردارهای مرتبط با هر توکن به صورت جداگانه، یک نماینده ثابت (بردار معنادار) برای کل جمله استخراج می‌شود. این نمایه می‌تواند خلاصه‌ای از اطلاعات موجود در جمله باشد و نمایانگر معنای کلی آن باشد.
- **روش‌های رایج در لایه‌های:Pooling**

1. **:Mean Pooling**در این روش، میانگین تمام بردارهای خروجی کلمات جمله گرفته می‌شود. این میانگین می‌تواند به عنوان نمای کلی جمله استفاده شود.
2. **:Max Pooling**در این روش، بیشترین مقدار در هر بُعد از بردارهای خروجی کلمات انتخاب می‌شود. این روش ویژگی‌های غالب جمله را برجسته می‌کند.
3. **:CLS Token**در مدلBERT ، توکن [CLS] (که در ابتدای جمله اضافه می‌شود) به عنوان نماینده جمله استفاده می‌شود. بردار خروجی مرتبط با این توکن، خلاصه‌ای از کل جمله است و می‌تواند برای بسیاری از تسک‌ها به کار گرفته شود.

روش‌های بهینه‌سازی Pooling با استفاده از:Sentence-BERT

Sentence-BERTیکی از مدل‌هایی است که برای استخراج نمای کلی جملات بهینه‌سازی شده است. این مدل از معماری BERT پایه استفاده کرده، اما تغییراتی در فرآیند تنظیم دقیق اعمال کرده است:

1. **هدف :Sentence-BERT** تبدیل هر جمله به یک بردار معنادار در فضای برداری، به گونه‌ای که جملات با معنای مشابه، به بردارهای نزدیک به هم تبدیل شوند.

2. **کاربرد :Pooling** در این مدل، از روش‌های مختلف Pooling مانند mean pooling یا max poolingاستفاده می‌شود تا جمله به یک نمایه برداری ثابت تبدیل شود.
3. **مزایا :**استفاده از این تکنیک باعث می‌شود مدل بتواند به طور مستقیم در تسک‌هایی مانند مقایسه معنایی جملات، طبقه‌بندی متون و بازیابی اطلاعات استفاده شود.