به نام خدا

# تمرین سری پنجم درس یادگیری ماشین دکتر علی شریفی زارچی

# فرزان رحمانی 403210725

## سوال اول

الف)

* تعداد کل = 204 = 170 + 34  
  در یک مدل Skip-gram، هر کلمه به عنوان کلمه مرکزی (center word) عمل می کند و مدل تلاش می کند تا کلمات بافت (context words) اطراف را در اندازه پنجره مشخص شده پیش بینی کند. با اندازه پنجره 2، هر کلمه می تواند تا 2 کلمه متنی در هر طرف داشته باشد، که منجر به حداکثر 4 کلمه متنی در هر کلمه مرکزی می شود.

برای جمله ای با 10 کلمه، تعداد جفت های مثبت به موقعیت کلمه در جمله بستگی دارد:

* + کلمات در ابتدا و انتهای جمله: کلمات اول و آخر context words کمتری دارند. به عنوان مثال، کلمه اول فقط دو کلمه بعدی را به عنوان متن دارد و کلمه آخر فقط دو کلمه قبلی را به عنوان متن دارد.
  + کلمات وسط جمله: این کلمات دارای پنجره کامل کلمات زمینه (full window of context words) هستند.

برای محاسبه تعداد کل جفت های مثبت داریم:

* + کلمه اول: 2 کلمه زمینه (context word)
  + کلمه دوم: 3 کلمه زمینه
  + کلمات سوم تا هشتم (6 کلمه): هر کدام 4 کلمه زمینه
  + کلمه نهم: 3 کلمه زمینه
  + کلمه دهم: 2 کلمه زمینه

Total positive pairs = 2 + 3 + (6 × 4) + 3 + 2 = 2 + 3 + 24 + 3 + 2 = 34

برای هر جفت مثبت (positive pair)، نمونه برداری منفی (negative sampling) شامل انتخاب تعداد معینی از نمونه های منفی (negative samples) است. در این حالت، با 5 نمونه منفی در هر جفت مثبت، تعداد کل نمونه های منفی برابر است با:

Total negative samples = 34 positive pairs × 5 negative samples per pair = 170

بنابراین تعداد کل نمونه های آموزشی اعم از مثبت و منفی عبارتند از:

Total training samples = 34 positive pairs + 170 negative samples = 204

* تعداد کل ورودی ها = 8 = 4 \* 2

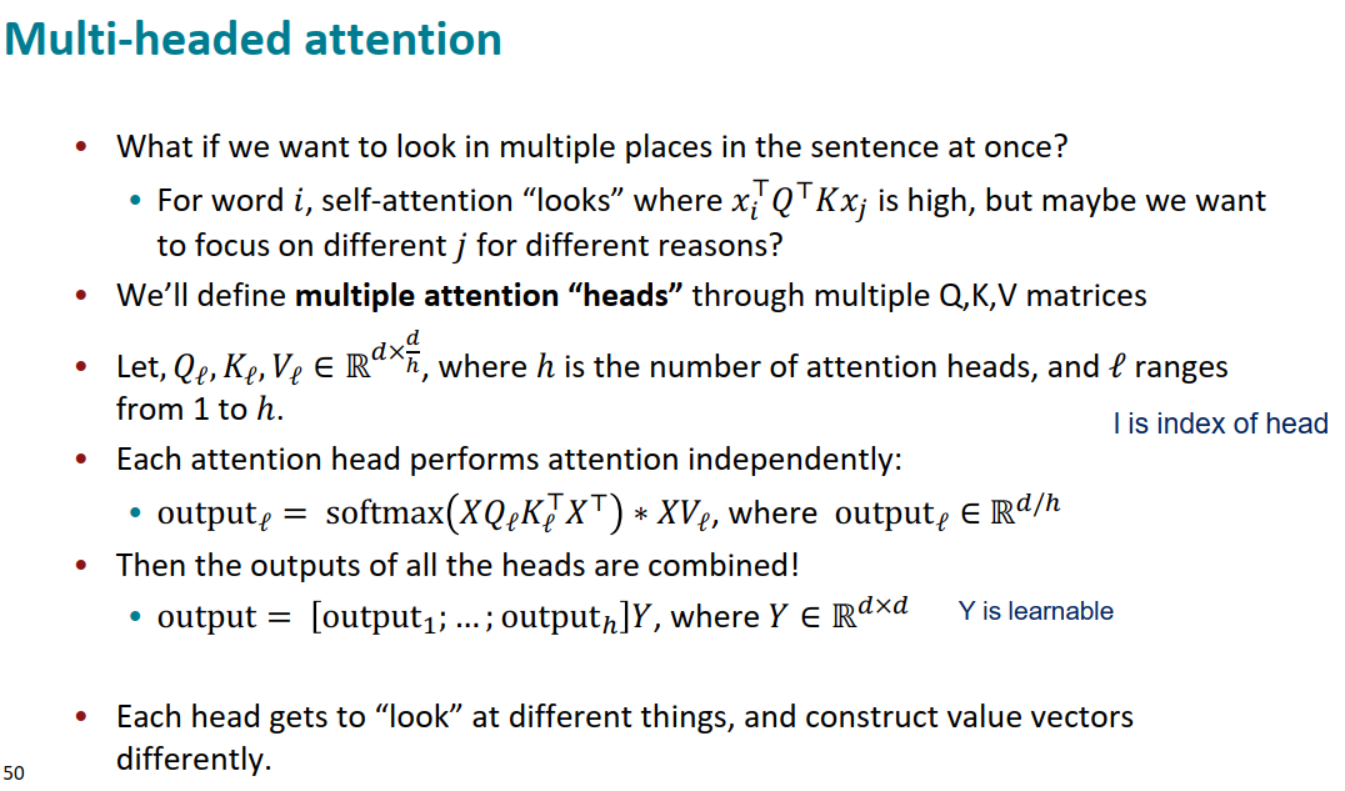
در مدل Continuous Bag of Words (CBOW)، اندازه پنجره زمینه (context window size) به تعداد کلمات در نظر گرفته شده در هر طرف کلمه مورد نظر اشاره دارد. بنابراین، اندازه پنجره 4 به معنای 4 کلمه قبل و 4 کلمه بعد از کلمه مورد نظر به عنوان context استفاده می شود.

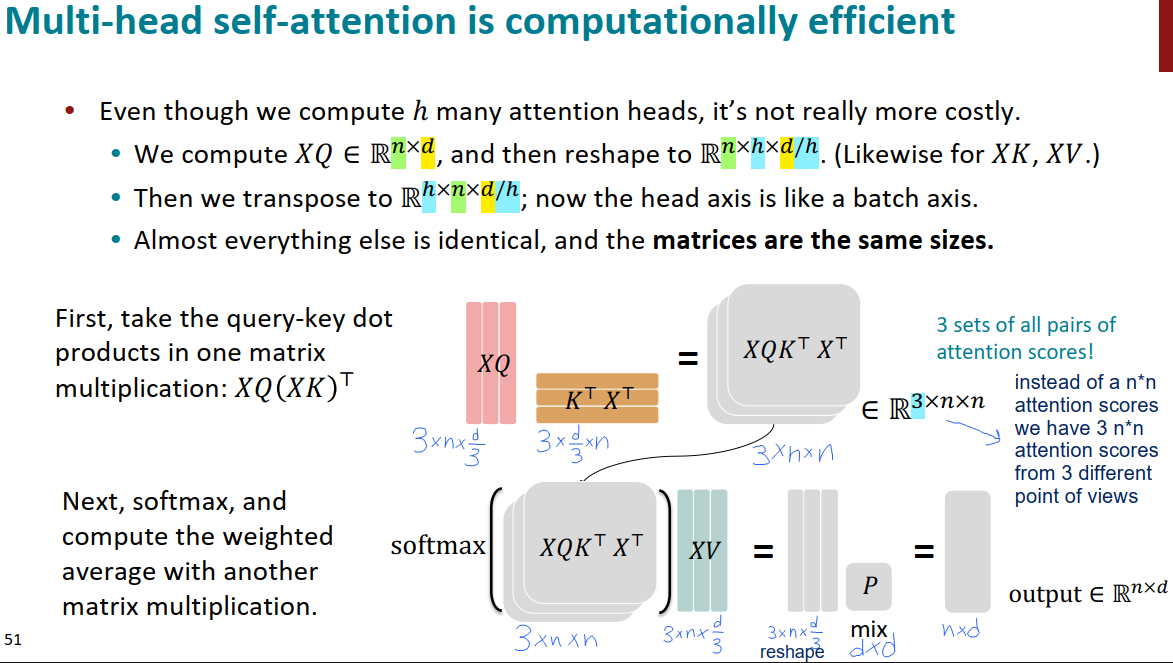
Total context words per target word = 4 (before) + 4 (after) = 8

بنابراین، 8 کلمه ورودی برای پیش بینی هر کلمه هدف (target word) در این مدل CBOW استفاده می شود.

ملاحضه دقیق تر: البته در نظر داریم که 4 کلمه قبل و بعد در صورت وجود استفاده می شوند برای مثال برای کلمات اول و آخر متن چون فقط کلمات در یک طرف موجود هستند صرفا 4 کلمه داریم یعنی برای کلمه اول متن فقط 4 کلمه بعد را داریم و برای کلمه آخر متن فقط چهار کلمه قبل را داریم. به همین دلیل به جز 4 کلمه اول متن و 4 کلمه آخر متن همگی دارای 8 input words برای پیش بینی هستند. البته می توانیم با استفاده از تکنیک padding و استفاده از توکن اختصاصی [PAD] در ابتدا و انتهای متن، 4 توکن [PAD] اضافه کنیم تا همه کلمات از جمله کلمات ابتدایی و انتهایی دقیقا دارای 8 کلمه ورودی برای پیش بینی هر کلمه هدف (target word) در این مدل CBOW باشند.

* برای توضیح این سوال ابتدا به اسلاید زیر از کورس CS224n دانشگاه Sandford توجه کنید که شکل های خوبی به همراه توضیحات کاملی را داده است. سپس توضیحات من آمده است.





در Multi-Head Attention، بعد embedding ورودی بین سرهای توجه h   
(h attention heads) تقسیم می شود. هر head در subspace با بعد عمل می‌کند. (Each head processes a smaller dimensional embedding)

ماتریس های پروجکشن برای نمایش embedding های ورودی در بردارهای query، key و value برای هر head استفاده می شوند. برای هر head، این ماتریس های projection دارای ابعاد زیر هستند:

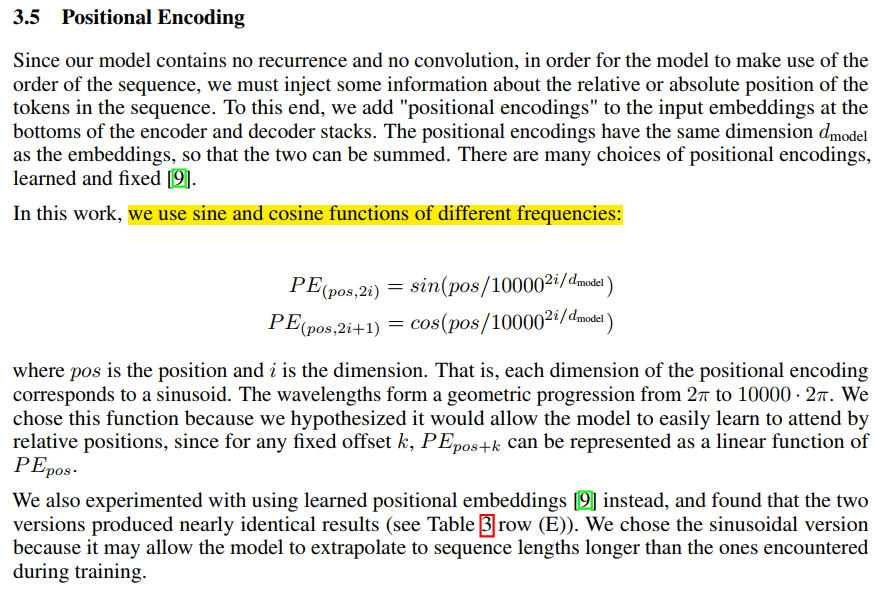
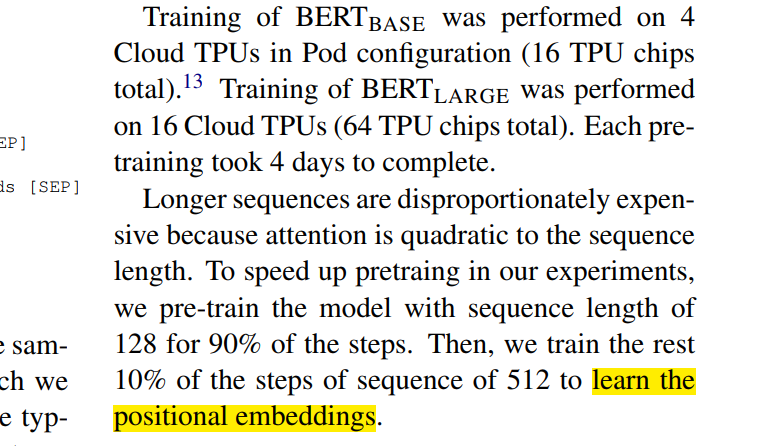
از آنجایی که ، ابعاد را می توان به صورت زیر بیان کرد:

این تنظیم کارایی محاسباتی را با اجازه دادن به هر head توجه برای تمرکز بر روی زیرفضاهای مختلف(different subspaces) embedding های ورودی غنی تر می‌کند و مدل را قادر می‌سازد تا ویژگی‌ها و روابط متنوع را ثبت کند. به عنوان مثال یک head به روابط semantic و head دیگر به روابط syntactic توجه میکنند. با پردازش موازی این زیرفضاها (processing these subspaces in parallel)، مدل می تواند محاسبات توجه را در مقایسه با پردازش کل فضای تعبیه شده به طور همزمان به صورت computationally more efficient انجام دهد.

در واقع، تقسیم به ابعاد کوچکتر امکان محاسبه موازی برای هر head توجه را فراهم می‌کند. این کار حافظه مورد نیاز و پیچیدگی محاسباتی را کاهش می دهد، به ویژه برای بزرگ.  
همچنین توجه داریم در مقایسه با توجه عادی، تقریباً همه چیزهای دیگر یکسان هستند و ماتریس ها هم اندازه هستند ولی با این حال پردازش داده ها در subspace های مختلف امکان پردازش داده از جنبه های مختلف و بهبود عملکرد را می دهد.

ب)

* این عبارت صحیح است.   
  در معماری Transformer اصلی، همانطور که توسط Vaswani و همکارانش در "توجه همه شما نیاز است" معرفی شد، رمزگذاری های موقعیتی سینوسی ثابت (fixed sinusoidal positional encodings) به جاسازی های ورودی (input embeddings) اضافه می شوند تا اطلاعاتی در مورد موقعیت کلمات در دنباله ارائه دهند. در مقابل، BERT (Bidirectional Encoder Representations from Transformers) از   
  learned positional embeddings استفاده می‌کند و به مدل اجازه می‌دهد اطلاعات موقعیتی را در طول آموزش یاد بگیرد. در تصاویر زیر قسمتی از مقالات که به آن اشاره کردند را میبینید:



* این عبارت غلط است.   
  افزودن attention heads بیشتر، عملکرد بهتر را تضمین نمی کند. در حالی که افزایش تعداد attention heads به مدل اجازه می دهد تا طیف وسیع تری از ویژگی ها و روابط را با توجه به بخش های مختلف ورودی به تصویر بکشد، عملکرد بهبود یافته را تضمین نمی کند. diminishing returns فراتر از یک نقطه مشخص وجود دارد و افزایش بیش از حد تعداد heads ها می‌تواند منجر به overfitting، افزایش پیچیدگی محاسباتی و مصرف منابع بدون افزایش عملکرد قابل‌توجه شود. بنابراین، تعداد سرهای توجه با توجه به task خاص و منابع محاسباتی موجود باید با دقت انتخاب شود.   
  همچنین میدانیم:
  + Head ها ممکن است به اطلاعات اضافی (redundant information) توجه کنند.
  + پارامتر بیش از حد (Overparameterization) ممکن است منجر به آموزش ناکارآمد یا overfitting شود.
  + با افزایش تعداد head ها هر head بخش کوچکتر از را دریافت میکند و این کاهش میتواند منجر به کاهش ظرفیت هر head برای یادگیری ویژگی های مهم و پیچیده شود. اگر دیتا به اندازه کافی متنوع نباشد افزایش تعداد head ها میتواند منجر به یادگیری ویژگی های تکراری یا غیرمفید شود.

## سوال دوم

الف) محاسبه گرادیان:

هدف ما محاسبه گرادیان با توجه به بردار "context" است. بیایید آن را تجزیه کنیم:

مرحله 1: احتمال log را بنویسیم با توجه به خاصیت تبدیل تقسیم به تفریق log:

مرحله 2: با توجه به مشتق میگیریم:

گرادیان برابر است با:

مرحله 3: مشتقات جزئی را محاسبه کنیم:

1. برای عبارت اول:

2. برای عبارت دوم داریم:

با استفاده از قانون زنجیره داریم:

این ساده می شود به:

مرحله 4: نتایج را ترکیب میکنیم:

بنابراین پاسخ نهایی به شرح زیر است:

ب) تأثیر اندازه ضمینه یا Context Size (c) بر آموزشی بردارهای کلمه:

اندازه زمینه c در مدل Skip-gram تعداد کلمات در نظر گرفته شده به عنوان زمینه در دو طرف کلمه مورد نظر را تعیین می کند. تغییر c هم بر کیفیت و هم ماهیت بردارهای کلمه یاد گرفته شده تأثیر می گذارد. در واقع، هنگام آموزش بردارهای کلمه، انتخاب اندازه زمینه (c) به طور قابل توجهی بر جاسازی(embedding) های حاصل تأثیر می گذارد. در اینجا اتفاقی است که برای مقادیر مختلف c اتفاق می افتد آمده است:

1. c کوچک (به عنوان مثال، c = 1):

* زمینه محلی(local context) را در بر می گیرد: این مدل بر روابط نزدیک بین کلمات، مانند وابستگی های نحوی (syntactic dependencies) مثلاً "dog" → "barks" تمرکز می کند. که آن را مناسب برای تسک هایی مانند part-of-speech tagging میکند.
* تعبیه‌ها بیشتر مختص زمینه‌های بلافاصه (immediate contexts) هستند، اما ممکن است نتوانند روابط معنایی گسترده‌تری را به تصویر بکشند (به عنوان مثال، "dog" و "wolf" ممکن است به شدت به هم مرتبط نباشند).

2. c متوسط ​​(به عنوان مثال، c = 5):

* تعادل روابط نحوی و معنایی (syntactic and semantic relationships) را به دست می آورد.
* کلماتی که هم از نظر نحوی و هم از نظر معنایی به هم مرتبط هستند، تعبیه های معناداری خواهند داشت. این معمولاً رایج‌ترین انتخاب برای جاسازی کلمات همه منظوره است که برای تسک های مختلف قابل استفاده است.

3. c بزرگ (مثلاً c = 100):

* زمینه کلی (global context) را به تصویر می کشد: مدل روابط معنایی گسترده تری را بین کلمات می آموزد، که برای کارهایی مانند مدل سازی موضوع (topic modeling) یا document classification یا در بر گرفتن روابط مبتنی بر قیاس (analogy-based) مفید است (به عنوان مثال، "پادشاه" → "ملکه").
* با این حال، ممکن است اطلاعات نحوی دقیق (fine-grained syntactic information) را از دست بدهد و به دلیل ظاهر شدن کلمات نامرتبط در پنجره زمینه (context window)، نویز ایجاد کند.

خلاصه: انتخاب اندازه زمینه c شامل یک trade-off بین در بر گرفتن ساختارهای نحوی محلی و semantic meaning های گسترده تر است. انتخاب بهینه به ویژگی های زبانی خاص مربوط به کاربرد مورد نظر بستگی دارد.

* c کوچک: بر نحو (syntax)، بافت محلی (local context) تمرکز می کند. این مدل بر همسایگان immediate متمرکز است و روابط نحوی مانند ترتیب کلمات و وابستگی‌های محلی را به تصویر می‌کشد.
* c متوسط: نحو و معنا را متعادل می کند. (Balances syntax and semantics)
* c بزرگ: بر معناشناسی global تأکید دارد اما ممکن است اطلاعات نحوی (syntactic) را کمرنگ کند. این مدل زمینه گسترده‌تری (broader context) را در بر می‌گیرد و روابط معنایی و انسجام موضوعی را در فواصل طولانی‌تر نشان می‌دهد.

تاثیر بر Training Dynamics:

* c کوچک: context word های کمتر به ازای هر کلمه target منجر به آموزش سریع‌تر می‌شود، اما ممکن است به دلیل context محدود، جاسازی‌های less robust داشته باشد.
* c بزرگ: context word های بیشتر، تعداد نمونه‌های آموزشی را در هر کلمه target افزایش می‌دهد اما به منابع محاسباتی بیشتر و زمان‌های آموزشی طولانی‌تری نیاز دارد.

## سوال سوم

بخش 1. توسعه توجه به خود (**Self-Attention**) به عنوان یک شبکه کاملاً متصل

مکانیسم توجه به خود را می توان به صورت زیر نوشت:

که در آن :

فرم شبکه کاملاً متصل:

در اصل، ما می خواهیم نشان دهیم که فرآیند توجه به خود (Self-Attention) را می توان با ضرب ماتریس نشان داد. چرا که ضرب ماتریسی معادل لایه تماما متصل است.

بیایید معادله را بشکنیم:

با جایگزینی داریم:

با فرض اینکه X یک دنباله ورودی از بردارهای کلمه با بعد D است، تبدیل‌های ، و ابعاد D را به بعد جدید نگاشت می‌ کنند:

ماتریس شکلی از الگوی اتصال (a form of connectivity pattern) است که می تواند به عنوان یک ماتریس مجاورت وزنی (weighted adjacency matrix) در یک شبکه کاملا متصل در نظر گرفته شود:

بنابراین

این اثبات نشان می‌دهد که مکانیسم توجه اساساً یک تبدیل (ضرب ماتریسی) را انجام می‌دهد که می‌تواند به طور پیوسته از ورودی به خروجی (continuously from input to output)، معادل فعالیت های یک شبکه کاملاً متصل (actions of a fully connected network) نگاشت شود. (می توانیم عبارت را به شکل هم در نظر بگیریم.)

بخش 2. شمارش پارامترها در ماتریس

توجه به خود (Self-attention) شامل ضرب ماتریس و عملیات Softmax است. به طور خاص، داریم:

• ضرب های شامل : هر کدام با اندازه های

بنابراین، برای هر ماتریس تبدیل (transformation matrix):

از آنجایی که ما این سه ماتریس را داریم:

حال اگر داشته باشیم که و یا اینکه که معمولا چنین است مثلا در مقاله اصلی Transformer ابعاد ورودی و خروجی به attention ثابت اند آنگاه: (D = dimension of each token embedding)

بنابراین با فرض داشتن N ورودی (N = sequence length):

* برای هر موقعیت i,j به یک ماتریس نیاز داریم.
* ما به تعداد از چنین موقعیت هایی داریم. (ماتریس وزن توجه M دارای اندازه N×N است زیرا pairwise similarity ها را برای N توکن محاسبه می کند.)
* بنابراین تعداد کل پارامتر ها چنین می شود:

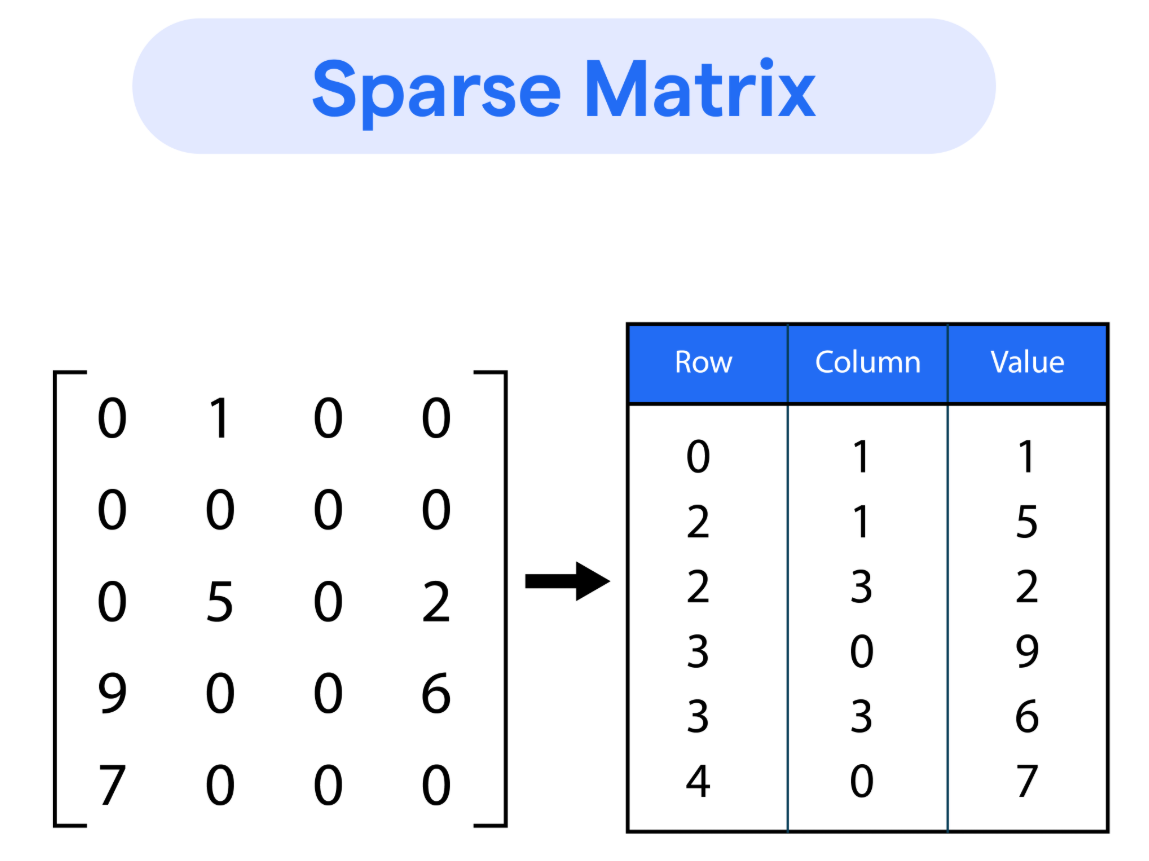
بخش 3. نمایش پراکنده و به اشتراک گذاری پارامتر (Sparse Representation and Parameter Sharing)

در توجه به خود، فرمول ماتریس را می توان به عنوان یک الگوی اتصال پراکنده (sparse connectivity pattern) تفسیر کرد که در آن پارامترهای خاصی به اشتراک گذاشته می شوند:

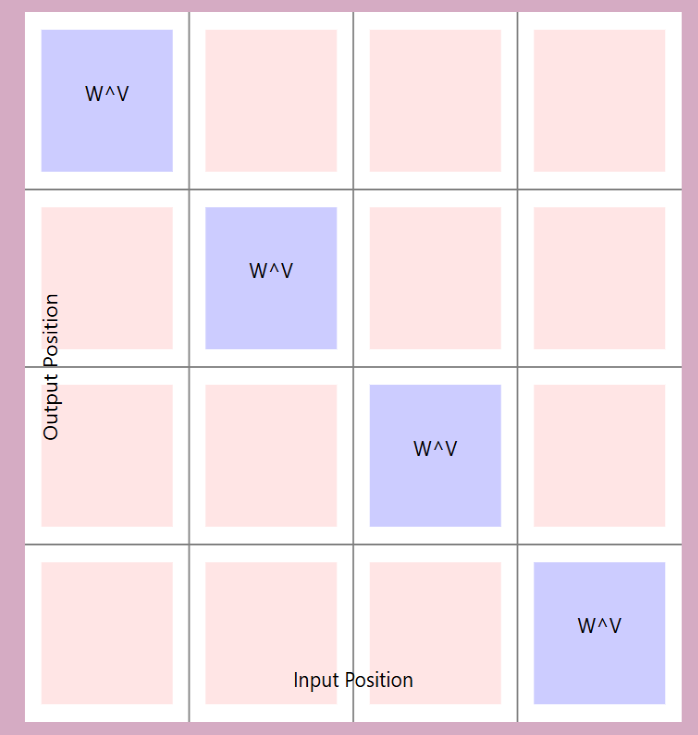
• نکته کلیدی در اینجا پراکندگی (sparsity ) است که به دلیل تمرکز مکانیسم Softmax بر روی چند مقدار کلیدی  
(a few key values) ایجاد می شود.

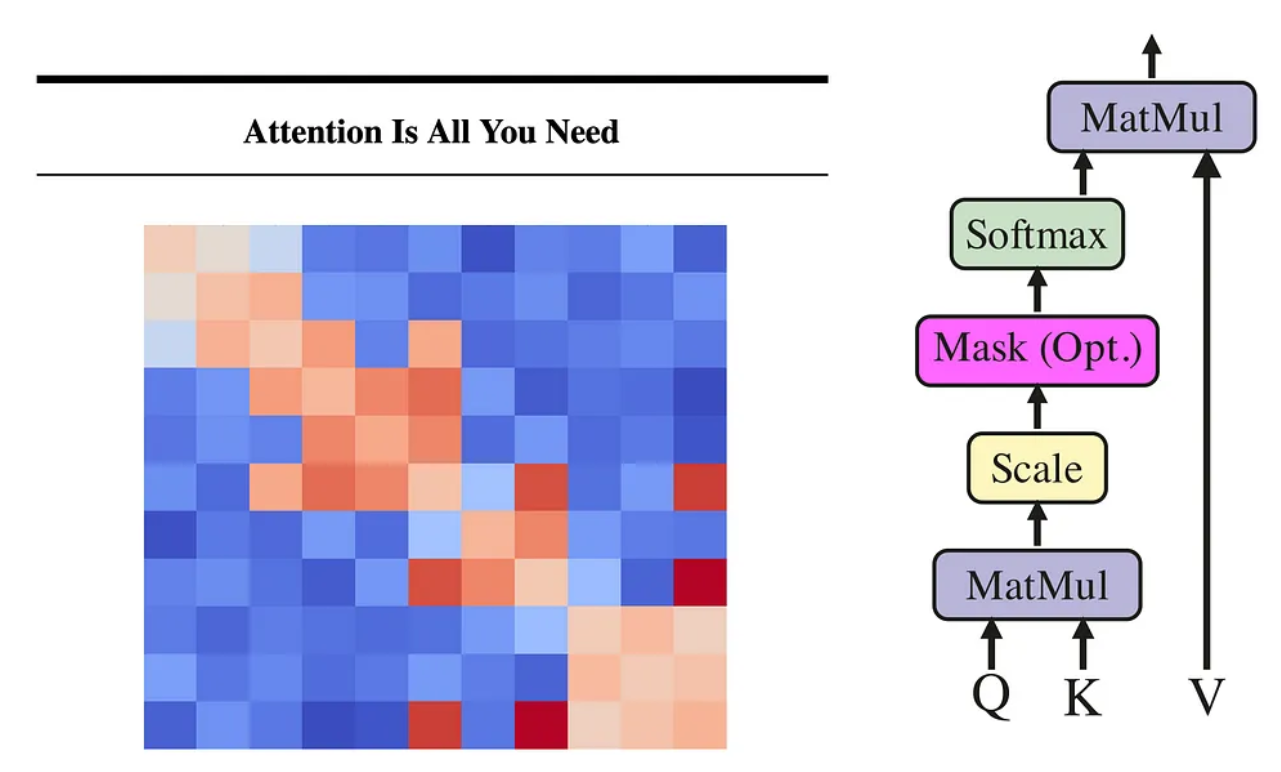
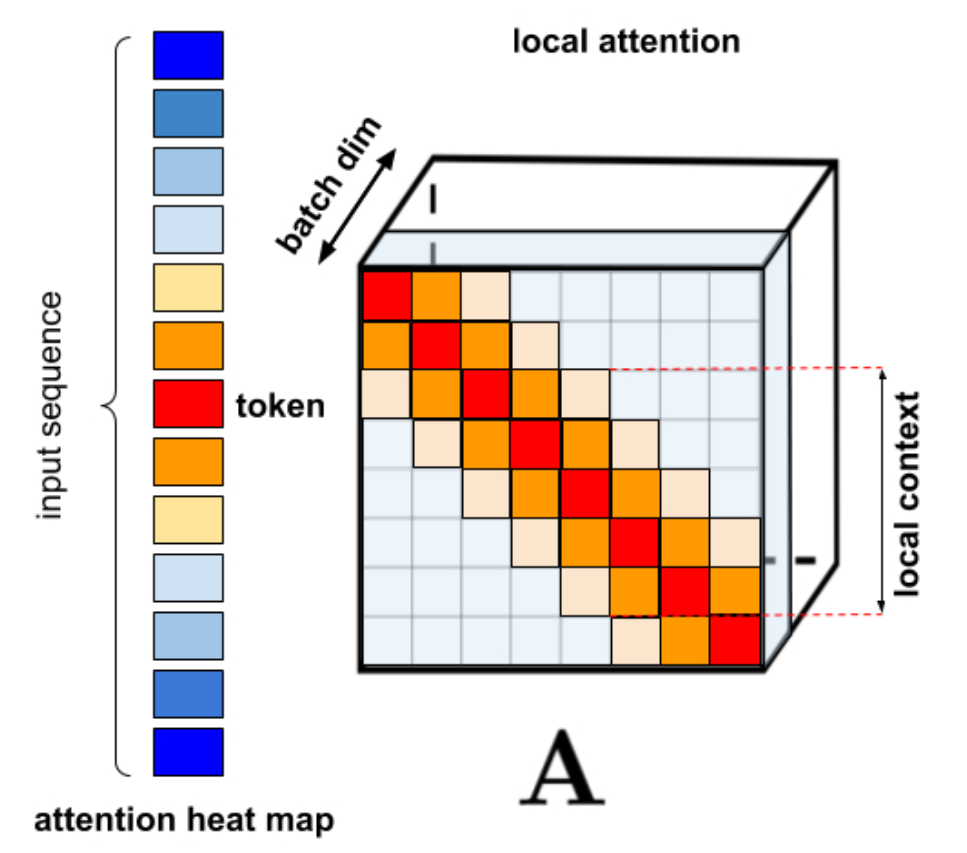
• اشتراک پارامتر در های مشترک رخ می دهد. این اشتراک گذاری تعداد پارامترها را در مقایسه با یک ماتریس کاملا متصل که دارای پارامترهای منحصر به فرد است، کاهش می دهد.

تصویری نمونه از یک Sparse Matrix در ادامه آمده است:



که در مکانیزم توجه چنین می شود:





ماتریسی را تصور کنید که در آن بلوک‌های غیرصفر با مسیرهای حیاتی مشخص شده توسط Softmax مطابقت دارند، در حالی که مناطق وسیع حاوی صفر هستند که نشان‌دهنده اتصالات کمتر مهم یا قابل چشم پوشی (less significant or ignorable) هستند. این را می توان به صورت بصری با یک block matrix نشان داد که در آن:

• **Diagonal blocks** حاوی مقادیر اند (به روز رسانی پارامتر). (بلوک های پارامتر مرتبط با )

• **Off-diagonals** معمولاً در ساختار ماتریسی پراکنده (sparse matrix structure) صفر هستند. (بخش های صفر شده یا Non-contributing ماتریس به دلیل عملیات softmax)

بخش 4. رمزگذاری های موقعیتی (Positional Encodings) و لایه های چند سر (Multi-headed Layers)

حذف positional encoding بر sequential order representation تأثیر می گذارد:

1. توجه به خود بدون رمزگذاری موقعیتی (Multi-headed Self-Attention Without Positional Encodings):

* اگر positional encodings وجود نداشته باشند، مکانیسم توجه چند سر تفاوتی بین ترتیب token های ورودی ایجاد نمی کند.
* بنابراین، ترتیب دهی مجدد ورودی‌ها (reordering inputs)، خروجی‌های توجه حاصل از حذف positional encoding را به دست می‌آورد (same resultant attention outputs upon positional encoding removal):

کد در آن:

2. معادل بودن در هم ریختگی ورودی (Equivariance to Input Shuffling):

برای هر ورودی به هم ریخته (shuffled input)، فقدان آگاهی از ترتیب توالی (sequential awareness) به این معنی است که مکانیسم توجه، نسبت به جایگشت‌ها به طور یکسان رفتار می‌کند - تنسورهای خروجی مشابهی را تولید می‌کند. بنابراین اطلاعات موقعیت (Positional information) برای اطمینان از توجه به ترتیب ورودی بسیار مهم است. بدون آن (Positional Encodings):

این ثابت می‌کند که بدون رمزگذاری موقعیتی (positional encoding)، مکانیسم توجه ذاتاً هیچ ترتیب معنی‌داری را حفظ نمی‌کند (does not inherently preserve any meaningful order).

به بیان دیگر، اگر رمزگذاری موقعیتی حذف شود:

1. ترتیب دهی مجدد ورودی X منجر به همان ترتیب مجدد خروجی Y می شود.
2. این به این دلیل است که مکانیسم توجه روابط زوجی بین token ها را محاسبه می‌کند، که مستقل از ترتیب token ها بدون رمزگذاری موقعیتی هستند.

پیامدها:

* حذف کدگذاری موقعیتی (positional encoding) باعث می شود که خروجی معادل جایگشت های ورودی باشد، به این معنی که هر جایگشتی در دنباله ورودی منجر به یک جایگشت متناظر در خروجی می شود.
* این توانایی مدل را برای گرفتن ساختار توالی (sequence structure)، که برای کارهایی مانند ترجمه ماشینی یا تولید متن حیاتی است، محدود می کند.

نتیجه گیری:

رمزگذاری موقعیتی برای تمایز بین ترتیب های ورودی مختلف ضروری است. بدون آن، توجه به خود، ورودی‌ها را به‌عنوان یک مجموعه به‌جای یک sequence در نظر می‌گیرد و وابستگی‌های sequential حیاتی را از دست می‌دهد.

خلاصه کل سوال

• قسمت 1: نشان دادیم توجه به خود به عنوان یک شبکه کاملاً متصل میتواند توسعه یابد.

• قسمت 2: مشخص شد که ماتریس دارای پارامترهای O(N²D²) است.

• قسمت 3: ماهیت پراکنده و اشتراک پارامترها (sparse nature and parameter sharing) را با استفاده از matrix blocks نشان دادیم.

• قسمت 4: تبیین equivariance  خروجی بدون رمزگذاری موقعیتی. (equivariance of output without positional encoding)

## سوال چهارم

1.  
ابعاد سیگنال ها در مدل Transformer:

پارامترهای داده شده:

* Input vocabulary size: 30,000
* Maximum input sequence length: 2,048
* Hidden vector (embedding) dimension: 768
* Number of encoder blocks: 8
* Number of decoder blocks: 12
* Number of attention heads: 4
* Feedforward layers: 2 (اولی بعد را نصف می کند، دومی آن را دو برابر می کند)
* Batch size: 32
* Input token embedding: 1,024

**اجزا Encoder:**

Input Embedding:

* ورودی: ابعاد اندیس های توکن ها (32, 2048)
* خروجی (After embedding): ابعاد توکن های جاسازی شده (Embedded tokens) (32, 2048, 1024)

Multi-head Self-Attention Mechanism:

* خروجی ماتریس های Query, Key, Value: هر یک دارای ابعاد (32, 2048, 768)
* در واقع، پس از تقسیم به 4 head: هر head دارای ابعاد (32, 2048, 192) است.
* ابعاد Attention scores: (32, 4, 2048, 2048)
* خروجی Scaled Dot-Product Attention در هر head : (32, 2048, 192)
* خروجی Concatenated: (32, 2048, 768)
* خروجی پس از اعمال Output Projection Matrix یا همان : (32, 2048, 1024)
* لایه Add & norm تغییری در ابعاد وجود نمی آورد: (32, 2048, 1024)

Feedforward Network:

* خروجی لایه اول: ابعاد به (32, 2048, 512) نصف میشوند.
* خروجی لایه دوم: ابعاد دو برابر شده و به (32, 2048, 1024) برمیگردند.
* لایه Add & norm تغییری در ابعاد وجود نمی آورد: (32, 2048, 1024)

**اجزا Decoder:**

تعبیه ورودی (Input Embedding):

* ورودی: ابعاد اندیس های توکن ها (Token indices) 🡨 (32, 2048)
* خروجی: ابعاد توکن های جاسازی شده (Embedded tokens) 🡨 (32, 2048, 1024)

Masked Multi-head Self-Attention Mechanism:   
مشابه Multi-head Self-Attention Mechanism رمزگذار (encoder) است.

* خروجی ماتریس های Query, Key, Value: هر یک دارای ابعاد (32, 2048, 768)
* در واقع، پس از تقسیم به 4 head: هر head دارای ابعاد (32, 2048, 192) است.
* ابعاد Attention scores: (32, 4, 2048, 2048)
* ابعاد Mask: (2048, 2048)
* خروجی Scaled Dot-Product Attention در هر head : (32, 2048, 192)
* خروجی Concatenated: (32, 2048, 768)
* خروجی پس از اعمال Output Projection Matrix یا همان : (32, 2048, 1024)
* لایه Add & norm تغییری در ابعاد وجود نمی آورد: (32, 2048, 1024)

توجه رمزگذار-رمزگشا (Multi-head Cross-Attention Mechanism):

* Query از خروجی Decoder: (32, 2048, 1024)
* Key و Value از خروجی Encoder: (32, 2048, 1024)
* خروجی ماتریس های Query, Key, Value: هر یک دارای ابعاد (32, 2048, 768)
* در واقع، پس از تقسیم به 4 head: هر head دارای ابعاد (32, 2048, 192) است.
* ابعاد Attention scores: (32, 4, 2048, 2048)
* خروجی توجه به ازای هر head: (32, 2048, 192)
* خروجی Concatenated: (32, 2048, 768)
* خروجی Output Projection Matrix یا همان : (32, 2048, 1024)
* لایه Add & norm تغییری در ابعاد وجود نمی آورد: (32, 2048, 1024)

Feedforward Network:

* خروجی لایه اول (نصف کننده): (32, 2048, 512)
* خروجی لایه دوم (دوبرابر کننده): (32, 2,048, 1024)
* لایه Add & norm تغییری در ابعاد وجود نمی آورد: (32, 2048, 1024)

Final Linear Layer (LM head for output vocabulary):

* ورودی: ابعاد بردار های توکن های خروجی (32, 2048, 1024)
* خروجی: ابعاد اندیس های توکن ها (32, 2048)

2.  
تعداد پارامترها در هر Component:

Embedding Layer:

* Embedding توکن ها:  
   30,000 (vocab size) \* 1024 (embedding dim) = 30,720,000 parameters

Multi-head Self-Attention Mechanism (در هر لایه):

* وزن ماتریس های Query، Key، Value: هر کدام 1024 \*192 \* 4 = 1024 \* 768 = 786,432 پارامتر است.
* مجموع کل برای Q, K, V: 3 \* 786,432 = 2,359,296 پارامتر
* Output Projection Matrix یا همان : 768 \* 1024 = 786,432 پارامتر
* مجموع در هر لایه توجه به خود: 2,359,296 + 786,432 = 3,145,728 پارامتر

Feedforward Network (در هر لایه):

* لایه اول: 1024 \* 512 = 524,288 پارامتر (با فرض نبودن بایاس در غیر این صورت باید به علاوه 512 یعنی در نهایت 524,800 شود.)
* لایه دوم: 512 \* 1024 = 524,288 پارامتر (با فرض نبودن بایاس در غیر این صورت باید به علاوه 1024 یعنی در نهایت 524,312 شود.)
* مجموع در هر لایه Feedforward: 524,288 + 524,288 = 1,048,576 پارامتر

Encoder (در هر بلوک):

* توجه به خود: 3,145,728 پارامتر
* شبکه Feedforward: 1,048,576 پارامتر (با فرض در نظر نگرفتن بایاس)
* فرض میکنیم روش norm به صورت learnable است مثلا در صورت استفاده از LayerNorm با پارامتر های قابل یادگیری باید تعداد پارامتر های learnable آن را بشماریم یعنی همان scaling و shifting (gamma (γ) و beta (β)). در هر لایه LayerNorm: 2 × 1024 = 2048 پارامتر داریم. چون دو لایه LayerNorm در هر بلوک Encoder داریم پس 2 × 2048 = 4096 پرامتر در مجموع خواهیم داشت.)  
  در غیر این صورت، یعنی در صورت unlearnable بودن normalization تعداد پارامتر های این لایه برابر با 0 بود.
* مجموع پارامتر ها در هر بلوک Encoder: 3,145,728 + 1,048,576 + 4,086 = 4,198,400 پارامتر
* مجموع پارامتر ها برای 8 بلوک Encoder: 8 \* 4,198,400 = 33,587,200 پارامتر

Decoder (در هر بلوک):

* Masked Self-Attention: 3,145,728 پارامتر (محاسبات مشابه Multi-head Self-Attention Mechanism در encoder که در بالا انجام دادیم چون ابعاد ورودی ها و خروجی ها و هایپر پارامتر ها ثابت است.)
* Encoder-Decoder Cross Attention: 3,145,728 پارامتر (محاسبات مشابه Multi-head Self-Attention Mechanism در encoder که در بالا انجام دادیم چون ابعاد ورودی ها و خروجی ها و هایپر پارامتر ها ثابت است.)
* Feedforward Network: 1,048,576 پارامتر (محاسبات مشابه Feedforward Network در encoder که در بالا انجام دادیم. توجه داریم که بایاس ها را در نظر نگرفتیم.)
* فرض میکنیم روش norm به صورت learnable است مثلا در صورت استفاده از LayerNorm با پارامتر های قابل یادگیری باید تعداد پارامتر های learnable آن را بشماریم یعنی همان scaling و shifting (gamma (γ) و beta (β)). در هر لایه LayerNorm: 2 × 1024 = 2048 پارامتر داریم. چون سه لایه LayerNorm در هر بلوک Decoder داریم پس 3 × 2048 = 6144 پرامتر در مجموع خواهیم داشت.)  
  در غیر این صورت، یعنی در صورت unlearnable بودن normalization تعداد پارامتر های این لایه برابر با 0 بود.
* مجموع هر بلوک Decoder: 3,145,728 + 3,145,728 + 1,048,576 + 6144 = 7,346,176 پارامتر
* مجموع برای 12 بلوک رمزگشا: 12 \* 7,346,176 = 88,154,112 پارامتر

Final Linear Layer (LM head for output vocabulary):

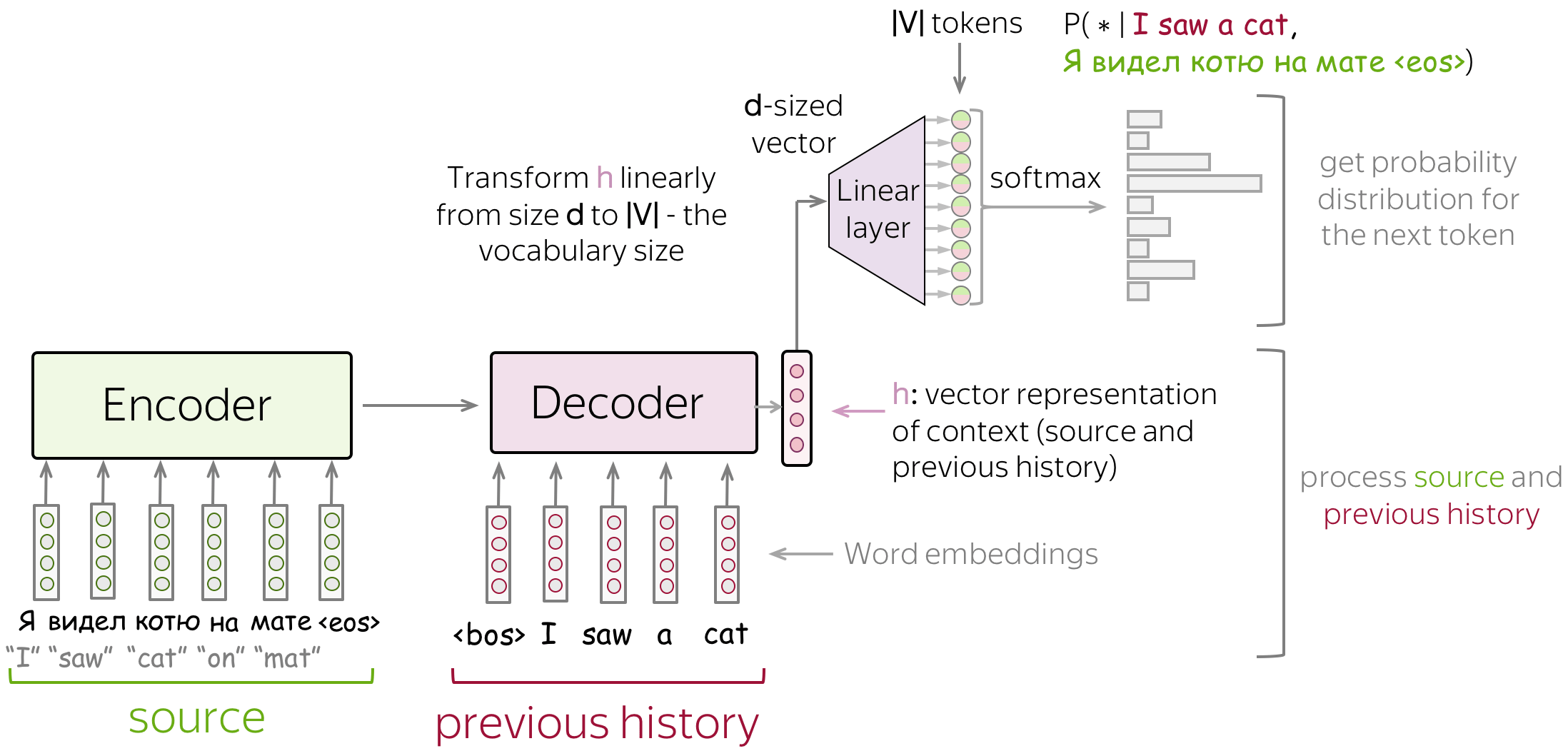
* + 1024 (embedding dim) \* 30,000 (vocab size) = 30,720,000 پارامتر

بنابراین، کل پارامترها در مدل به شرح زیر است:

* Embedding: 30,720,000 (با فرض اینکه برای encoder و decoder از یک embedding استفاده میکنیم. در صورت استفاده از embedding های مختلف این عدد ضرب در 2 خواهد شد به عنوان مثال اگر جملات مبدا انگلیسی و جملات مقصد فرانسوی در ترجمه باشند از دو embedding مجزا استفاده میکنیم.)
* بلوک های رمزگذار (Encoder Blocks): 33,587,200
* بلوک های رمزگشا (Decoder Blocks): 88,154,112
* Final Linear Layer: 30,720,000
* مجموع: 30,720,000 + 33,587,200 + 88,154,112 + 30,720,000 = 183,181,312 پارامتر

3.  
در دو شکل زیر مثالی از این تسک به همراه توضیحات خلاصه آمده است. ابتدا به این دو شکل توجه میکنیم. سپس به توضیح دقیق میپردازیم.

texte du
lien



در یک مدل ترجمه ماشینی مبتنی بر Transformer، مانند ترجمه انگلیسی به فرانسوی، فرآیند شامل دو جزء اصلی است: رمزگذار و رمزگشا (the encoder and the decoder). در ادامه نحوه عملکرد هر کدام آمده است:

**رمزگذار:**

1. پردازش ورودی:
   * جمله ورودی: "The cat sits on the mat."
   * Tokenization: جمله به tokens تقسیم می شود: ["The", "cat", "sits", "on", "the", "mat", "."]
   * Embedding: هر token به یک بردار با اندازه ثابت (Embedding) تبدیل می شود که معرف معنای معنایی (semantic meaning) آن است.
2. رمزگذاری (Encoding):
   * توالی embedding ها از لایه های رمزگذار متعدد عبور داده می شود.
   * هر لایه مکانیسم‌های خودتوجهی (self-attention mechanisms) و feedforward neural networks را اعمال می‌کند تا روابط بین کلمات را بدون در نظر گرفتن موقعیت آنها به تصویر بکشد. (برای همین در ابتدا positional encoding اضافه میکنیم.)
   * خروجی دنباله ای از context-rich representations برای هر token ورودی است.

**رمزگشا:**

1. پردازش ورودی:
   * Start Token: رمزگشا با یک token شروع دنباله (start-of-sequence) خاص (به عنوان مثال، "<start>") شروع می شود.
   * Embedding: این token در یک وکتور embed می شود.
2. رمزگشایی (Decoding):
   * Masked self-attention: رمزگشا ورودی خود را پردازش می کند (در ابتدا token "<start>") برای درک context خروجی تولید شده تاکنون.
   * Encoder-Decoder Cross Attention: به خروجی غنی شده رمزگذار توجه می کند تا جمله ورودی را به زبان مقصد تراز و ترجمه کند.
   * Feedforward Network: اطلاعات ترکیبی را برای تولید توکن بعدی پردازش می کند.
   * تولید خروجی: رمزگشا token بعدی را در دنباله پیش بینی می کند (به عنوان مثال، "Le").
3. فرآیند مکرر (Iterative Process):
   * token پیش بینی شده به رمزگشا برگشت داده می شود.
   * مراحل 2 و 3 تکرار می شوند و توکن ها را یکی یکی تولید می کنند: (one token at a time)  
     "Le", "chat", "s'assoit", "sur", "le", "tapis", "."
   * این فرآیند تا زمانی ادامه می‌یابد که یک token پایان دنباله (end-of-sequence token) خاص (به عنوان مثال، "<end>") تولید شود یا اینکه به maximum length جمله برسیم.

خلاصه:

* رمزگذار جمله ورودی را به مجموعه ای از بازنمایی های context-aware تبدیل می کند و معنی و روابط بین کلمات را ضبط می کند.
* رمزگشا با توجه به این نمایش ها و تولید یک نشانه در یک زمان در زبان مقصد (one token at a time in the target language)، جمله ترجمه شده را تولید می کند.

این مکانیزم به مدل Transformer اجازه می دهد تا با استفاده از معماری رمزگذار-رمزگشا، متن را به طور مؤثر ترجمه کند، جایی که رمزگذار ورودی را درک می کند، و رمزگشا خروجی مربوطه را می سازد.

## سوال پنجم

الف) چرا حذف NSP ممکن است عملکرد مدل را بهبود بخشد؟

وظیفه Next Sentence Prediction (NSP) در آموزش BERT به دلایل مختلفی مورد انتقاد قرار گرفته است، و حذف آن اغلب می‌تواند عملکرد در تسک های پایین‌دستی را بهبود بخشد:

1. عدم ارتباط به بسیاری از وظایف پایین دستی:

* NSP بر پیش بینی اینکه آیا دو جمله به صورت متوالی به هم متصل هستند یا خیر، تمرکز دارد، که مستقیماً به بسیاری از وظایف پایین دستی مانند طبقه بندی، پاسخگویی به سؤال، یا شناسایی موجودیت نامدار (classification,   
  question-answering, or named entity recognition) مرتبط نیست.
* حذف NSP به مدل این امکان را می دهد که بیشتر بر روی تسک Masked Language Modeling (MLM) تمرکز کند، که توانایی آن را برای یاد گرفتن بازنمایی های متنی بهبود می بخشد.

2. ماهیت ساده NSP:

* NSP یک تسک طبقه بندی باینری است که ممکن است به خوبی با پیچیدگی های روابط بین جملات در وظایف دنیای واقعی متن هماهنگ نباشد.
* مطالعات (به عنوان مثال، RoBERTa) نشان داده است که حذف NSP و آموزش فقط با MLM امکان پیش‌آموزش (pretraining) بهتر و تعمیم بهتر را فراهم می‌کند.

3. عدم تعادل داده ها و نویز:

* در وظیفه NSP، جملات اغلب به طور مصنوعی جفت می شوند، که ممکن است نویز را به فرایند آموزش وارد کند. به عنوان مثال، جملات "non-matching" ممکن است هنوز از نظر معنایی مرتبط باشند اما به عنوان نمونه های منفی طبقه بندی شوند و مدل را گیج کنند.

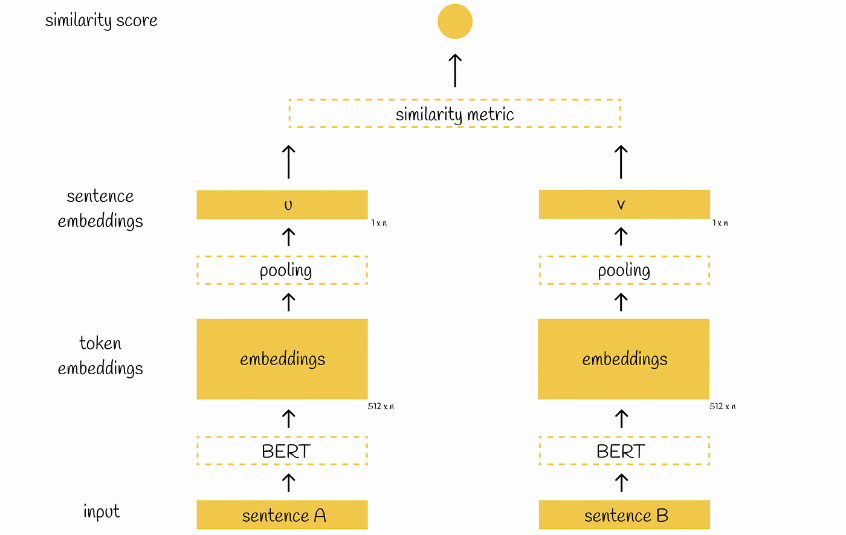
4. بهبودهای مشاهده شده در RoBERTa به صورت تجربی:

* RoBERTa که NSP را حذف می کند و از objective آموزشی قوی تر فقط MLM با مجموعه داده های بزرگتر و هایپرپارامترهای بهینه شده استفاده می کند، در مقایسه با BERT عملکرد برتر را در معیارهای مختلف نشان داده است.
* این نشان می‌دهد که تمرکز کامل روی MLM به مدل کمک می‌کند تا جاسازی‌های متنی بهتری را بیاموزد بدون اینکه حواسش به تسک NSP تعمیم‌پذیری کمتری دارد پرت شود.

به طور خلاصه، pre-training یا همان پیش‌آموزش BERT شامل دو تسک است: MLM و NSP. وظیفه NSP مدل را آموزش می دهد تا پیش بینی کند که آیا یک جمله داده شده B به طور طبیعی از جمله A پیروی می کند یا خیر، با هدف کمک به BERT در درک روابط بین جملات.

با این حال، تحقیقات بعدی اثربخشی وظیفه NSP را زیر سوال برد. به عنوان مثال، مدل RoBERTa، که objective تسک NSP را حذف می کند، عملکرد بهتری را در وظایف مختلف پایین دستی نشان داده است. این نشان می دهد که حذف NSP می تواند به تعمیم و کارایی بهتر منجر شود. یک دلیل این است که وظیفه NSP ممکن است نویز ایجاد کند، زیرا طبقه بندی باینری جفت جملات همیشه روابط ظریف بین جملات را نشان نمی دهد. با تمرکز صرف بر روی وظیفه MLM، مدل می تواند بر درک بازنمایی کلمات متنی بدون پیچیدگی اضافی طبقه بندی جفت جمله تمرکز کند.

ب) آموزش درک جملات عمومی با استفاده از لایه های pooling



در Sentence-BERT (SBERT) و دیگر رویکردهای جاسازی جمله، لایه های ادغام (pooling) برای استخراج یک نمایش برداری از کل جمله به جای تکیه بر نمایش های single token استفاده می شود. در اینجا نحوه استفاده از لایه های ادغام(pooling) برای آموزش درک کلی از جملات به یک مدل مبتنی بر BERT آمده است:

1. روش های ادغام

لایه‌های ادغام، token embeddings را در یک بردار با طول ثابت ادغام می‌کنند و معنای کلی یک جمله را به تصویر می‌کشند. روش های متداول ادغام عبارتند از:

* [CLS] Token Representation:
  + در BERT، اولین token ([CLS]) اغلب برای نشان دادن کل جمله استفاده می شود. تعبیه آن در طول آموزش تسک پایین دست به خوبی تنظیم می شود.
  + با این حال، این ممکن است همیشه معنای معنایی کامل جمله را در بر نگیرد.
* Mean Pooling:
  + میانگین همه token embeddings در جمله را محاسبه میکنیم.
  + این روش فرض می کند که همه token ها به یک اندازه در معنای جمله نقش دارند.
* Max Pooling:
  + حداکثر مقدار را در هر بعد از همه جاسازی‌های token انتخاب میکنیم.
  + مهمترین ویژگی های جمله را برجسته می کند.

2. Sentence-BERT Approach

* در Sentence-BERT، یک معماری شبکه siamese برای محاسبه جاسازی جملات با تنظیم دقیق BERT برای کارهایی مانند شباهت معنایی(semantic similarity) استفاده می شود.
* هدف آموزش (Training Objective): به جای NSP، SBERT از وظایف زوجی(pairwise tasks) مانند contrastive learning (به عنوان مثال، پیش بینی اینکه آیا دو جمله از نظر معنایی مشابه یا متفاوت هستند) استفاده می کند. این مدل را تشویق می کند تا جاسازی جملات کلی (general sentence embeddings) را بیاموزد.

3. تنظیم دقیق با لایه های ادغام (Fine-Tuning with Pooling Layers)

هنگام تنظیم دقیق BERT برای وظایف پایین دست، لایه های ادغام را می توان به صورت زیر اعمال کرد:

* یک لایه ادغام اضافه کنیم:
  + پس از لایه transformer نهایی، یک مکانیسم ادغام (به عنوان مثال، mean pooling or max pooling) را اعمال کنیم تا جاسازی های توکن را در یک بردار یک جمله واحد جمع کنیم.
* آموزش در Sentence-Level Tasks:
  + از جاسازی‌های pooled به‌عنوان ورودی برای طبقه‌بندی‌کننده پایین‌دست یا task-specific layer استفاده کنیم.
  + وظایف نمونه عبارتند از:
    - طبقه بندی متن: از بردار ادغام شده (pooled vector) برای طبقه بندی جمله به دسته ها(categories) استفاده کنیم.
    - تشابه معنایی (Semantic Similarity): شباهت کسینوسی بین جاسازی‌های جمله برای وظایف مرتبط را محاسبه کنیم.
* تنظیم دقیق نظارت شده (Supervised Fine-Tuning):
  + مدل را بر روی داده های برچسب دار با اهداف در سطح جمله (sentence-level objectives) آموزش دهیم. به عنوان مثال، در تجزیه و تحلیل احساسات، pooled embedding برای پیش بینی برچسب احساس یک جمله استفاده می شود.

4. چرا ادغام کمک می کند؟

لایه های ادغام یک نمایش با طول ثابت از یک جمله را ارائه می دهند که مدل را قادر می سازد:

* زمینه جهانی (global context) و semantic meaning جمله را دریابد.
* در کارهای پایین دستی که نیاز به درک در سطح جمله دارند، کارآمد عمل کند.

نمونه Workflow

* جمله ورودی: "Artificial intelligence is transformative."
* Token embeddings:
* ادغام را اعمال کنیم: embedding ها را در یک بردار واحد Aggregate میشوند:
* از pooled embedding برای طبقه بندی یا کارهای مشابه استفاده کنیم.

با ادغام لایه‌های ادغام(pooling layers) و آموزش وظایف در سطح جمله، مدل می‌تواند نمایش‌های غنی‌تر و کلی‌تری از جملات مناسب برای کاربردهای پایین‌دستی را بیاموزد.

به بیان دیگر:

تنظیم دقیق BERT برای کارهایی که نیاز به درک در سطح جمله دارند، اغلب شامل استخراج یک بردار با اندازه ثابت است که کل جمله را نشان می دهد. لایه های ادغام کننده (Pooling layers) در این فرآیند نقش اساسی دارند. یک رویکرد متداول این است که عملیات ادغام را بر روی جاسازی های توکن تولید شده توسط BERT اعمال کنیم تا یک نمایش برداری واحد برای جمله ایجاد کنیم.

Sentence-BERT (SBERT) نمونه ای از این رویکرد است. SBERT مدل BERT را با استفاده از جفت‌های جمله و برچسب‌های شباهت (pairs and similarity labels) fine-tune می‌کند و مدل را برای تولید جاسازی‌های semantically meaningful بهینه می‌کند. در طول تنظیم دقیق، SBERT یک عملیات ادغام - مانند mean pooling or max pooling - را روی جاسازی‌های توکن اعمال می‌کند تا یک بردار جمله با اندازه ثابت به دست آورد. این بردار معنای کلی جمله را به تصویر می‌کشد و عملکرد مؤثری را در کارهایی مانند semantic textual similarity و خوشه‌بندی ممکن می‌سازد.

مدل‌هایی مانند SBERT می‌توانند با ترکیب لایه‌های pooling در طول تنظیم دقیق، یاد بگیرند که جاسازی‌های جملاتی قوی ایجاد کنند که معنای کلی جملات را در بر می‌گیرد و کاربرد آن‌ها را برای طیف گسترده‌ای از وظایف پایین‌دستی افزایش می‌دهد.

پایان