نام و نام خانوادگی:

رضا قربانی پاجی

شماره دانشجویی:

403206565

تمرین پنجم درس یادگیری ماشین

سوال 1

الف)

الف) قسمت 1

وقتی سایز پنجره برابر 2 است هر کلمه می تواند با دو کلمه قبل و بعدش جفت بشه. منتهی برای جفت های اول و آخر این تعداد به 2 و 3 جفت میرسه در نهایت خواهیم داشت

تعداد جفتهای مثبت $= 2+3+6\times4+3+2=34$

به ازای هر جفت مثبت $k=5$ تا جفت منفی خواهیم داشت

جفتهای منفی $= k \times$ تعداد جفتهای مثبت $= 5\times34 = 170$ $\Rightarrow$ تعداد کل $= 34+170 = 204$

الف) قسمت 2

در مدل CBOW اگر اندازه پنجره 4 باشد (یعنی context window size برابر با 4)، تعداد کلمات ورودی (input words) برای پیش بینی هر target word برابر $2\times4=8$ خواهد. این شامل 4 کلمه از سمت چپ و 4 کلمه از سمت راست هدف است

الف) قسمت 3

اگر ابعاد embedding ورودی d_{model} باشد و تعداد h، head وجود داشته باشد، هر head بعدی برابر با $\frac{d_{model}}{h}$ خواهد داشت. بنابراین ابعاد ماتریکس های W_Q، W_K، W_V برای هر head برابر با:

$W_Q: d_{model} \times \frac{d_{model}}{h}$

$W_K:$ 〃 $\times$ 〃

$W_V:$ 〃 $\times$ 〃

تنظیم این پارامترها به کاهش پیچیدگی محاسباتی کمک می کند، زیرا تعداد عملیات کاهش یافته و محاسبات به صورت موازی روی head ها توزیع می شود

ب)

ب-1)

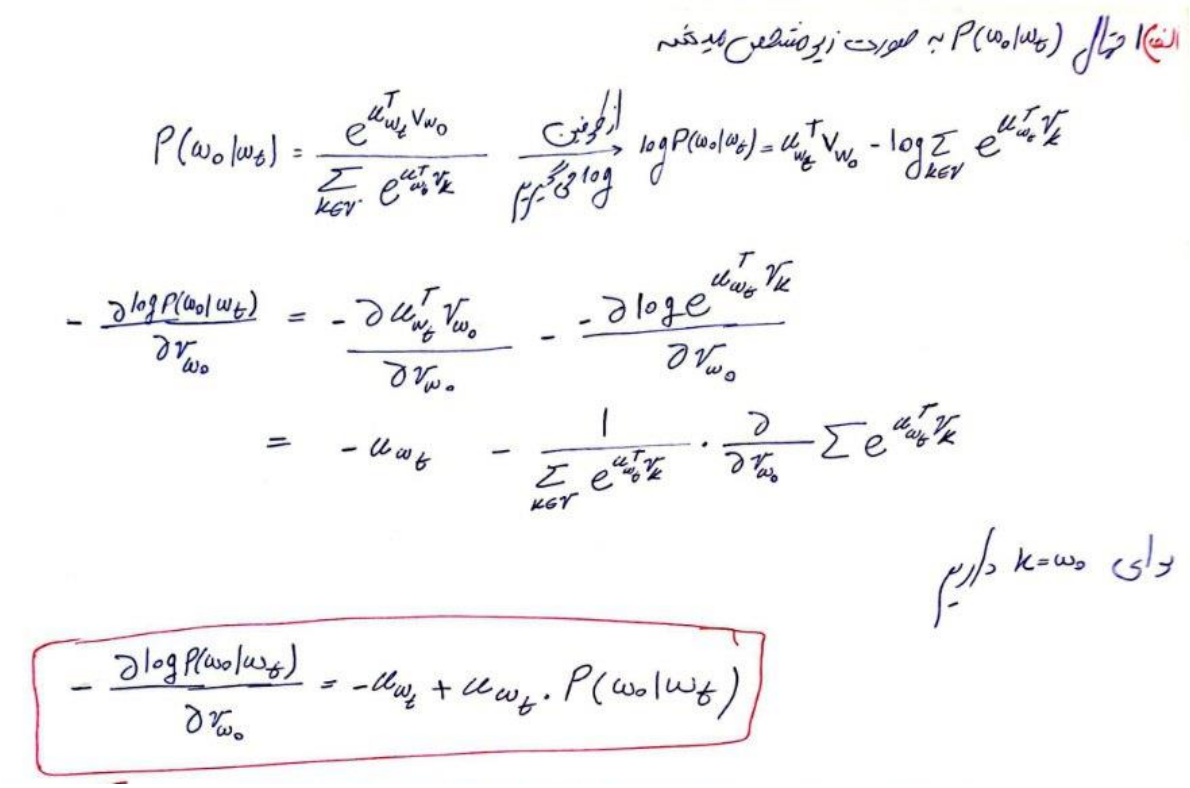
صحیح است. در مدل BERT، positional encoding به صورت trainable embedding است و در طول مراحل آموزش بهینه می‌شود در حالی که در معماری اصلی Transformer، positional encoding به صورت ثابت و سینوسی طراحی شده است

ب-2)

غلط است. افزایش تعداد attention heads همیشه باعث بهبود عملکرد نمی‌شود اگر تعداد head ها بیش از حد افزایش یابد ممکن است منجر به افزایش پیچیدگی محاسباتی و مشکلات overfitting شود همچنین، تاثیر افزایش تعداد heads بستگی به ظرفیت مدل و تنظیمات دیگر مانند اندازه embedding دارد.

سوال 2

الف)



ب)

c=1 :

- یک پنجره کوچک روابط بافت محلی را ثبت می‌کند. به این معنا که:
  - مدل بر روی جفت کلماتی که مستقیماً در کنار یکدیگر ظاهر می‌شوند، تمرکز می‌کند.
  - نمایش‌هایی که یاد گرفته می‌شوند، بیشتر روابط نحوی (مانند ساختار دستوری یا وابستگی کلمات) را منعکس می‌کنند (مثلاً "گربه" و "است" یا "یک").
- انتظار :تعبیه‌های کلمات (word embeddings) به خوبی وابستگی‌های کوتاه‌برد را ثبت می‌کنند، اما ممکن است در مدل‌سازی روابط معنایی گسترده‌تر ناتوان باشند.

C=5:

- یک پنجره متوسط هم بافت محلی و هم برخی روابط معنایی را ثبت می‌کند.

- کلماتی که در یک ساختار شبیه به جمله ظاهر می‌شوند، مد نظر قرار می‌گیرند و تعادلی بین الگوهای نحوی و معنایی ایجاد می‌شود.

- انتظار :تعبیه‌های کلمات، اطلاعات نحوی و معنایی را به طور همزمان نمایش می‌دهند و آن‌ها را برای بسیاری از وظایف پردازش زبان طبیعی کاربردی می‌سازد.

:C=100

- یک پنجره بزرگ روابط معنایی گسترده را ثبت می‌کند و بر روی هم‌وقوعی کلمات در دامنه‌های بزرگ‌تر تمرکز دارد.
  - کلماتی که از نظر موضوعی به هم مرتبط هستند، اما لزوماً از نظر نحوی نزدیک نیستند، بیشتر در نظر گرفته می‌شوند (مثلاً "گربه" و "حیوان").
- انتظار :تعبیه‌ها وابستگی‌های معنایی بلندبرد را ثبت می‌کنند که برای وظایفی که به درک معنایی نیاز دارند (مانند مدل‌سازی موضوعی یا اندازه‌گیری شباهت معنایی) مفید است. با این حال، ممکن است جزئیات ظریف نحوی از دست برود.

سوال 3

قسمت 1)

در مکانیزمSelf-Attention ، دنباله‌ی ورودی که شامل جاسازی کلمات (یا توکن‌ها) است از طریق سه ماتریس متفاوت پردازش می‌شود:

1. پرسش‌ها :(Q) این ماتریس نمایانگر کلمه یا توکنی است که در حال حاضر مورد توجه قرار گرفته است. برای هر کلمه در دنباله، یک بردار پرسش محاسبه می‌شود تا اهمیت آن نسبت به سایر توکن‌ها مشخص شود.
2. کلیدها :(K) این بردارها ویژگی‌هایی از هر توکن در دنباله را نشان می‌دهند که ممکن است با یک پرسش مطابقت داشته باشد. کلیدها برای محاسبه امتیازات توجه با پرسش‌ها جفت می‌شوند.
3. مقادیر :(V) این ماتریس حاوی اطلاعات واقعی توکن‌هاست که بر اساس میزان اهمیت (امتیازات توجه) وزن‌دهی می‌شوند.

مراحل در فرم ماتریسی:

1. محاسبه‌ی امتیازات توجه:(Dot Product Attention)

   امتیازات توجه از طریق ضرب نقطه‌ای بین ماتریس پرسش (Q) و ترانهاده‌ی ماتریس کلید (K^T) محاسبه می‌شود. به صورت ریاضی:

$$Scores = Q \cdot K^T$$

این عملیات تعیین می‌کند که هر پرسش چقدر به هر کلید در دنباله مرتبط است. اگر حاصل ضرب نقطه‌ای بالا باشد، نشان‌دهنده‌ی ارتباط قوی است.

2. مقیاس‌دهی با $\sqrt{D}$:

   بعد بُردار کلید (D) برای مقیاس‌دهی امتیازات توجه استفاده می‌شود. این مقیاس‌دهی با تقسیم امتیازات بر $\sqrt{D}$ از بزرگ شدن بیش از حد مقادیر در هنگام کار با ابعاد بالا جلوگیری می‌کند:

$$Scaled\ Scores = \frac{Q \cdot K^T}{\sqrt{D}}$$

3. تابع Softmax:

امتیازات مقیاس‌شده از طریق تابع Softmax نرمال‌سازی می‌شوند تا به احتمال تبدیل شوند. این مرحله تضمین می‌کند که جمع وزن‌های توجه برای هر بردار پرسش برابر ۱ باشد:

$$Attention\ Weights = Softmax\left(\frac{Q \cdot K^T}{\sqrt{D}}\right)$$

این وزن‌ها نشان‌دهنده‌ی میزان اهمیت نسبی هر کلمه در دنباله برای یک پرسش خاص هستند.

4. وزن‌دهی مقادیر:

وزن‌های توجه سپس در ماتریس مقدار (V) ضرب می‌شوند. این مرحله اطلاعات همه‌ی توکن‌ها را بر اساس ارتباطشان با پرسش فعلی ترکیب می‌کند:

$$Y = Softmax\left(\frac{Q \cdot K^T}{\sqrt{D}}\right).V$$

این فرمول‌بندی ماتریسی به صورت کارآمد توجه را برای تمام توکن‌ها در دنباله به طور همزمان محاسبه می‌کند. نتیجه (Y) دنباله‌ای از بردارهاست، که هر بردار خروجی ترکیب وزن‌دهی‌شده‌ای از بردارهای مقدار ورودی است. وزن‌ها بر اساس شباهت بین پرسش و کلیدها و نرمال‌سازی توسط Softmax تعیین می‌شوند.

این فرآیند هسته‌ی اصلی مکانیزم Self-Attention است که به مدل‌هایی مانند Transformer امکان می‌دهد تا وابستگی‌های بلندمدت و روابط متنی در دنباله‌ها را به خوبی درک کنند.

قسمت 2)

تحلیل ابعاد به صورت گام به گام:

1. ماتریس‌های ورودی:
   - ماتریس‌های Q(پرسش‌ها) و K(کلیدها) ابعادی برابر با N × D دارند:
     - N: طول دنباله که تعداد توکن‌های موجود در دنباله ورودی است.
     - D: ابعاد بردارهای کلمه (یا اندازه جاسازی هر توکن).
2. ماتریس کلید ترانهاده‌شده:
   - ماتریس ترانهاده‌شده K^T ابعادی برابر با D × N دارد. ترانهاده کردن، سطرها و ستون‌های K را جابجا می‌کند.
3. ضرب ماتریس:$(Q \cdot K^T)$
   - ضرب Q(N×D) در $K^T(D \times N)$ ماتریسی با ابعاد N×N ایجاد می‌کند.
   - هر عنصر در این ماتریس N×N نشان‌دهنده امتیاز توجه بین دو توکن در دنباله است (مثلاً اینکه توکن i چقدر به توکن j توجه می‌کند).
4. نرمال‌سازی:Softmax
   - تابع Softmax به صورت عنصری روی هر سطر ماتریس N×N اعمال می‌شود. این کار اطمینان می‌دهد که مجموع وزن‌های توجه برای هر توکن برابر ۱ باشد. اندازه ماتریس همچنان N×Nباقی می‌ماند.

5. ضرب در مقادیر(V):

- ماتریس امتیازات توجه (N×N) سپس در ماتریس V که ابعاد N×D دارد ضرب می‌شود. این مرحله کمک می‌کند تا سهم وزنی تمام توکن‌ها برای هر پرسش ترکیب شود.

  خروجی این عملیات ماتریسی با ابعاد N×D خواهد بود که با ابعاد ماتریس اصلی پرسش یا مقدار یکسان است.

تحلیل پارامترها:

تعداد کل پارامترهای مورد استفاده در محاسبات Self-Attention به صورت زیر تقسیم می‌شود:

1. ماتریس‌های Q، Kو V هر کدام دارای N×D عنصر هستند، زیرا از ماتریس‌های وزنی یاد گرفته‌شده اعمال‌شده روی دنباله ورودی تولید می‌شوند.
   - در مجموع، این سه ماتریس 3×N×D پارامتر ایجاد می‌کنند.
2. هنگام محاسبه $Q \cdot K^T$، N^2عملیات ضرب نقطه‌ای انجام می‌شود که هر کدام نیاز به D ضرب و جمع دارند.
   - این مرحله $N^2 \times D$ عملیات به خود اختصاص می‌دهد.
3. ضرب ماتریس امتیازات توجه (N×N) در V (N×D) نیاز به N2×Dعملیات اضافی دارد.

توضیحات بالا پیچیدگی محاسبات را برای یک گام (Attention Head) مشخص می‌کند. حال اگر بخواهیم تمام پارامترهای مدل را در نظر بگیریم:

1. هر (Attention Head) شامل سه ماتریس یادگیری‌پذیر استW_Q ، W_K، و W_V.
   - هرکدام از این ماتریس‌ها ابعاد D×D دارند.
   - تعداد کل پارامترها در این سه ماتریس $3 \times D^2$ است.

2. حال اگر $N \times D$ توکن را پردازش کنیم و بخواهیم محاسبات مربوط به هر توکن را برای همه توکن‌های دنباله انجام دهیم:

   - D^2 پارامترهای ماتریس‌ها ضرب در N^2 محاسبات برای توجه زوجی بین تمام توکن‌ها می‌شود.

در نتیجه:

$$\text{Total Complexity} = O(N^2 D^2)$$

قسمت 3)

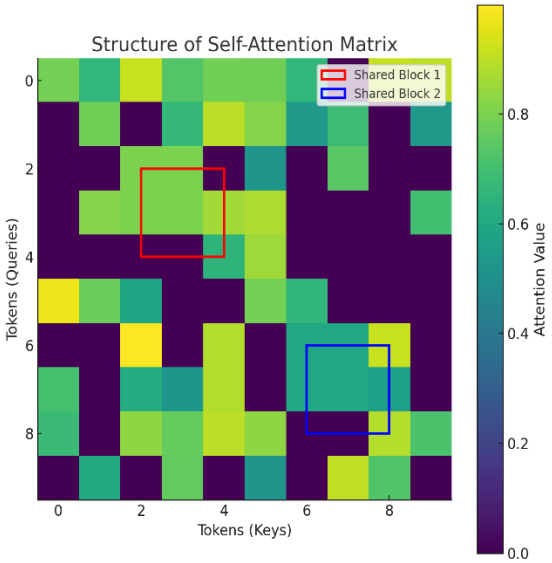
1.اشتراک پارامترها در ماتریسSelf-Attention

- در مکانیزمSelf-Attention ، بلوک‌های ماتریس تعامل ($Q \cdot K^T$) اغلب پارامترهای مشترک دارند:
  - پارامترهای مشترک به دلیل وجود وزن‌های یادگیری‌پذیر (W_Q, W_K, W_V) برای همه توکن‌ها هستند.
  - به عنوان مثال، اگر دو توکن وزن‌های توجه مشابهی برای سایر توکن‌ها داشته باشند، بلوک‌های مربوط به تعامل آن‌ها در ماتریس توجه یکسان خواهند بود.

2.ساختار ماتریسSparse

- در یک ماتریس:Sparse
  - بسیاری از عناصر به دلیل عملیات Softmax و محدودیت مقادیر کوچک به صفر نزدیک یا دقیقاً صفر هستند.
  - این ساختار خلوت باعث کاهش حجم محاسبات می‌شود، زیرا عملیات تنها بر روی عناصر غیر صفر انجام می‌شود.
- در ساختار ماتریس:
  - بلوک‌هایی که تعامل قوی دارند (وزن توجه بالا)، مقادیر غیر صفر خواهند داشت.
  - بلوک‌های دیگر که تعامل کمی دارند یا اهمیتی ندارند، به صفر تبدیل می‌شوند.

3. نمودار ساختار ماتریس



این نمودار ساختار یک ماتریس Self-Attention را نمایش می‌دهد.

- هر عنصر در ماتریس نشان‌دهنده تعامل (وزن توجه) بین یک جفت توکن است.
- بلوک‌های قرمز و آبی نشان‌دهنده بخش‌هایی از ماتریس هستند که پارامترهای مشترک دارند.
- عناصر نزدیک به صفر (نمایش داده‌شده با رنگ‌های تیره) ناشی از تابع Softmax هستند، که خلوت بودن (Sparsity) ماتریس را نشان می‌دهند.

قسمت 4)

اثر حذفPositional Encoding

1. بدون:Positional Encoding

- اگر Positional Encodingحذف شود، مدل قادر به تشخیص موقعیت کلمات در دنباله ورودی نخواهد بود.
- در نتیجه، خروجی لایه Attention تنها بر اساس روابط محتوایی (Semantic) بین کلمات محاسبه می‌شود، بدون در نظر گرفتن ترتیب آن‌ها.
- به عنوان مثال:
  - دنباله ورودی"The cat sat on the mat" :
  - ترتیب تغییر یافته"Mat the on sat cat the" :
  - خروجی برای هر دو دنباله مشابه خواهد بود، زیرا مدل موقعیت کلمات را در نظر نمی‌گیرد.

2. ترتیب مجدد دنباله‌ها:

- اگر ترتیب دنباله‌ها تغییر کند، خروجی به ترتیب دنباله جدید خواهد بود. این به این معناست که مدل همچنان خروجی معادل تولید می‌کند، اما این خروجی دیگر ترتیب معنایی اصلی را حفظ نمی‌کند.

اثر حذف بردارهای ورودی

1. اگر بردارهای ورودی حذف شوند:

- بدون ورودی‌های معنایی(Semantic Embeddings) ، مدل هیچ اطلاعاتی درباره کلمات یا توکن‌های ورودی ندارد.
- در این حالت، خروجی Attention به طور کامل بی‌معنی خواهد بود، زیرا هیچ محتوای معنایی برای محاسبه ارتباط وجود ندارد.
- تمام توجه‌ها و خروجی‌ها ممکن است به صورت تصادفی یا صفر تولید شوند.

2. وابستگی به:Positional Encoding

- حتی اگر Positional Encoding به تنهایی باقی بماند، بدون اطلاعات معنایی بردارهای ورودی، مدل نمی‌تواند تصمیم‌گیری مناسبی درباره ترتیب و معنای کلمات داشته باشد.

سوال 4

Encoder:

- Input Shape: (BatchSize,SequenceLength,HiddenDimension)=(32,2048,768)
- Key, Query, Value Sizes:
  Each size = (HiddenDimension,HiddenDimension)=(768,768)
- Self-Attention Layer in Encoder Block:
  - Input Shape: (32,2048,768)
  - Output Shape: (32,2048,768)
- Feed-Forward Network in Encoder Block:
  - Input Shape: (32,2048,768)
  - Intermediate Output: (32,2048,384)
  - Final Output: (32,2048,768)
- Output Dimensions per Encoder Block:
  (32,2048,768)
- Total Number of Encoder Blocks: 8
  - Final Output Shape after Encoder:
    (32,2048,768)

Decoder:

- Input Shape: (BatchSize,TargetSequenceLength,HiddenDimension)=(32,1024,768)
- Key, Query, Value Sizes:
  Each size = (HiddenDimension,HiddenDimension)=(768,768)
- Self-Attention Layer in Decoder Block:
  - Input Shape: (32,1024,768)
  - Output Shape: (32,1024,768)
- Encoder-Decoder Attention in Decoder Block:
  - Input Shape: (32,1024,768)

- Output Shape: (32,1024,768)

- Feed-Forward Network in Decoder Block:
  - Input Shape: (32,1024,768)
  - Intermediate Output: (32,1024,384)
  - Final Output: (32,1024,768)
- Output Dimensions per Decoder Block: (32,1024,768)
- Total Number of Decoder Blocks: 12
  - Final Output Shape after Decoder: (32,1024,768)

ب)

Parameter Calculation for the Architecture:

1. Embedding Layers (Encoder + Decoder):

- Embedding Matrix Dimensions:

  $\text{VocabSize} \times \text{HiddenSize} = 30{,}000 \times 768 = 23{,}040{,}000$

  For both encoder and decoder:

  $$23{,}040{,}000 \times 2 = 46{,}080{,}000 \text{ parameters}$$

2. Encoder Blocks:

Each encoder block has two main components:

- Self-Attention Layer:
  - Query, Key, Value Matrices: $\text{HiddenSize} \times \text{HiddenSize} = 768 \times 768 = 589,$

    Total for all three matrices:

    $$589{,}824 \times 3 = 1{,}769{,}472 \text{ parameters}$$

    Output projection matrix: $\text{HiddenSize} \times \text{HiddenSize} = 768 \times 768 = 589{,}824$ parameters

Total Self-Attention Parameters:

1,769,472+589,824=2,359,296 parameters

- Feed-Forward Network (FFN):
  - First layer:

    HiddenSize×FFN_Size=768×384=294,912
  - Second layer: FFN_Size×HiddenSize=384×768=294,912
  - Total FFN Parameters:

    294,912+294,912=589,824
- Total Parameters per Encoder Block:

  2,359,296+589,824=2,949,120 parameters
- Total for 8 Encoder Blocks:

  2,949,120×8=23,592,960 parameters

3. Decoder Blocks:

Each decoder block has three main components:

- Self-Attention Layer:

  Same as Encoder:2,359,296 parameters
- Encoder-Decoder Attention Layer:

  Same asSelf−Attention:2,359,296 parameters
- Feed-Forward Network (FFN):

  Same as Encoder:589,824 parameters
- Total Parameters per Decoder Block:

  2,359,296+2,359,296+589,824=5,308,416 parameters
- Total for 12 Decoder Blocks:

  5,308,416×12=63,700,992 parameters

4. Output Layer:

- Fully connected layer mapping hidden size to vocabulary size: HiddenSize×VocabSize=768×30,000=23,040,000 parameters

Total Parameters:

1. Embedding Layers: 46,080,00046,080,000
2. Encoder Blocks: 23,592,96023,592,960
3. Decoder Blocks: 63,700,99263,700,992
4. Output Layer: 23,040,00023,040,000

Grand Total:

46,080,000+23,592,960+63,700,992+23,040,000=156,413,952 parameters

ج)

تسک: ترجمه جمله "I love programming." از انگلیسی به فارسی

1.ورودی به انکودر:

- جمله ورودی"I love programming." :
- تبدیل جمله به توکن‌ها (اندیس‌ها)[350,200,101] :نمایانگر"programming" ,"love"، "I"
- شکل ورودی به لایه جاسازی:(Embedding Layer)
  (BatchSize,SequenceLength)=(3,1)
- بعد از جاسازی :هر توکن به برداری با بُعد 768تبدیل می‌شود :
  (BatchSize,SequenceLength,HiddenDimension)=(768,3,1)

2.کدگذاری موقعیت:(Positional Encoding)

- به هر توکن ورودی، اطلاعات مربوط به موقعیت آن در جمله اضافه می‌شود(مثلاً "I" اولین کلمه است)

3.عبور از بلاک‌های انکودر (8 بلاک):

در هر بلاک انکودر:

1. لایه توجه چندسری:(Multi-Head Self-Attention)
   - ورودی(768,3,1) :
   - سه ماتریس Query (Q)، Key (K)و Value (V)ایجاد می‌شود :
     - شکل هر ماتریس(768,3,1) :
   - مکانیزم توجه محاسبه می‌شود Attention(Q,K,V)
   - خروجی: همان شکل ورودی (768,3,1)(1,3,768)
2. عملیات نرمال‌سازی:(Add & Norm)
   - خروجی توجه به ورودی اولیه اضافه می‌شود و نرمال‌سازی انجام می‌گیرد.
   - نتیجه(768,3,1):
3. شبکه پیش‌خور:(Feed-Forward Network)
   - لایه اول (افزایش بُعد):
     ورودی : (768,3,1) ,خروجی: (384,3,1)
   - لایه دوم (کاهش بُعد):
     ورودی : (384,3,1) ,خروجی: (768,3,1)
4. عملیات نرمال‌سازی دوم:(Add & Norm)
   - خروجی شبکه پیش‌خور به نتیجه توجه اضافه می‌شود و نرمال‌سازی انجام می‌گیرد.
   - خروجی(768,3,1) :

4. خروجی نهایی انکودر:

- بعد از عبور از 8 بلاک انکودر، شکل نهایی خروجی :
(BatchSize,SequenceLength,HiddenDimension)=(768,3,1)
- این خروجی شامل اطلاعات غنی‌شده درباره روابط و معانی کلمات در جمله انگلیسی است که به دیکودر ارسال می‌شود.

1. ورودی به دیکودر:

- دیکودر ترجمه را به صورت مرحله‌به‌مرحله تولید می‌کند.
- ورودی اولیه به دیکودر، تنها توکن شروع (START) است.
مثال:

Input Tokens: [1]

- این توکن از طریق لایه جاسازی (Embedding Layer) به یک بردار 768-بعدی تبدیل می‌شود :
شکل ورودی به دیکودر :(768,1,1)=(BatchSize,TargetSequenceLength,HiddenDimension)

2.عبور از بلاک‌های دیکودر (12 بلاک):

در هر بلاک دیکودر:

1. لایه توجه چندسری:(Self-Attention)
   - این لایه تنها روی دنباله خروجی فعلی دیکودر (ترجمه تولیدشده تاکنون) تمرکز می‌کند.
   - سه ماتریس Query (Q)، Key (K)و Value (V)ایجاد می‌شود :
     - شکل هر ماتریس (768,1,1)(768,1,1) برای مرحله اول.
   - مکانیزم توجه محاسبه می‌شود Attention(Q,K,V)
   - :(Masked Attention) برای اطمینان از اینکه دیکودر تنها به خروجی‌های قبلی خود دسترسی دارد، از ماسک استفاده می‌شود.
   - خروجی: همان شکل ورودی (768,1,1)

2. عملیات نرمال‌سازی اول:(Add & Norm)
   - خروجی توجه به ورودی اولیه اضافه می‌شود و نرمال‌سازی انجام می‌گیرد (768,1,1)
3. توجه انکودر-دیکودر:(Encoder-Decoder Attention)
   - در این مرحله، دیکودر روی خروجی انکودر تمرکز می‌کند.
   - ماتریس‌های Query (Q)، Key (K)و Value (V)به ترتیب از دیکودر و انکودر ایجاد می‌شوند :
     - Qاز دیکودر(768,1,1) :
     - Kو V از انکودر(768,3,1) :
   - مکانیزم توجه محاسبه می‌شود Attention(Q,K,V) :
   - خروجی(768,1,1)
4. عملیات نرمال‌سازی دوم:(Add & Norm)
   - خروجی توجه انکودر-دیکودر به خروجی قبلی اضافه شده و نرمال‌سازی می‌شود (768,1,1) :
5. شبکه پیش‌خور:(Feed-Forward Network)
   - لایه اول (افزایش بُعد) (768,1,1)→(384,1,1) :
   - لایه دوم (کاهش بُعد) (384,1,1)→(768,1,1) :
6. عملیات نرمال‌سازی سوم:(Add & Norm)
   - خروجی شبکه پیش‌خور به نتیجه قبلی اضافه شده و نرمال‌سازی می‌شود (768,1,1) :

.3پیش‌بینی کلمه خروجی:

- خروجی دیکودر به یک لایه کاملاً متصل (Fully Connected Layer) داده می‌شود که اندازه آن برابر با تعداد کلمات واژگان (Vocabulary Size) است:
- VocabularySize → HiddenDimension =30,000→768
- خروجی این لایه احتمال هر کلمه در واژگان را نشان می‌دهد. کلمه با بیشترین احتمال انتخاب می‌شود :

Predicted Token: [7] ("من" در زبان فارسی)

.4تکرار مراحل برای کلمه بعدی:

- توکن پیش‌بینی‌شده (مثلاً [7]) به دنباله ورودی دیکودر اضافه می‌شود.
- دیکودر مجدداً مراحل بالا را تکرار می‌کند تا کلمه بعدی پیش‌بینی شود :
  - گام 1" → [1] :من"
  - گام 2" → [7,1] :عاشق"
  - گام 3" → [50,7,1] :برنامه‌نویسی"
  - گام 4<END> → [200,50,7,1] :

.5خروجی نهایی دیکودر:

- خروجی نهایی دیکودر، ترجمه کامل جمله است:
  "من عاشق برنامه‌نویسی هستم".

سوال 5

الف)

حذف مرحلهٔ «پیش‌بینی جملهٔ بعدی (NSP) «از پیش‌پردازش BERT در مطالعات مختلف (مانند معماریRoBERT)نشان داده است که می‌تواند در بسیاری از وظایف پایین‌دستی (Downstream Tasks) عملکرد مدل را بهبود دهد. دلایل اصلی چنین بهبودی را می‌توان به‌صورت زیر خلاصه کرد:

1. سادگی و محدودیت:NSP

در BERT اصلی، NSPبا یک کار ساده (تشخیص «آیا جمله دوم از نظر توالی طبیعی به جملهٔ اول متصل است یا نه») آموزش می‌بیند. این کار، مدل را وادار می‌کند تا نوعی تشخیص «تصادفی‌بودن یا نبودن» بین دو جمله انجام دهد که الزاماً در بسیاری از وظایف زبانی پیچیده مفید نیست. بنابراین بخشی از ظرفیت مدل صرف یادگیری الگویی می‌شود که در بسیاری از کاربردهای عملی نقش کلیدی ندارد.

2. کاهش نویز و تمرکز بیشتر بر بافت کلمه‌ها:

در رویکردNSP ، جفت‌جمله‌های ساختگی (Random) ممکن است باعث ایجاد سیگنال‌های آموزشی پرت (Noise) شود و مدل برای تشخیص همین سیگنال از داده‌های ماسک‌شده (MLM) منحرف شود. حذف NSP به مدل اجازه می‌دهد توجه و ظرفیتش را به شکل متمرکزتری روی پیش‌بینی توکن‌های ماسک‌شده بر اساس بافت (Context) صرف کند و در نتیجه نمایش (Representation) کلی بهتری از زبان به دست آورد.

3. الگوی نامناسب اتصال جمله‌ها:

روش NSP معمولاً با انتخاب جمله‌هایی از همان سند یا از اسناد متفاوت صورت می‌گیرد. اما در داده‌های واقعی، نحوهٔ گذار از یک جمله به جملهٔ بعدی می‌تواند بسیار متنوع‌تر از «متصل/نامتصل» ساده باشد. به عبارت دیگر، بسیاری از متون دنیای واقعی (مثل پاراگراف‌های طولانی، مباحث چندبخشی و غیره) به روش NSP پوشش مناسبی داده نمی‌شوند. حذف NSP باعث می‌شود مدل درگیر چنین الگوی محدودی نشود.

4. بهبود عملکرد در وظایفی که نیازمند انسجام درون‌جمله‌ای هستند:

بسیاری از وظایف پایین‌دستی (مانند تحلیل احساس، طبقه‌بندی جمله، استخراج موجودیت‌ها و غیره) بیش از آنکه به رابطهٔ میان دو جمله احتیاج داشته باشند، نیازمند فهم عمیق‌تر محتوای تک‌جمله یا مجموع جملات هستند. حذف NSPبه مدل امکان می‌دهد تا از لایه‌های بیشتری برای فهم بهتر کلمات و جملات استفاده کند و در نتیجه، در این وظایف ساده‌تر به دقت بالاتری برسد.

5. یافته‌های تجربی:

در معماری RoBERTa و برخی مدل‌های دیگر، با حذف NSP مشاهده شد که مدل در اکثر وظایف GLUE و سایر

بنچمارک‌ها نتایج بهتری کسب می‌کند. این یافته‌های تجربی نشان می‌دهد که هدف NSP آن‌چنان که تصور می‌شد برای فراگیری روابط میان‌جمله‌ای مؤثر نبوده و حتی می‌تواند موجب هدررفت ظرفیت مدل شود.

در مجموع، اگرچه ایدهٔ NSP در ابتدا برای بهبود درک روابط سطح بالا بین جملات در نظر گرفته شده بود، اما در عمل، شکل ساده و دوحالتی آن (پیوسته بودن یا نبودن جملات) کمک چندانی به نمایش بهتر زبان نمی‌کند و حتی ممکن است مانع شود که مدل به‌صورت عمیق‌تری بر خودِ کلمات و بافت درونی جملات تمرکز کند. ازاین‌رو، حذف NSP می‌تواند در کاربردهای واقعی، به‌ویژه وظایف پایین‌دستی که بیشتر بر درک محتوای تک‌جمله یا چند جملهٔ پیوسته متمرکزند، عملکرد را بهبود بخشد.

ب)

برای رسیدن به «درک کلی از جمله» در هنگام تنظیم دقیق (Fine-tuning) مدل‌هایی مانند BERT یا RoBERTa و به‌کارگیری آن‌ها در تسک‌های پایین‌دستی (مثلاً طبقه‌بندی جملات، تحلیل احساس، بازیابی متون و غیره)، معمولاً نیاز داریم که ورودی (یک جمله یا عبارت) را به یک بردار نهایی تبدیل کنیم؛ برداری که بتواند به‌عنوان نماینده یا چکیدهٔ معنایی کل جملهٔ ورودی عمل کند. به این فرایند اغلب «نمایش جمله (Sentence Representation) «گفته می‌شود. در ادامه، چند روش متداول برای دست‌یابی به چنین نمایشی توضیح داده شده است:

1. استفاده از بردار([CLS] روش پیش‌فرضBERT)

در معماریBERT ، توکن ویژه‌ای به نام [CLS]در ابتدای دنباله قرار داده می‌شود. در خروجی آخرین لایهٔ ترانسفورمر (یا یکی از لایه‌های آخر بسته به طراحی)، بردار متناظر با [CLS]به‌نوعی نقش «خلاصه‌کننده» یا «نمایندهٔ کل دنباله» را دارد. در بسیاری از سناریوهای طبقه‌بندی، از همین بردار به عنوان ورودی یک شبکهٔ کوچک (مثلاً یک لایهDense) استفاده می‌شود تا پیش‌بینی نهایی انجام گیرد.

- **مزیت** :پیاده‌سازی ساده و سازگاری با طراحی اولیه.BERT
- **چالش** :لزوماً تضمین نمی‌شود که بردار [CLS]همیشه «بهترین» یا «تنها» راه برای نمایش کل جمله باشد. در عمل، گاهی مشاهده می‌شود که روش‌های دیگر (مثلاً تجمیع همه توکن‌ها) نتایج بهتری در برخی تسک‌ها ارائه می‌کنند.

2. استفاده از(Pooling تجمیع بردارهای همه توکن‌ها)

در این روش، به‌جای اتکا به بردار خاصی مانند [CLS]، تمام توکن‌های خروجی از آخرین لایه یا چند لایهٔ آخر مدل را در نظر می‌گیریم. سپس با نوعی عملگر pooling (مانند میانگین، حداکثر یا ترکیبی از آن‌ها) یک بردار واحد را به دست می‌آوریم. به این ترتیب، نمود هر توکن (و به‌تبع آن بخش‌های مختلف جمله) در نمایش نهایی لحاظ می‌شود.

- **Average Pooling**:

$$Representation = \frac{1}{N}\sum_{i=1}^{n}\frac{1}{n}h_i$$

.

- **Max Pooling**:

$$Representation[d] = \max_{1=1\dots n}(h_i[d])$$

که $h_i[d]$ مؤلفهٔ dام بردار نهفتگی توکن iاست و در پایان، ماکس در هر بعد گرفته می‌شود.

- **مزیت:**
  - توجه به تمام توکن‌ها و در نتیجه دریافت اطلاعات جامع‌تر از جمله.
  - سادگی پیاده‌سازی(فقط یک لایه ساده‌ی)Poolingو قابل استفاده روی هر لایه از خروجی (برای مثال آخرین لایه یا میانگین چند لایه).
- **چالش:**
  - برخی مواقع ممکن است نویزهای موجود در بخشی از جمله بر تجمیع کلی غلبه کنند(خصوصاً در روش Max Pooling.)
  - انتخاب مناسب نوع Pooling و لایه/لایه‌هایی که از آن Pool می‌گیریم، نیازمند آزمون‌وخطا است.
  -

3 .مدل‌های تخصصی برای نمایش جمله (مانندSentence-BERT)

Sentence-BERTیا به اختصار **SBERT**، معماری‌ای مبتنی بر BERT است که مشخصاً برای نمایش سطح جمله طراحی شده و اغلب به‌صورت Siamese Network (دوقلو) آموزش می‌بیند. در این روش:

1. دو جمله (یا بیشتر) به‌صورت موازی از مدل گذر داده می‌شوند.

2. در انتهای هر شاخه، یک لایه) Pooling معمولاً Average Pooling یا (CLS Pooling روی خروجی آخرین لایه اعمال می‌شود تا بردار نهایی جمله به دست آید.
3. برای آموزش، از زیان‌های مبتنی بر شباهت بردارها (مثلاً کازین شباهت یا کنتراستی) استفاده می‌شود تا مدل بیاموزد جمله‌های «مشابه» بردارهای نهایی نزدیکی داشته باشند و جمله‌های «نامشابه» بردارهای دورتری داشته باشند.

- **مزیت:**
  - آموزش صریحاً بر اساس نمایش سطح جمله انجام می‌شود، بنابراین در وظایف متناظر (مثلاً بازیابی مبتنی بر شباهت جمله، خوشه‌بندی جملات، یا پرسش-پاسخ‌های کوتاه) کارایی بالاتری دارد.
  - سرعت بالاتر در انجام مقایسهٔ جملات (چون هر جمله یک بار به یک بردار تبدیل می‌شود و نیازی به pairwise comparisonدر خود مدل نیست)
- **چالش:**
  - راه‌اندازی و آموزش اولیه آن نیازمند داده‌های مناسب (مثلاً جفت‌جملات مشابه/نامشابه) و کمی پیچیدگی بیشتر نسبت به استفادهٔ مستقیم از BERT دارد.

4 .لایه‌های اضافه و معماری‌های ترکیبی

در برخی موارد، برای به‌دست‌آوردن نمایش بهتر سطح جمله، یک لایه یا بلاک اضافی به خروجی مدل اضافه می‌شود که کارش «تجمیع و تلفیق بردارهای همه توکن‌ها»ست. این بلاک می‌تواند شامل مکانیسم توجه (Attention) یا مکانیزم‌های پیشرفته‌تر باشد تا کاستی‌های pooling ساده برطرف شود. به عنوان مثال، ممکن است:

1. ابتدا با یک یا چند لایهٔ ترانسفورمر اضافی، بردارهای توکن‌ها دوباره با هم تعامل کنند.
2. در انتها از یک لایهٔ pooling (مثلاًMean Pooling) یا حتی مکانیزمی شبیه attention-pooling استفاده شود.

این رویکرد گاهی در معماری‌های پیشرفته‌تر دیده می‌شود تا از تمام اطلاعات در سطوح مختلف مدل بیشترین بهره گرفته شود.

جمع‌بندی

- درک کلی از جمله را می‌توان با استفاده از بردار خاص [CLS](روش پیش‌فرض)، یا انواع Pooling (میانگین، حداکثر و ...)، یا مدل‌های تخصصی نظیر Sentence-BERT به دست آورد.
- انتخاب بهترین راهکار، وابسته به نوع تسک پایین‌دستی است. برای مثال، در تسک‌های «تشخیص شباهت جملات» یا «جستجوی معنایی» روش‌هایی مثل SBERT معمولاً نتایج بهتری می‌دهند. اما در تسک‌های ساده‌تر (مثل طبقه‌بندی دودویی جملات) گاهی همان بردار [CLS]یا یک Pooling ساده، به‌اندازه کافی خوب عمل می‌کند.
- **تنظیم دقیق (Fine-tuning)** نیز نقش مهمی دارد. حتی یک روش ساده مانند بردار [CLS]هم در صورتی که به‌درستی روی داده‌های تسک هدف (Task-specific data) تنظیم (Fine-tune) شود، می‌تواند عملکرد قابل قبولی داشته باشد. اما گاه با کمی تلاش بیشتر در طراحی مرحله نمایندگی جمله (مثلاً با استفاده از Pooling یا روش‌های تناوبی)، می‌توان بهبود محسوسی در عملکرد نهایی به دست آورد.

در نتیجه، برای آنکه مدلی مانند BERT یا RoBERTa بتواند «درک کلی از جمله» ارائه دهد، لازم است یا از مکانیزم‌های مناسب در لایه خروجی (نظیرPooling) استفاده کنیم یا از مدل‌هایی بهره بگیریم که به‌طور اختصاصی برای تولید بردار جملات آموزش دیده‌اند (همچونSentence-BERT).