دانشکده‌ی مهندسی کامپیوتر

یادگیری ماشین

زمستان ۱۴۰۳

علیرضا میرکنی - ۴۰۱۱۰۶۶۱۷

تمرین پنجم

پاسخ مسئله‌ی ۱.

الف)

- با توجه به اطلاعات داده‌شده، در ابتدا تعداد positive pair ها را به دست می‌آوریم:

$$\#\text{ of positive pairs} = ۲ + ۳ + ۶ \times ۴ + ۲ + ۳ = ۳۴$$

دقت کنید که با قرارگیری هرکدام از کلمات اول و دهم به‌عنوان کلمه مرکزی ۲ عدد positive pair، و با قرارگیری هرکدام از کلمات دوم و نهم به عنوان کلمه مرکزی ۳ عدد positive pair ایجاد خواهد شد. برای هر کدام از کلمات دیگر نیز ۴ عدد positive pair خواهیم داشت. حال دقت کنید که برای هر positive pair، ۵ عدد negative sample داریم و در نتیجه تعداد کل training sample ها از رابطه زیر به‌دست می‌آید:

$$\#\text{ of training samples } = ۶ \times \#\text{ of positive pairs} = ۲۰۴$$

- اگر target word در جایگاه i ام corpus قرار گرفته‌باشد (کلمه i ام corpus باشد) و طول corpus را با T نمایش دهیم، آنگاه تعداد input word های مورد استفاده برای پیش‌بینی آن از رابطه زیر محاسبه می‌شود:

$$\#\text{ of input words} = \min(i - ۱, ۴) + \min(T - i, ۴)$$

- طبق اطلاعات مسئله، d_{model} برابر با ابعاد embedding ورودی و h برابر تعداد attention head ها است. پس ابعاد هر head برابر است با:

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

در نتیجه ماتریس‌های projection یعنی W_i^Q، W_i^K، و W_i^V برای $i \in \{۱, \dots, h\}$ دارای ابعاد زیر خواهند بود:

$$W_i^Q, W_i^K, W_i^V \in \mathbb{R}^{d_{model} \times d_k}$$

این تنظیمات باعث می‌شوند که عملیات محاسباتی روی بخش‌های کوچک‌تر از فضای بُرداری اصلی انجام شوند که باعث می‌شود محاسبات ماتریسی سریع‌تر انجام شوند، قابلیت پردازش موازی بهبود پیدا کند و کاهش حافظه مورد نیاز برای ذخیره ماتریس‌های projection اتفاق بیفتد.

ب)

- **صحیح**. در معماری اصلی Transformer، از fixed sinusoidal positional encodings استفاده می‌شود که به صورت از پیش تعریف شده و بدون نیاز به یادگیری اعمال می‌شوند. این کدگذاری‌های موقعیتی به منظور اضافه کردن اطلاعات ترتیبی به ورودی‌ها به کار می‌روند، زیرا مکانیزم Self-Attention ذاتاً نسبت به ترتیب توکن‌ها Permutation-Equivariant است.

  در مقابل، مدل BERT از learnable positional embeddings استفاده می‌کند. این embeddings به صورت پارامترهای قابل یادگیری تعریف می‌شوند و طی فرایند training بهینه‌سازی می‌شوند. در نتیجه، برخلاف Transformer که از کدگذاری‌های سینوسی ثابت بهره می‌گیرد، BERT اطلاعات موقعیتی را به صورت پارامترهای مستقل و قابل یادگیری در طول آموزش به دست می‌آورد.

- **غلط**. افزایش تعداد attention head ها در مدل Transformer همیشه باعث بهبود عملکرد نمی‌شود. این افزایش تعداد می‌تواند به استخراج ویژگی‌های متفاوت کمک کند، اما معایب زیر را نیز ممکن است ایجاد کند:
  - **هزینه محاسباتی بالاتر:** افزایش تعداد head ها، نیاز به محاسبات بیشتری دارد.
  - **خطر بیش‌برازش (Overfitting):** مدل ممکن است به داده‌های آموزشی وابستگی زیادی پیدا کند و عملکرد آن روی داده‌های جدید کاهش یابد.
  - **نیاز به حافظه بیشتر:** تعداد بیشتر head ها به معنای ذخیره ماتریس‌های بیشتر است که نیاز به حافظه بیشتری دارد.

پاسخ مسئله‌ی ۲.

الف) به‌سادگی می‌توان نوشت:

$$
\begin{aligned}
-\frac{\partial \log P(w_o|w_t)}{\partial \mathbf{v}_{w_o}} &= -\frac{\partial \log P(w_o|w_t)}{\partial P(w_o|w_t)} \cdot \frac{\partial P(w_o|w_t)}{\partial \mathbf{v}_{w_o}} \\
&= -\frac{۱}{P(w_o|w_t)} \cdot \frac{\partial \left(\frac{\exp(\mathbf{u}_{w_t}^\top \mathbf{v}_{w_o})}{\sum_{k\in\mathbf{v}} \exp(\mathbf{u}_{w_t}^\top \mathbf{v}_k)} \right)}{\partial \mathbf{v}_{w_o}} \\
&= -\frac{۱}{P(w_o|w_t)} \cdot \frac{\mathbf{u}_{w_t} \exp(\mathbf{u}_{w_t}^\top \mathbf{v}_{w_o}) \sum_{k\in\mathbf{v}} \exp(\mathbf{u}_{w_t}^\top \mathbf{v}_k) - \mathbf{u}_{w_t} \exp(\mathbf{u}_{w_t}^\top \mathbf{v}_{w_o})^۲}{\left(\sum_{k\in\mathbf{v}} \exp(\mathbf{u}_{w_t}^\top \mathbf{v}_k)\right)^۲} \\
&= -\frac{۱}{P(w_o|w_t)} \cdot \mathbf{u}_{w_t} \left(\frac{\exp(\mathbf{u}_{w_t}^\top \mathbf{v}_{w_o})}{(\sum_{k\in\mathbf{v}} \exp(\mathbf{u}_{w_t}^\top \mathbf{v}_k)} - \left(\frac{\exp(\mathbf{u}_{w_t}^\top \mathbf{v}_{w_o})}{\sum_{k\in\mathbf{v}} \exp(\mathbf{u}_{w_t}^\top \mathbf{v}_k)} \right)^۲ \right) \\
&= -\frac{۱}{P(w_o|w_t)} \cdot \mathbf{u}_{w_t} \left(P(w_o|w_t) - P(w_o|w_t)^۲ \right) \\
&= \mathbf{u}_{w_t} (P(w_o|w_t) - ۱)
\end{aligned}
$$

ب) در جدول زیر اثر مقادیر مختلف c مقایسه شده‌اند.

اندازه پنجره (c)	تاثیر روی وابستگی‌های معنایی	پیچیدگی محاسباتی
۱	مدل تنها به همسایه‌های بسیار نزدیک توجه می‌کند و روی روابط محلی تمرکز دارد. این تنظیم می‌تواند برای داده‌هایی با ساختار محلی قوی مناسب باشد. با این حال، روابط طولانی‌مدت نادیده گرفته می‌شوند.	محاسبات سریع‌تر و بهینه‌تر انجام می‌شود زیرا تعداد جفت‌های مثبت و منفی کمتر است و حافظه کمتری نیاز دارد.
۵	یک تعادل بین روابط محلی و جهانی ایجاد می‌کند. مدل قادر به یادگیری ارتباطات معنی‌دار بین کلمات نزدیک و دورتر است. این اندازه معمولاً برای داده‌های عمومی مناسب است.	هزینه محاسباتی متوسط است. حافظه و زمان پردازش بیشتر از $c = ۱$ اما کمتر از $c = ۱۰۰$ است.
۱۰۰	تمرکز بر روابط جهانی و طولانی‌مدت دارد. این اندازه برای داده‌هایی که روابط دور برد دارند (مانند متون طولانی یا مقالات علمی) مناسب است، اما نویز بیشتری به مدل اضافه می‌کند.	نیاز به حافظه و زمان پردازش بالاتر دارد. تعداد نمونه‌های آموزشی به‌طور چشمگیری افزایش می‌یابد که منجر به بار محاسباتی زیاد می‌شود.

جدول ۱: مقایسه تاثیر اندازه پنجره در مدل skip-gram

پاسخ مسئله‌ی ۳.

۱. : فرض کنید دنباله ورودی شامل بردارهای کلمه به شکل زیر است:

$$X = \{x_۱, x_۲, \ldots, x_N\}$$

که N طول دنباله و D ابعاد هر بردار کلمه است. مکانیزم خودتوجه وزن توجه را برای هر کلمه در دنباله بر اساس روابط بین تمام کلمات در دنباله محاسبه می‌کند. عملیات خودتوجه به‌صورت زیر فرمول‌بندی می‌شود:

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

برای نمایش این عملیات به‌عنوان یک شبکه کاملاً متصل در قالب یک ماتریس، می‌توان بردارهای کلمه x_i را در کنار هم قرار داد تا یک ماتریس X' با ابعاد $N \times D$ تشکیل شود که هر ردیف آن یک بردار کلمه را نشان می‌دهد. در این صورت خواهیم داشت:

$$Q = X'W^Q, \quad K = X'W^K, \quad V = X'W^V$$

حال، ماتریس توجه A را می‌توان به‌صورت زیر (با استفاده از ماتریس X' و ترانهاده آن) محاسبه کرد:

$$A = X'(X')^\mathsf{T}$$

این ضرب ماتریسی، روابط بین تمام بردارهای کلمه در دنباله ورودی را نشان می‌دهد. درنهایت، برای نگاشت توالی کامل ورودی به یک بردار خروجی با همان ابعاد، می‌توان ضرب ماتریس دیگری را بین A و یک ماتریس وزن قابل یادگیری W به‌دست آورد:

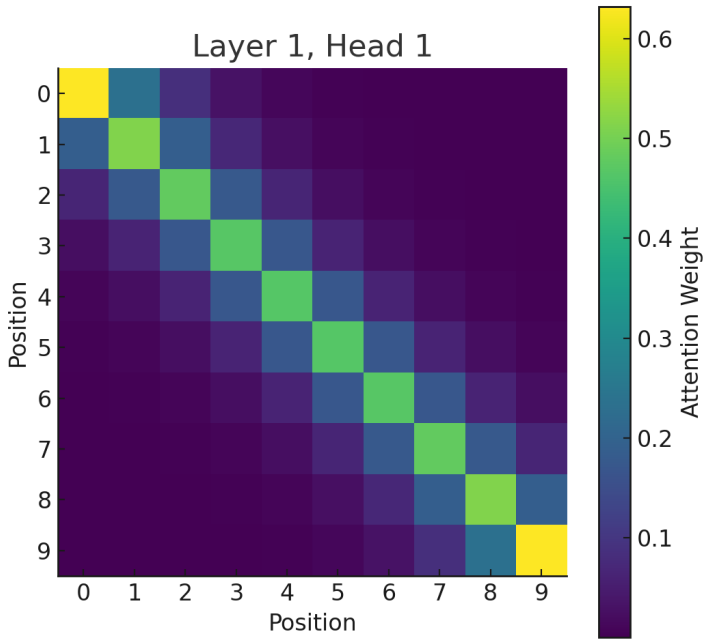
$$\text{Output} = \text{Softmax}(AW)$$

عبارت بالا خروجی یک شبکه تمام‌متصل با ماتریس وزن W و تابع فعال‌سازی Softmax است.

۲. دقت کنید که در شبکه به دست‌آمده در قسمت قبل، ماتریس W به شکل $N \times N$ است (با فرض حفظ ابعاد یکسان) و شامل $O(N^۲)$ پارامتر می‌باشد. این نمایش شبکه کاملاً متصل، دارای $O(N^۲D^۲)$ پارامتر خواهد بود، زیرا $N^۲$ درایه در ماتریس توجه A وجود دارند و هر کدام آن‌ها حاصل ضرب بردارهای D بعدی است.

۳. در مکانیزم خودتوجه، هر کلمه در دنباله ورودی به همه کلمات دیگر توجه می‌کند، اما تمام توجه‌ها به یک اندازه مهم نیستند. برای دستیابی به این هدف، وزن توجه با استفاده از یک تابع Softmax محاسبه می‌شود که منجر به توزیع توجه پراکنده می‌شود، به‌طوری که تنها چند کلمه توجه قابل توجهی دریافت می‌کنند.

هر موقعیت در دنباله به همه موقعیت‌ها توجه می‌کند، اما قدرت توجه توسط ضرب نقطه‌ای تعیین می‌شود که منجر به یک ماتریس پراکنده می‌شود. در این ماتریس، بیشتر وزن‌های توجه نزدیک به صفر هستند. این پراکندگی از طریق تابع softmax به‌دست می‌آید که بر مرتبط‌ترین کلمات تأکید کرده و کلمات کمتر مرتبط را سرکوب می‌کند.



شکل ۱: ماتریس وزن‌ها در مکانیزم خودتوجه

- بلوک‌های سایه‌دار وزن‌های توجه غیرصفر را نشان می‌دهند که بیانگر اهمیت توجه است.
- بلوک‌های مورب نشان‌دهنده توجه به یک کلمه (توجه به خود) هستند که ممکن است به دلیل اهمیت توجه به یک کلمه صفر نباشد.
- بلوک‌های خارج از مورب نشان‌دهنده توجه به کلمات دیگر در دنباله هستند، جایی که بیشتر وزن‌های توجه به دلیل پراکندگی نزدیک به صفر هستند.

این ساختار پراکنده با به‌اشتراک‌گذاری پارامترها به مکانیزم خودتوجه اجازه می‌دهد تا وابستگی‌های بین کلمات را در توالی ورودی به‌طور مؤثر ثبت کند و درعین‌حال قابلیت پردازش محاسباتی را حفظ نماید.

۴. مکانیزم Self-Attention به دلیل ساختار خود، نسبت به ترتیب ورودی‌ها Permutation-Equivariant است؛ به این معنا که اگر ترتیب سطرهای ورودی جابه‌جا شود، خروجی نهایی نیز مطابق همان جابه‌جایی تغییر خواهد کرد. این ویژگی زمانی صادق است که مکانیزم Positional Encoding استفاده نشده باشد. در اینجا این رفتار را به طور دقیق‌تر بررسی می‌کنیم.

فرض کنید ورودی به صورت یک ماتریس $X \in \mathbb{R}^{N \times D}$ باشد، که در آن N تعداد توکن‌ها و D نیز Embedding Dimension است. اکنون فرض کنید یک جایگشت سطرها با تابع

$$\pi : \{۱, ۲, \dots, N\} \to \{۱, ۲, \dots, N\}$$

اعمال شده است و ترتیب سطرهای X را به $X' \in \mathbb{R}^{N \times D}$ تبدیل کرده است؛ به طوری که $X'_{\pi(i)} = X_i$ برقرار می‌باشد. برای ماتریس ورودی X'، بردارهای Query، Key، و Value به صورت زیر محاسبه می‌شوند:

$$Q' = X'W^Q, \quad K' = X'W^K, \quad V' = X'W^V.$$

به دلیل خطی بودن ضرب ماتریسی و اعمال جایگشت، رابطه زیر برقرار است:

$$Q'_{\pi(i)} = Q_i, \quad K'_{\pi(i)} = K_i, \quad V'_{\pi(i)} = V_i$$

اکنون ضرب درونی ماتریس‌های Q' و K' به صورت $Q'K'^{\mathsf{T}}$ خواهد بود. در اینجا، جایگشت در سطرها و ستون‌ها به طور همزمان اعمال می‌شود:

$$(Q'K'^{\mathsf{T}})_{\pi(i),\pi(j)} = (QK^{\mathsf{T}})_{i,j}$$

با اعمال Softmax روی هر سطر، ضرایب توجه نیز به همان ترتیب سطرها تحت جایگشت قرار می‌گیرند:

$$\mathrm{Softmax}\Big(\frac{Q'K'T}{\sqrt{d_k}}\Big)_{\pi(i),\pi(j)} = \mathrm{Softmax}\Big(\frac{QK^{\mathsf{T}}}{\sqrt{d_k}}\Big)_{i,j}$$

بنابراین وزن‌های به‌دست‌آمده در Self-Attention مستقیماً با جایگشت ورودی مطابقت دارند. محاسبه خروجی نهایی با استفاده از بردارهای وزن‌دار شده‌ی V' به صورت زیر انجام می‌شود:

$$Y' = \mathrm{Attention}(Q', K', V') = \mathrm{Softmax}\Big(\frac{Q'K'T}{\sqrt{d_k}}\Big)V' \implies Y'_{\pi(i)} = Y_i$$

و این همان چیزی است که می‌خواستیم نشان دهیم.

پاسخ مسئله‌ی ۴.

۱.
- ورودی Encoder: [۳۲, ۲۰۴۸, ۱۰۲۴] (سایز بچ، طول دنباله، ابعاد Embedding)
- خروجی Self-Attention در Encoder (هر سر): [۳۲, ۲۰۴۸, ۱۹۲]
- خروجی Self-Attention ادغام شده در Encoder: [۳۲, ۲۰۴۸, ۱۰۲۴]
- خروجی لایه FeedForward اول در Encoder: [۳۲, ۲۰۴۸, ۵۱۲]
- خروجی لایه FeedForward دوم در Encoder: [۳۲, ۲۰۴۸, ۱۰۲۴]
- خروجی Encoder: [۳۲, ۲۰۴۸, ۱۰۲۴]
- ورودی Decoder: [۳۲, ۲۰۴۸, ۱۰۲۴]
- خروجی Self-Attention در Decoder (هر سر): [۳۲, ۲۰۴۸, ۱۹۲]
- خروجی Self-Attention ادغام شده در Decoder: [۳۲, ۲۰۴۸, ۱۰۲۴]
- خروجی Cross-Attention در Decoder (هر سر): [۳۲, ۲۰۴۸, ۱۹۲]
- خروجی Cross-Attention ادغام شده در Decoder: [۳۲, ۲۰۴۸, ۱۰۲۴]
- خروجی لایه FeedForward اول در Decoder: [۳۲, ۲۰۴۸, ۵۱۲]
- خروجی لایه FeedForward دوم در Decoder: [۳۲, ۲۰۴۸, ۱۰۲۴]
- خروجی نهایی مدل: [۳۲, ۲۰۴۸, ۳۰۰۰۰]

۲. فرضیات داده شده:

- طول دنباله ورودی: ۲۰۴۸
- سایز واژگان ورودی: ۳۰۰۰۰
- بعد بُردار کلمه (Embedding Vector): ۷۶۸
- تعداد بلوک‌های Encoder: ۸
- تعداد بلوک‌های Decoder: ۱۲
- تعداد سرهای توجه (Attention Heads): ۴
- سایز بچ: ۳۲
- تعداد لایه‌های FeedForward: ۲

محاسبات Encoder

(آ) پارامترهای Self-Attention:

$$d_k, d_v : \frac{۷۶۸}{۴} = ۱۹۲$$
$$W_Q, W_K, W_V : ۳ \times ۱۰۲۴ \times ۱۹۲ = ۵۸۹,۸۲۴$$
$$W_O : ۷۶۸ \times ۱۰۲۴ = ۷۸۶,۴۳۲$$

(ب) کل پارامترهای Multi-Head Attention:

$$۴(۵۸۹,۸۲۴) + ۷۸۶,۴۳۲ = ۳,۱۴۵,۷۲۸$$

(ج) پارامترهای FeedForward:

$$W_۱ : ۱۰۲۴ \times ۵۱۲ + ۵۱۲ = ۵۲۴,۸۰۰$$
$$W_۲ : ۵۱۲ \times ۱۰۲۴ + ۱۰۲۴ = ۵۲۵,۳۱۲$$

(د) مجموع پارامترهای هر بلوک:

$$۳,۱۴۵,۷۲۸ + ۵۲۴,۸۰۰ + ۵۲۵,۳۱۲ = ۴,۱۹۵,۸۴۰$$

(ه) پارامترهای تمام بلوک‌های Encoder:

$$۸ \times ۴,۱۹۵,۸۴۰ = ۳۳,۵۶۶,۷۲۰$$

محاسبات Decoder

(آ) مشابه Encoder، Decoder دارای پارامترهای زیر است: (دقت کنید که Decoder یک لایه Cross-Attention نیز دارد که تعداد پارامترهای آن کاملا مشابه لایه Self-Attention می‌باشد)

$$۱۲ \times (۴,۱۹۵,۸۴۰ + ۳,۱۴۵,۷۲۸) = ۸۸,۰۹۸,۸۱۶$$

مجموع پارامترهای اولیه و نهایی

(آ) پارامترهای Embedding: (برای لایه linear آخر که خروجی Decoder را به خروجی نهایی مدل تبدیل می‌کند)

$$۱۰۲۴ \times ۳۰۰۰۰ + ۳۰۰۰۰ = ۳۰,۷۵۰,۰۰۰$$

(ب) مجموع کل پارامترها:

$$۳۳,۵۶۶,۷۲۰ + ۸۸,۰۹۸,۸۱۶ + ۳۰,۷۵۰,۰۰۰ = ۱۵۲,۴۱۵,۵۳۶$$

۳. فرض کنید ورودی جمله زیر باشد:

"گربه روی تشک خوابیده است."

و هدف ترجمه به انگلیسی باشد:

"The cat is sleeping on the mat."

مدل با مراحل زیر عمل می‌کند:

(آ) **جاسازی توکن‌ها (Token Embedding):**

- هر کلمه به برداری با ابعاد ۱۰۲۴ نگاشت می‌شود. این بردارها به صورت مقادیر عددی شناور نمایش داده می‌شوند و هر بردار اطلاعات معنایی و دستوری کلمه را رمزگذاری می‌کند.
- به عنوان مثال، جمله ورودی "گربه روی تشک خوابیده است." به تعدادی توکن تقسیم شده و سپس هر توکن به یک بردار ۱۰۲۴ بُعدی تبدیل می‌شود.
- شکل خروجی اولیه این مرحله به صورت [۳۲, ۲۰۴۸, ۱۰۲۴] است که ۳۲ نشان‌دهنده سایز batch، ۲۰۴۸ طول دنباله (sequence length) و ۱۰۲۴ ابعاد هر بردار است.
- در این مرحله، Positional Encoding نیز به بردارها اضافه می‌شود تا مدل بتواند ترتیب کلمات را در نظر بگیرد.

(ب) **پردازش Encoder:**

- ابتدا، برای هر توکن ورودی، ماتریس‌های Q, K, V (پرسش، کلید و مقدار) محاسبه می‌شوند. این ماتریس‌ها ابعاد [۳۲, ۲۰۴۸, ۱۹۲] برای هر Head دارند.
- محاسبات Attention با استفاده از فرمول زیر انجام می‌شود:

$$\text{Attention}(q, k, v) = \text{Softmax}\left(\frac{QK^{\mathsf{T}}}{\sqrt{d_k}}\right)V$$

که در آن $d_k = ۱۹۲$ است. این عملیات برای یافتن میزان توجه به هر توکن دیگر در دنباله استفاده می‌شود.

- سپس، نتایج Head ها ادغام شده و پس از ضرب شدن در ماتریس $W^O \in \mathbb{R}^{۷۶۸\times۱۰۲۴}$ به ابعاد اولیه بازگردانده می‌شود ([۳۲, ۲۰۴۸, ۱۰۲۴]).
- نتایج از دو لایه FeedForward عبور می‌کنند: (σ تابع فعال‌سازی است)

$$FFN(X) = \sigma(\sigma(XW_۱ + b_۱)W_۲ + b_۲)$$

- لایه‌های FeedForward به ترتیب ابعاد [۳۲, ۲۰۴۸, ۵۱۲] و [۳۲, ۲۰۴۸, ۱۰۲۴] را تولید می‌کنند که خروجی دومین لایه FeedForward همان خروجی نهایی Encoder است.

(ج) **پردازش Decoder:**

- ابتدا Self-Attention مشابه Encoder برای توکن‌های هدف محاسبه می‌شود.
- سپس، Cross-Attention برای ارتباط بین خروجی Encoder و ورودی Decoder انجام می‌شود.
- فرمول Cross-Attention به صورت زیر است:

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

که در آن Q از Decoder و K, V از Encoder محاسبه می‌شوند.

- مشابه Encoder، نتایج از دو لایه FeedForward عبور کرده و بردارهای نهانی نهایی تولید می‌شوند.

(د) پیش‌بینی کلمات:

- خروجی Decoder به یک ماتریس خطی با ابعاد [۳۰۰۰۰ × ۱۰۲۴] ضرب می‌شود که در آن ۳۰۰۰۰ تعداد کل واژگان است.
- با استفاده از تابع Softmax، احتمالات توزیع روی واژگان محاسبه می‌شود:

$$p(w) = \frac{e^{z_w}}{\sum_{v\in} e^{z_v}}$$

که در آن z_w امتیاز (logit) مربوط به واژه w است.

- واژه‌ها به صورت Autoregressive تولید می‌شوند. این به این معنی است که هر کلمه به ترتیب پیش‌بینی شده و به ورودی اضافه می‌شود تا کلمه بعدی پیش‌بینی گردد.
- فرآیند پیش‌بینی تا رسیدن به توکن پایان جمله ([EOS]) یا تکمیل طول حداکثر ادامه می‌یابد.

پاسخ مسئله‌ی ۵.

الف) مدل BERT در زمان آموزش اولیه خود از هدف Next Sentence Prediction (NSP) برای یادگیری روابط معنایی بین جملات متوالی استفاده می‌کند. در این فرآیند، مدل با جفت جملات آموزش داده می‌شود تا تشخیص دهد که آیا جمله دوم به‌طور منطقی به جمله اول متصل است یا خیر. این هدف به مدل کمک می‌کند تا قابلیت درک روابط متنی و ترتیب جملات را بیاموزد. با این حال، در برخی تسک‌های پردازش زبان طبیعی، نیاز به چنین اطلاعاتی وجود ندارد.

در مدل‌هایی مانند RoBERTa، هدف NSP حذف شده و تمرکز کامل بر روی Masked Language Modeling (MLM) قرار گرفته است. دلیل اصلی این تغییر آن است که بسیاری از وظایف پایین‌دستی، مانند طبقه‌بندی متن یا تشخیص احساسات، به درک معنایی کلی جملات نیاز دارند و نیازی به بررسی روابط ترتیبی بین جملات ندارند. حذف NSP باعث می‌شود منابع محاسباتی بیشتری به یادگیری ارتباطات معنایی میان کلمات اختصاص یابد و عملکرد مدل در این نوع وظایف بهبود پیدا کند.

همچنین، حذف NSP پیچیدگی آموزش را کاهش می‌دهد و باعث می‌شود مدل به‌طور کارآمدتری برای تسک‌های متنی مانند پاسخ به سوالات یا ترجمه آموزش داده شود. مدل‌هایی مانند RoBERTa با تکیه بر این استراتژی موفق به دستیابی به نتایج بهتری در مقایسه با BERT در بسیاری از معیارهای استاندارد ارزیابی عملکرد شده‌اند.

ب) برای بهبود عملکرد BERT در وظایف پایین‌دستی، روش‌هایی مانند Sentence-BERT (SBERT) معرفی شده‌اند که هدف آن‌ها افزایش توانایی مدل در درک معنایی جملات کامل است. برخلاف BERT که بر روی توکن‌های منفرد تمرکز دارد، SBERT از جملات به‌عنوان واحدهای معنایی مستقل استفاده می‌کند و بردارهای متراکم و معنایی برای جملات استخراج می‌کند.

در SBERT، به‌جای استفاده از خروجی‌های مربوط به توکن‌های منفرد، کل جمله به یک بردار تبدیل می‌شود که خلاصه‌ای از محتوای آن است. این روش معمولاً از تکنیک pooling برای تولید بردار نهایی جمله استفاده می‌کند. در این فرآیند، ابتدا بردارهای خروجی مربوط به تمام توکن‌های جمله از آخرین لایه مدل استخراج می‌شوند. سپس از یکی از روش‌های pooling مانند mean pooling، max pooling یا انتخاب بردار مربوط به توکن [CLS] برای ترکیب اطلاعات تمامی توکن‌ها به یک بردار استفاده می‌شود.

روش mean pooling با محاسبه میانگین بردارهای تمامی توکن‌ها، یک نمای کلی و متوازن از اطلاعات جمله ایجاد می‌کند. در مقابل، max pooling با انتخاب بیشترین مقدار در هر بُعد بردارها، بر ویژگی‌های قوی‌تر تاکید دارد. استفاده از توکن [CLS] نیز برای نمایندگی جمله محبوب است، زیرا این توکن به‌طور خاص در طول آموزش برای خلاصه‌سازی اطلاعات کل جمله آموزش دیده است.

بردار نهایی به‌دست‌آمده از pooling می‌تواند به‌عنوان ورودی برای مدل‌های پایین‌دستی مانند تطابق معنایی جملات یا خوشه‌بندی متن به کار رود. این رویکرد به‌ویژه برای تسک‌هایی که نیاز به مقایسه جملات دارند، مانند ارزیابی شباهت معنایی یا بازیابی اطلاعات، بسیار کارآمد است.

یکی دیگر از مزایای SBERT این است که به دلیل استفاده از بردارهای معنایی جمله، در مقایسه با BERT عملکرد بهتری در وظایف ارزیابی شباهت جملات دارد. همچنین، این روش برای تسک‌هایی مانند پاسخ به سوالات و طبقه‌بندی متن بسیار موثرتر است، زیرا نیاز به پردازش جداگانه توکن‌ها را از بین می‌برد.

به طور کلی، تنظیم دقیق BERT با استفاده از روش‌هایی مانند SBERT، امکان یادگیری بردارهای معنایی دقیق‌تر را فراهم می‌کند و باعث می‌شود مدل در تسک‌های پایین‌دستی عملکرد بهتری داشته باشد.