بسم الله الرحمن الرحیم

دانشگاه صنعتی شریف
دانشکده‌ی مهندسی کامپیوتر

یادگیری عمیق

نیم‌سال دوم ۰۴-۰۳

مدرس: مهدیه سلیمانی

تمرین سوم | ددلاین تمرین: تمرین تئوری:۱۰ اردیبهشت ، تمرین عملی: ۱۹ اردیبهشت

- برای ارسال هر تمرین تا ساعت ۲۳:۵۹ روز ددلاین فرصت دارید. دقت شود که تمرین تئوری سوم تاخیر مجاز ندارد (به دلیل نزدیکی با میانترم درس) اما برای تمرین عملی یک هفته تاخیر مجاز دارید (یعنی حداکثر تاریخ ارسال تمرین تئوری ۱۰ اردیبهشت و تمرین عملی ۲۶ اردیبهشت است.)
- در هر کدام از سوالات، اگر از منابع خارجی استفاده کرده‌اید باید آن را ذکر کنید. در صورت همفکری با افراد دیگر هم باید نام ایشان را در سوال مورد نظر ذکر نمایید.
- پاسخ تمرین باید ماحصل دانسته‌های خود شما باشد. در صورت رعایت این موضوع، استفاده از ابزارهای هوش مصنوعی با ذکر نحوه و مصداق استفاده بلامانع است.
- پاسخ ارسالی واضح و خوانا باشد. در غیر این صورت ممکن است منجر به از دست دادن نمره شود.
- پاسخ ارسالی باید توسط خود شما نوشته شده باشد. به اسکرین‌شات از منابع یا پاسخ افراد دیگر نمره‌ای تعلق نمی‌گیرد.
- در صورتی که بخشی از سوال‌ها را جای دیگری آپلود کرده و لینک آن را قرار داده باشید، حتما باید تاریخ آپلود مشخص و قابل اتکا باشد.
- محل بارگذاری سوالات نظری و عملی در هر تمرین مجزا خواهد بود. به منظور بارگذاری بایستی تمارین تئوری در یک فایل pdf با نام HW3_[First-Name]_[Last-Name]_[Student-Id].pdf و تمارین عملی نیز در یک فایل مجزای زیپ با نام HW3_[First-Name]_[Last-Name]_[Student-Id].zip بارگذاری شوند.
- در صورت وجود هرگونه ابهام یا مشکل، در کوئرای درس آن مشکل را بیان کنید و از پیغام دادن مستقیم به دستیاران آموزشی خودداری کنید.
- طراحان این تمرین : محمد امانلو، آرش رسولی، مهرداد صالحی، محمد جواد رنجبر، شایگان ادیم، امیرمحمد ایزدی

بخش نظری (۱۰۰ نمره)

پرسش اول (۱۵ نمره)

در این سوال به بررسی Backpropagation در شبکه‌های LSTM می‌پردازیم. معماری استاندارد LSTM را معرفی کرده و فرآیند کامل پس انتشار خطا (Backpropagation) آن را به صورت ریاضیاتی استخراج کنید. گام به گام مشتقات را با تمامی معادلات مرتبط ارائه دهید.

نمادگذاری متغیرها:

- $\mathbf{x}_t$: ورودی در گام زمانی t
- $\mathbf{h}_t$: حالت پنهان در گام زمانی t
- $\mathbf{c}_t$: حالت سلول (حافظه) در گام زمانی t

- $\mathbf{f}_t$: خروجی دروازه فراموشی در گام زمانی t
- $\mathbf{i}_t$: خروجی دروازه ورودی در گام زمانی t
- $\mathbf{o}_t$: خروجی دروازه خروجی در گام زمانی t
- $\mathbf{g}_t$: کاندیدای حالت سلول در گام زمانی t
- $\tilde{\mathbf{x}}_t$: ترکیب حالت پنهان قبلی و ورودی فعلی، یعنی $\tilde{\mathbf{x}}_t = \begin{bmatrix} \mathbf{h}_{t-1} \\ \mathbf{x}_t \end{bmatrix}$
- $\mathbf{W}_f, \mathbf{W}_i, \mathbf{W}_g, \mathbf{W}_o$: ماتریس‌های وزن مربوط به دروازه‌های فراموشی، ورودی، کاندیدای حالت سلول و خروجی
- $\mathbf{b}_f, \mathbf{b}_i, \mathbf{b}_g, \mathbf{b}_o$: بایاس‌های مربوط به دروازه‌های مذکور

پرسش دوم (۲۰ نمره)

در این سوال به بررسی ویژگی‌های تابع خودتوجه و عملکرد آن در مدل‌های ترانسفورمر می‌پردازیم.

۱. **بخش الف**نشان دهید تابع خودتوجه که به صورت زیر تعریف می شود:

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

به صورت یک ماتریس تمام متصل (fully connected) در قالب یک ماتریس که تمام دنباله ورودی از بردارهای کلمه به صورت پیوسته را به یک بردار خروجی با همان بعد نگاشت می‌کند، می‌تواند توسعه یابد.

۲. **بخش ب**: سپس ثابت کنید که چنین ماتریسی از (N^2D^2) پارامتر برخوردار خواهد بود.

۳. **بخش ج**: نشان دهید که شبکه خودتوجه متناظر با یک نسخه پراکنده(sparse) از این ماتریس با اشتراک‌گذاری پارامترها است. یک نمودار از ساختار این ماتریس را بکشید و نشان دهید که کدام بلوک‌های پارامتر به اشتراک گذاشته می‌شوند و کدام بلوک‌ها تمام عناصرشان صفر است.

۴. **بخش د**: فرض کنید که $E \in \mathbb{R}^{n \times d_{\text{model}}}$ ماتریسی است که شامل بردارهای موقعیتی E_t است که موقعیت t را در یک دنباله ورودی به طول n نشان می‌دهد. به عبارت دیگر، این ماتریس تابعی از $e : \{1, \ldots, n\} \to \mathbb{R}^{d_{\text{model}}}$ است که به صورت زیر تعریف می‌شود:

$$e(t) = E_{t,:} := \begin{bmatrix} \sin\left(\frac{t}{f_1}\right) \\ \cos\left(\frac{t}{f_2}\right) \\ \sin\left(\frac{t}{f_3}\right) \\ \cos\left(\frac{t}{f_4}\right) \\ \vdots \\ \sin\left(\frac{t}{f_{d_{\text{model}}}}\right) \\ \cos\left(\frac{t}{f_{d_{\text{model}}}}\right) \end{bmatrix}$$

که در آن فرکانس‌ها به صورت زیر تعریف می‌شوند:

$$f_m = \frac{1}{\lambda_m} := 10000^{\frac{2m}{d_{\text{model}}}}.$$

سپس نشان دهید که یک تبدیل خطی $T^{(k)} \in \mathbb{R}^{d_{\text{model}} \times d_{\text{model}}}$ وجود دارد که برای آن رابطه زیر برقرار است:

$$T^{(k)} E_{t,:} = E_{t+k,:}$$

به این معنی که این تبدیل خطی برای هر انحراف موقعیتی $k \in \{1, \ldots, n\}$ در هر موقعیت معتبر $t \in \{1, \ldots, n-k\}$ در دنباله برقرار است.

پرسش سوم (۳۰ نمره)

خودتوجه چندسر جزء اصلی مدل‌سازی ترنسفرمرها است. در این سوال، ما می‌خواهیم بررسی کنیم که چرا خودتوجه چندسر می‌تواند به خودتوجه تک‌سر ترجیح داده شود. می‌دانیم که مکانیسم توجه را می‌توان به عنوان یک عملیات پرس‌وجو دید که در آن $q \in \mathbb{R}^d$ عبارت پرس و جو، در کنار $\{v_1, ..., v_n\}$ ، $v_i \in \mathbb{R}^d$ به عنوان مجموعه‌ای از بردارهای مقدار و مجموعه‌ای از بردارهای کلید $\{k_1, ..., k_n\}$، $k_i \in \mathbb{R}^d$ به صورت زیر مشخص می‌شود.

$$c = \sum_{i=1}^{n} v_i \alpha_i \qquad (۱)$$

$$\alpha_i = \frac{\exp(k_i^\top q)}{\sum_{j=1}^{n} \exp(k_j^\top q)} \qquad (۲)$$

در اینجا α_i وزن توجه نامیده می‌شود.همان‌طور که می‌بینید خروجی $c \in \mathbb{R}^d$ یک میانگین وزندار از بردارهای مقدار با وزن α_i است.

(الف) امکان شباهت بردار خروجی مکانیسم توجه به یکی از بردارهای مقدار: در این بخش می‌خواهیم بگوییم که در شرایط خاصی امکان این وجود دارد که بردار c شباهت زیادی به یکی از بردارهای v_i از بردارهای مقدار داشته باشد. به این منظور به سوالات زیر پاسخ دهید.

(i) توضیح دهید که چرا α را می‌توان به عنوان توزیع احتمال Categorical تفسیر کرد .

(ii) توزیع α معمولاً نسبتاً "پراکنده" است، به طوری که جرم احتمال بین α_i های مختلف توزیع می‌شود. اما این همیشه درست نیست. در یک جمله توضیح دهید که در چه شرایطی توزیع طبقه‌ای α تقریباً تمام وزن خود را بر روی مقدار α_j، به طوری که $j = \{1, ..., n\}$، متمرکز می‌کند؟ (به عبارت دیگر پرس‌وجو q و یا کلیدهای $k_1, ..., k_n$ چه ویژگی‌هایی باید داشته باشند؟)

(iii) با توجه به توضیحاتی که در (ii) بیان کردید، اگر توزیع α پراکنده باشد، خروجی c چه ویژگی‌هایی خواهد داشت؟

(iv) در راستای توضیح امکان شباهت بین بردار خروجی مکانیسم توجه و یکی از بردارهای مقدار، به طور خلاصه توضیح دهید که از (iii) و (iiii) چه نتیجه‌ای می‌توان گرفت؟

(ب) قابلیت ترکیب: یک مدل ترنسفورمر که فقط بر یک بردار v_j متمرکز است، ممکن است بخواهد اطلاعات را از چندین بردار منبع ترکیب کند. فرض کنید که می‌خواهیم اطلاعات دو بردار v_a و v_b را با بردارهای کلید k_a و k_b ترکیب کنیم.

(i) چگونه باید دو بردار v_b و v_a را در یک بردار خروجی ترکیب کنیم تا اطلاعات هر دو بردار حفظ شود؟ یکی از روش‌های رایج برای انجام این کار در یادگیری ماشین، محاسبه میانگین است:

$$c = \frac{1}{2}(v_a + v_b)$$

استخراج اطلاعات در مورد بردارهای اصلی v_a و v_b از حاصل ممکن است به نظر دشوار بیاید، اما در شرایط خاصی می‌توان این کار را انجام داد. در این مسئله، خواهیم دید که جز این موارد است.

فرض کنید که اگرچه ما v_a یا v_b را نمی‌دانیم، اما می‌دانیم که v_a در یک زیرفضای A قرار دارد که توسط m بردار پایه $\{a_1, a_2, \ldots, a_m\}$ تشکیل شده است، در حالی که v_b در یک زیرفضای ناهمپوشان B قرار دارد که توسط بردارهای پایه $\{b_1, b_2, \ldots, b_p\}$ تشکیل شده است. تمام بردارهای پایه دارای طول ۱ و متعامد با یکدیگر هستند. علاوه بر این، فرض کنید که دو زیرفضا متعامد هستند، یعنی:

$$\langle a_i, b_k \rangle = 0 \quad \text{برای همه } i \text{ و } k.$$

اگر $v_a \in A$ و $v_b \in B$، می‌توانیم از ماتریس M برای استخراج v_a از بردار مجموع $s = v_a + v_b$ استفاده کنیم. به عبارت دیگر، می‌خواهیم M را طوری بسازیم که برای هر v_a:

$$M \cdot s = v_a.$$

نکته: v_a و v_b باید به صورت یک بردار در $\mathbb{R}^d$ بیان شوند، نه بر حسب بردارهای A و B.
نکته: با توجه به اینکه بردارهای $\{a_1, a_2, \ldots, a_m\}$ هم متعامد هستند و هم مبنای نرمال شده ای برای v_a می دانیم که مقدار $\{c_1, c_2, \ldots, c_m\}$ وجود دارد به طوری که $v_a = (c_1 a_1 + c_2 a_2 + \cdots + c_m a_m)$
(ii) همانطور که قبلاً گفتیم، تصور کنید v_a و v_b دو بردار مقدار باشند که به ترتیب مربوط به بردارهای کلید k_i هستند. فرض کنید که تمام بردارهای کلید متعامد هستند، بنابراین $\langle k_i, k_j \rangle = 0$ برای همه $i \neq j$ و تمام بردارهای کلید دارای طول ۱ هستند. یک عبارت برای بردار پرس و جو q پیدا کنید به طوری که:

$$\frac{1}{2}(v_a + v_b) \approx c.$$

(ج) رفع یکی از اشکالات توجه تک سر: در قسمت قبل دیدیم که چگونه ممکن است توجه تک سر به طور مساوی روی دو مقدار متمرکز شود. این مفهوم را می‌توان به راحتی به هر زیرمجموعه‌ای از بردارهای مقدار تعمیم داد. در این سوال خواهیم دید که چرا این راه‌حل عملی نیست.

مجموعه‌ای از بردارهای کلید $k_1, \ldots, k_n$ را در نظر بگیرید که اکنون به صورت تصادفی نمونه‌برداری شده‌اند:

$$k_i \sim N(\mu_i, \Sigma_i),$$

که در آن میانگین μ_i برای شما شناخته شده است، اما کوواریانس Σ_i ناشناخته است. علاوه بر این، فرض کنید:

$$\|\mu_i\| = 1,$$

و بردارهای میانگین μ_i همگی متعامد هستند. اگر $j \neq i$، آنگاه:

$$\langle \mu_i, \mu_j \rangle = 0.$$

(i) تصور کنید که ماتریس‌های کوواریانس برای α بسیار کوچک به صورت $\Sigma_i = \alpha I$، برای $\forall_i \in \{1, \ldots, n\}$ تعریف می‌شوند. یک پرس و جو q را بر حسب μ_i طراحی کنید که k_i ها از توزیع نرمالی با میانگین آن‌ها نمونه‌برداری شوند، طوری که مانند قبل:

$$c \approx \frac{1}{2}(v_a + v_b).$$

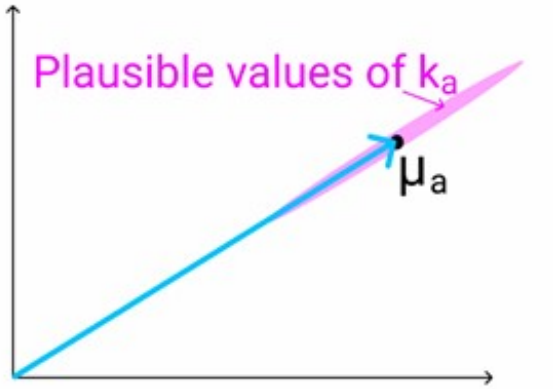
یک استدلال مختصر در مورد اینکه چرا این اتفاق می‌افتد ارائه دهید.
(ii) اگرچه توجه تک‌سر در برابر آشفتگی‌های کوچک در کلیدها مقاوم است، اما برخی از انواع آشفتگی‌های بزرگ‌تر ممکن است مشکلات جدی‌تری ایجاد کنند. به طور خاص، در برخی موارد، یکی از بردارهای کلید k_a ممکن است از نظر نرم بزرگ‌تر یا کوچک‌تر از بقیه باشد، در حالی که همچنان در همان راستای میانگین μ_a است. به عنوان مثال، تصور کنید که کوواریانس نمونه بسیار کوچک a به صورت زیر باشد:

$$\Sigma_a = \frac{1}{2}(\mu_a \mu_a^T) + \alpha I$$

در نظر بگیرید این باعث می‌شود که k_a تقریباً در جهتی مشابه μ_a باشد، اما با واریانس‌های بزرگ در اندازه. علاوه بر این، در نظر بگیرید که:

$$\Sigma_a = \alpha I$$

برای همه $a \neq i$.



شکل ۳: بردار μ_i (که در اینجا به صورت دوبعدی به عنوان مثال نشان داده شده است)، با محدوده‌ی مقادیر ممکن k_a به رنگ بنفش نشان داده شده است. همانطور که قبلاً ذکر شد، k_a تقریباً در جهتی مشاب μ_a است، اما ممکن است اندازه‌ی آن بزرگ‌تر یا کوچک‌تر باشد.

وقتی چندین بار از $\{k_1, \ldots, k_n\}$ نمونه‌برداری می‌کنید و از بردار q که در قسمت i تعریف کردید استفاده می‌کنید، انتظار دارید بردار c از نظر کیفی برای نمونه‌های مختلف چگونه باشد و مقدار آن در چه حدود مقادیری تغییر کند؟

(د) راه‌حل با استفاده از مزایای توجه چندسر: اکنون می‌خواهیم در مورد توانایی خودتوجهی چندسر در مواجهه با این مشکل صحبت کنیم. ما یک نسخه ساده از خودتوجهی چندسر را در نظر خواهیم گرفت که مشابه خودتوجهی تک‌سر است که در این تمرین ارائه کرده‌ایم، به جز اینکه دو بردار پرس و جو q_1, q_2 تعریف شده است که منجر به یک جفت از بردارهای c_1, c_2 می‌شود که هر یک خروجی خودتوجهی تک‌سر نسبت به بردار پرس و جو مربوطه‌ی خود هستند. خروجی نهایی خودتوجهی چندسر، میانگین آن‌هاست:

$$\frac{1}{2}(c_1 + c_2).$$

همانطور که در بخش سوم این سوال، مجموعه‌ای از بردارهای کلید $\{k_1, \ldots, k_n\}$ را در نظر گرفتیم که به طور تصادفی نمونه‌برداری می‌شوند:

$$k_i \sim N(\mu_i, \Sigma_i),$$

که در آن میانگین μ_i برای شما شناخته شده است، اما کوواریانس Σ_i ناشناخته است. فرض می‌کنیم که میانگین μ_i متعامد هستند و برای $j \neq i$:

$$\langle \mu_i, \mu_j \rangle = 0.$$

همچنین نرم واحد دارند:

$$\|\mu_i\| = 1.$$

(i) فرض کنید که ماتریس‌های کوواریانس برای α های بسیار کوچک به صورت $\Sigma_i = \alpha I$ تعریف شده‌اند. بردارهای q_1 و q_2 را به گونه‌ای پیدا کنید که خروجی برابر با میانگین دو بردار v_a و v_b شود.

(ii) فرض کنید که ماتریس‌های کوواریانس برای α های بسیار کوچک به صورت:

$$\Sigma_a = \frac{1}{2}(\mu_a \mu_a^T) + I$$

و برای هر i، اگر $a \neq i$، آنگاه برابر:

$$\Sigma_i = \alpha I$$

هستند. بردارهای پرس و جو q_1 و q_2 را که در بخش قبلی سوال طراحی شده‌اند، در نظر بگیرید. انتظار دارید خروجی c بر اساس نمونه‌های مختلف بردارهای کلید چگونه باشد؟ یا به عبارت دیگر، چه رابطه‌ای بین بردار c و بردارهای مقدار مربوط به کلیدها وجود خواهد داشت؟ (لطفاً به طور خلاصه توضیح دهید که چرا می‌توانید مواردی را که در آن $k_a^T q_i < 0$ است نادیده بگیرید.)

پرسش چهارم (۲۰ نمره)

یکی از مشکلات عملیاتی مکانیزم توجه با context window های بزرگ، هزینه‌ی محاسباتی بسیار زیاد محاسبه‌ی ماتریس توجه است. برای رفع این مشکل، روش‌های بسیاری به کار گرفته شده که مهم‌ترین آن‌ها، روش‌های KV-Cache هستند. اما این روش‌ها نیز مشکل مخصوص به خود را دارند که استفاده‌ی خیلی زیاد از حافظه‌ی cache است.
در این تمرین، قصد داریم هزینه محاسباتی یک گام inference تک توکنی از یک لایه Multi-head Attention در یک مدل زبانی بزرگ decoder-only را تحلیل کنیم و نقاط قوت و ضعف اعمال روش‌های KV-Cache را بررسی کنیم.
فرضیات:

- n: تعداد توکن‌ها
- d: بعد Embedding ورودی هر توکن
- d_h: بعد فضای Key-Query
- n_h: تعداد سرهای Attention

۱. تحلیلی دقیق از هزینه محاسباتی یک لایه از این ساختار را بر حسب متغیرهای داده شده ارائه دهید. توجه داشته باشید که در این بخش باید فرض کنید که هیچ روش بهینه‌سازی‌ای برای بهبود مکانیزم توجه نداریم.

۲. ابتدا KV-Cache را کامل توضیح دهید و سپس تحلیل قبل را تکرار کنید، اما این بار KV-cache معمول را برای ماتریس‌های کلید و مقدار در نظر بگیرید. تمام محاسبات را دوباره انجام دهید. در این بخش، تعداد عناصر(اعداد) مورد نیاز برای ذخیره در حافظه را نیز محاسبه کنید.

۳. ابتدا روش Multi-Query Attention را کامل توضیح دهید و بار دیگر، همان تحلیل را با استفاده از KV-cache معمول برای ماتریس‌های کلید و مقدار انجام دهید، اما این بار در معماری مدل از Attention Multi-Query استفاده می‌کنیم. همینطور مانند بخش قبل، تعداد عناصر(اعداد) مورد نیاز برای ذخیره در حافظه را محاسبه کنید.

۴. ابتدا روش Grouped-Query Attention را کامل توضیح دهید و سپس همان تحلیل را با استفاده از KV-cache معمول برای ماتریس‌های کلید و مقدار انجام دهید، اما این بار از Grouped-Query Attention استفاده می‌کنیم. تعداد عناصر(اعداد) مورد نیاز برای ذخیره در حافظه پنهان را محاسبه کنید. (g را به عنوان تعداد گروه‌ها در نظر بگیرید.)

۵. با فرض‌های زیر و با استفاده از fp۱۶، مقدار عددی پیچیدگی محاسباتی و همچنین حافظه‌ی cache استفاده شده را در هر یک از موارد بالا را محاسبه و مقایسه کنید:

- n = ۱۰۲۴
- d = ۷۶۸
- d_h = ۶۴
- n_h = ۱۲
- g = ۴

۶. (امتیازی) در یک نوآوری خلاقانه، در مدل DeepSeek-R۱ ،نوع جدیدی از معماری توجه به نام Attention Multi-Head Latent به کار گرفته شد که سرعت inference را به شدت افزایش و استفاده از حافظه را به شدت کاهش داد. معماری مکانیزم توجه در این مدل را به طور کلی توضیح دهید و نشان دهید که چگونه این روش استفاده از حافظه را به شدت کم می‌کند.

پرسش پنجم (۱۵ نمره)

۱. یکی از اولین و معروف‌ترین مدل‌ها Encoder-Only مدل BERT است که در سال ۲۰۱۸ معرفی شد. در مورد این مدل به سوالات زیر پاسخ دهید.

(آ) توضیح دهید که معنی اینکه می‌گوییم مدل BERT از ترنسفورمر دو طرفه استفاده می کند، چیست؟

(ب) آموزش این مدل بر اساس دو وظیفه Masked Language Modeling(MLM) و Next Sentence Prediction(NSP) بوده است. هر کدام از این وظایف را توضیح دهید و بگویید چگونه به آموزش و یادگیری مدل کمک می‌کند.

(ج) نقش توکن ویژه CLS چیست و چگونه از آن در هنگام تنظیم دقیق Fine-Tuning استفاده می شود؟

(د) آیا می‌توان با استفاده از مدل BERT وظیفه تولید متن را انجام داد؟ توضیح دهید که چگونه می‌شود یا چرا نمی‌شود.

(ه) یکی دیگر از مدل‌های موفق که پس از BERT معرفی شد مدل RoBERTa بود. توضیح دهید که چه تفاوتی در آن ایجاد شد.

(و) در مورد مدل ViT تحقیق کنید و به صورت مختصر بیان کنید که چه تفاوت و شباهتی با مدل BERT دارد.

۲. در همان سال‌هایی که مدل BERT معرفی شد، مدل Generative Pre-Training Transformer(GPT) معرفی شد که یک مدل Decoder-Only به حساب می‌آمد. معرفی معماری GPT پایه و ایده اصلی‌ای شده است برای تمام مدل‌های زبانی بزرگ مانند ChatGPT که امروزه می‌بینید. در این باره به سوالات زیر پاسخ دهید.

(آ) آموزش این مدل‌ها عموما با وظیفه Next Token Prediction(NTP) انجام می‌شوند. آن را توضیح دهید.

(ب) مکانزیم Masked-Attention در مدل‌های GPT چیست و چرا برای تولید متن ضروری است؟

(ج) روش تولید متن در این معماری را توضیح دهید. یعنی توضیح دهید که ورودی مدل در ابتدا چه بوده و هربار که لغت جدید تولید می‌شود چه اتفاقی صورت می‌گیرد.

(د) چرا معماری‌های Decoder-Only در زمان استنتاج کندتر از معماری‌های Encoder-Only هستند؟

(ه) طول کانتکست در یک مدل زبانی بزرگ چگونه بر توانایی آن در حفظ و بازیابی اطلاعات از ابتدای ورودی تأثیر می‌گذارد و چه تاثیری بر پیچیدگی محاسباتی و میزان حافظه مصرفی دارد؟

(و) چرا مدل‌های GPT ممکن است اطلاعات نادرست یا به اصطلاح "hallucination" تولید کنند، حتی اگر روی داده‌های واقعی آموزش دیده باشند؟

۳. یکی از روش‌های تنظیم دقیق مدل‌های بزرگ استفاده از روش‌های Parameter Efficient Fine-Tuning یا همان PEFT است که موجب صرفه‌جویی و بهینه کردن زمان و حافظه مورد نیاز برای آموزش است. حال به سوالات زیر پاسخ دهید:

(آ) توضیح دهید که روش‌های PEFT چگونه به بهینه کردن آموزش مدل‌ها کمک می‌کنند. همچنین در مورد روش‌های Adapter, Prompt-Tuning و LoRA که از این دسته از روش‌ها هستند توضیح مختصری بدهید و نحوه عملکرد آن‌ها را شرح دهید.

(ب) معمولاً در زمان آموزش با روش LoRA ماتریس B با مقدار صفر مقداردهی اولیه می‌شود. دلیل این کار را توضیح دهید.

(ج) فرض کنید یک معماری Transformer که فقط شامل یک بلوک Encoder داریم. اندازه بعد لایه‌های پنهان و امبدینگ‌ها برابر d است و همچنین h برابر تعداد head ها در بلوک Multi-Head Attention است و در لایه Feed-Forward سایز لایه پنهان ۴ برابر و سپس به اندازه خود باز می‌گردد. اگر بخواهیم روش LoRA با رنک r را بر روی تمامی ماتریس‌ها اعمال کنیم، تعداد پارامترهایی که آموزش می‌بینند را بدست آورید. (فقط پارامترهای Feed-Forward و Attention را در نظر بگیرید و از بایاس‌ها صرف نظر کنید.)

(د) در سال‌های اخیر با توجه به عملکرد خوب روش LoRA در تسک‌های مختلف، توجهات زیادی به این روش شده است و Extension های مختلفی از این روش ارائه شده است. با تحقیق و جستجو، ۴ مورد از این روش‌ها را پیدا کنید و توضیح مختصری در مورد آن‌ها و تفاوتشان با LoRA ارائه دهید. (هر مورد را نهایتاً در ۲ الی ۳ خط توضیح دهید.)

پرسش ششم (امتیازی - ۳۰ نمره)

ما این بخش را با مرور مختصری از فرمول‌بندی خود-توجه (Self-Attention) و کانولوشن‌ها آغاز می‌کنیم. هر دو روش به‌طور جداگانه و مشترک در وظایف مختلف پردازش زبان طبیعی استفاده شده‌اند. هدف ما از این سوال این است که درباره شباهت‌هایی که بین آن‌ها وجود دارد، بحث کنیم.

یک لایه خود-توجه در یک شبکه عصبی این گونه تعریف می‌شود که ورودی، ماتریس $Z \in \mathbb{R}^{N \times D_i}$ (نمایش‌دهنده N توکن ورودی) را می‌گیرد و ماتریس خروجی $O \in \mathbb{R}^{N \times D_o}$ را طبق عملیات زیر تولید می‌کند:

$$O = \text{Self-Attention}(Z) := \text{softmax}\left(\frac{ZW_qW_k^\top Z^\top}{\sqrt{d_k}}\right) ZW_v \qquad (۱)$$

که در آن $W_q \in \mathbb{R}^{(D_i \times D_{hid})}$، $W_k \in \mathbb{R}^{(D_i \times D_{hid})}$ و $W_v \in \mathbb{R}^{(D_i \times D_o)}$ ماتریس‌های قابل آموزش هستند که به ترتیب برای کوئری (query)، کلید (key) و مقدار (value) می‌باشند. این شکل خاص از توجه، به‌عنوان خود-توجه شناخته می‌شود، زیرا تمام ماتریس‌های وزن با همان ماتریس ورودی Z ضرب می‌شوند. برای وضوح بیشتر، ماتریس توجه $A \in \mathbb{R}^{n \times n}$ را به‌صورت زیر تعریف می‌کنیم:

$$A := ZW_qW_k^\top Z^\top \quad (۲)$$

عناصر این ماتریس به‌عنوان نمرات توجه (attention scores) شناخته می‌شوند و نشان می‌دهند که هر توکن چقدر به هر توکن دیگر مرتبط است. هر عنصر $\text{softmax}(A)$ را نیز احتمال‌های توجه (attention probabilities) می‌نامیم.
به طور تجربی ثابت شده است که انجام مکانیسم خود-توجه بر روی سرهای متعدد مفید است. به عبارت دیگر چند سر توجه داشتن به این معنی است که چندین بار به صورت موازی در یک لایه، هر بار با ماتریس های وزنی مختلف، فرایند خود-توجه را انجام دهیم. سپس خروجی های سرهای توجه به صورت زیر ترکیب شوند:

$$\text{MultiHead-Self-Attention}(Z)_{t,:} = \left(\underset{h \in [N_h]}{\text{concat}} \left(\text{Self-Attention}_h(Z) W_h^{\text{out}} \right) \right)_{t,:} + b_{\text{out}} \quad (۳)$$

که در آن $W_{\text{out}} \in \mathbb{R}^{(N_h \cdot D_o) \times D_{\text{out}}}$ و $b_{\text{out}} \in \mathbb{R}^{1 \times D_{\text{out}}}$ پارامترهای قابل آموزش هستند و N_h تعداد سرها است.
ردیف‌های ماتریس ورودی Z معمولاً تعبیه های توکن ها هستند که توکن را در یک فضای برداری نشان می‌دهند. با این حال، اخیراً از لایه‌های توجه با پچ تصویری به عنوان ورودی استفاده می‌شود که نتایج خوبی در وظایف بینایی رایانه مانند طبقه‌بندی تصویر به دست می‌آورد.
در این حالت، تصویر ورودی با یک تنسور $Z \in \mathbb{R}^{(W \times H \times D_i)}$ کدگذاری می‌شود، جایی که W و H اندازه‌های تصویر ورودی هستند و D_i تعداد کانال‌ها است. ما می‌خواهیم فرمول‌بندی خود-توجه در معادله ۱ را طوری تطبیق دهیم که با پیکسل‌ها $p = (i, j)$ به‌جای توکن‌های کلمه کار کند. برای ساده‌سازی نوشتار، ماتریس‌ها را با $Z_{p,:}$ به‌عنوان $Z_{i,j,:}$ نمایان می‌کنیم.

(الف)

چند پارامتر قابل آموزش در یک لایه خود-توجه چندسر با N_h سر، مانند آنچه در معادله ۳ توضیح داده شده، وجود دارد؟

(ب)

معادله ۱ و معادله ۳ را برای تک پیکسل کوئری q بازنویسی کنید، به طوری که خود توجه به صورت زیر بدست آید:

$$\text{Self-Attention}(Z)_{q,:} \in \mathbb{R}^{(1 \times D_o)}$$

و

$$\text{MultiHead-Self-Attention}(Z)_{q,:} \in \mathbb{R}^{(1 \times D_{\text{out}})}$$

ویژگی خاص خود-توجه همانطور که در معادله ۱ تعریف شده است، این است که تغییرات در ترتیب را تحمل می‌کند (permutation invariant). با این حال، هم زمانی که با دنباله‌های متنی کار می‌کنیم و هم زمانی که با تصاویر کار می‌کنیم، موقعیت هر توکن یا پیکسل اطلاعات مهمی را حمل می‌کند که می‌خواهیم آن را حفظ کنیم. برای انجام این کار، ما از یک ماتریس موقعیت (ماتریس $P \in \mathbb{R}^{N \times D_i}$ که می‌تواند یادگرفته شده یا ثابت باشد) استفاده می‌کنیم که فرم ماتریس توجه A در معادله ۲ را به‌صورت زیر تغییر می‌دهد:

$$A := (Z + P)W_qW_k^\top (Z + P)^\top \quad (۴)$$

رمزگذاری‌های موقعیتی می‌توانند مطلق(absolute) یا نسبی(relative) باشند. در رمزگذاری‌های مطلق، یک بردار موقعیت P_p برای پیکسل ورودی p، تنها به موقعیت مطلق p در ورودی بستگی دارد. بنابراین، ما می‌توانیم معادله ۴ را برای پیکسل کوئری q و پیکسل کلید k گسترش دهیم و به شکل زیر برسیم:

$$\begin{aligned} A_{q,k}^{\text{absolute}} &= (Z_{q,:} + P_{q,:})W_q W_k^\top (Z_{k,:} + P_{k,:})^\top \\ &= Z_{q,:}W_q W_k^\top Z_{k,:}^\top + Z_{q,:}W_q W_k^\top P_{k,:}^\top + P_{q,:}W_q W_k^\top Z_{k,:} + P_{q,:}W_q W_k^\top P_{k,:} \end{aligned} \tag{۵}$$

از طرف دیگر، در رمزگذاری نسبی، بردار موقعیت برای پیکسل کوئری تنها به تفاوت موقعیت نسبی $\delta := q - k = (\delta_1, \delta_2) \in \mathbb{Z}^2$ بین خود و پیکسل کلید بستگی دارد.

$$A_{q,k}^{\text{relative}} := Z_{q,:}W_q W_k^\top Z_{k,:}^\top + Z_{q,:}W_q \widetilde{W}_k r_\delta + u^\top W_k Z_{k,:} + v^\top \widetilde{W}_k r_\delta \tag{۶}$$

در اینجا، u و v بردارهای قابل یادگیری هستند و ما یک ماتریس پارامتر جدید $\widetilde{W}_k \neq W_k$ معرفی می‌کنیم. در حالی که انواع مختلفی از رمزگذاری‌های نسبی وجود دارد، در این سوال ما به یک فرم خاص از رمزگذاری نسبی به نام رمزگذاری گاوسی می‌پردازیم که از معادلات زیر تبعیت می‌کند:

$$v^{(h)} := -\alpha^{(h)} \begin{pmatrix} 1 \\ -2\Delta_1^{(h)} \\ -2\Delta_2^{(h)} \end{pmatrix}, \quad r_\delta := \begin{pmatrix} \|\delta\|^2 \\ \delta_1 \\ \delta_2 \end{pmatrix}, \quad W_q = W_k := 0, \quad \widetilde{W}_k := I \tag{۷}$$

ما D_p را به عنوان بُعد هر دو $v^{(h)}$ و r_δ در نظر می‌گیریم. در این حالت، داریم $D_p = 3$. همچنین، $\Delta_1^{(h)}$ و $\Delta_2^{(h)}$ پارامترهای قابل یادگیری هستند.

(ج)

عبارات داده شده در معادله ۷ را در معادله ۶ قرار دهید تا $A_{q,k}^{\text{relative}}$ را ساده کنید.

(د)

هزینه محاسباتی یک لایه توجه تک‌سر با استفاده از رمزگذاری مطلق چقدر است؟ فرض کنید که P قبلاً محاسبه شده است. هزینه محاسباتی چه خواهد بود اگر لایه توجه از رمزگذاری گاوسی استفاده کند؟ از نماد O بزرگ برای مقایسه دو حالت نسبت به $D_x := D_i = D_{\text{hid}} = D_o \gg D_p$ و N استفاده کنید.
ما اکنون به‌طور مختصر ساختار یک کانولوشن را مرور می‌کنیم. با در نظر گرفتن یک تنسور تصویر $Z \in \mathbb{R}^{W \times H \times C_{\text{in}}}$، یک ماتریس وزن $W \in \mathbb{R}^{K \times K \times C_{\text{in}} \times C_{\text{out}}}$، و یک بردار بایاس $b \in \mathbb{R}^{C_{\text{out}}}$، که در آن K اندازه کرنل و C_{in} و C_{out} به ترتیب تعداد کانال‌های ورودی و خروجی هستند، خروجی یک لایه کانولوشنی برای یک پیکسل منفرد $p = (i, j)$ به‌صورت زیر خواهد بود:

$$\text{Convolution}(Z)_{i,j,:} := \left(\sum_{(\delta_1, \delta_2) \in \Delta_K} Z_{i+\delta_1, j+\delta_2, :} W_{\delta_1, \delta_2, :} \right) + b \tag{۸}$$

که در آن Δ_K مجموعه‌ای از تمام جابجایی‌های ممکن است که توسط کرنل کانولوشنی با اندازه K مجاز است.

$$\Delta_K := \left\{ \left[\lfloor -\frac{K}{2} \rfloor, \ldots, \lfloor \frac{K-1}{2} \rfloor \right] \times \left[\lfloor -\frac{K}{2} \rfloor, \ldots, \lfloor \frac{K-1}{2} \rfloor \right] \right\} \tag{۹}$$

در نهایت، هدف ما اثبات قضیه‌ای است که شباهت‌های کلیدی میان قابلیت‌های محاسباتی لایه‌های خودتوجهی و لایه‌های کانولوشنی در یک معماری عصبی نمایان می‌سازد.
قضیه اصلی: یک لایه خود-توجه چند-سر که روی K^2 سر با ابعاد D_o و ابعاد خروجی D_{out} عمل می‌کند، با استفاده از یک رمزگذاری موقعیتی نسبی با ابعاد $D_p \geq 3$، می‌تواند وظیفه هر لایه کانولوشنی با کرنل اندازه $K \times K$ و کانال‌های خروجی D_{out} را ادا کند.
ما اثبات این قضیه اصلی را به اثبات جداگانه دو قضیه فرعی تقسیم می کنیم. قضیه ۱ به شرح زیر است:
قضیه ۱: با فرض اینکه یک لایه خود-توجه چند-سر با $N_h = K^2$ سر و $D_o \geq D_{\text{out}}$ داده شده باشد، فرض کنیم که $f : [N_h] \to \Delta_K$ یک نگاشت یک‌به‌یک بین سرها و جابجایی‌ها است. همچنین فرض می‌کنیم که برای هر سر داریم:

$$\text{softmax}(A^{(h)}_{q,:})_k = \begin{cases} 1 & \text{اگر } f(h) = q - k \\ 0 & \text{در غیر این صورت.} \end{cases} \qquad (۱۰)$$

آنگاه برای هر لایه کانولوشنی با کرنل $K \times K$ و کانال‌های خروجی D_{out}، یک مجموعه از وزن‌ها $\{W_v^{(h)}\}_{h \in N_h}$ وجود دارد که به‌گونه‌ای است که:

$$\text{MultiHead-Self-Attention}(Z) = \text{Convolution}(Z) \quad \text{برای هر تنسور ورودی } Z \in \mathbb{R}^{W \times H \times D_{\text{in}}}$$

(ه)

معادله ۳ را می‌توان به‌صورت زیر نوشت:

$$\text{MultiHead-Self-Attention}(Z) = \sum_{h \in [N_h]} \text{softmax}(A^{(h)}) Z W^{(h)} + b_{\text{out}} \qquad (۱۱)$$

وزن $W^{(h)}$ را به‌صورت تابعی از W_v و W_{out} بیان کنید.
سپس از $W^{(h)}$ برای نوشتن یک عبارت برای

$$\text{MultiHead-Self-Attention}(Z)_{q,:} \in \mathbb{R}^{D_{\text{out}}}$$

استفاده کنید.

(و)

از مفروضات قضیه ۱ استفاده کنید تا $\text{MultiHead-Self-Attention}(Z)_{q,:}$ را به شکلی بیان کنید که معادل یک لایه کانولوشنال طبق تعریف معادله ۸ باشد.

قضیه ۲

ما اکنون باید اثبات کنیم که آیا می‌توانیم یک رمزگذاری موقعیتی خاص بسازیم که مجموعه‌ای از مفروضاتی که در قضیه ۱ از آن‌ها استفاده کردیم، به‌دست آورد. بنابراین، ما تلاش می‌کنیم که قضیه زیر را اثبات کنیم:
قضیه ۲: این امکان وجود دارد که یک رمزگذاری نسبی $\{r_\delta \in \mathbb{R}^{D_p}\}_{\delta \in \mathbb{Z}^2}$ با استفاده از پارامترهای $W_q, W_k, \widetilde{W}_k$ و u بسازیم به طوری که برای هر جابجایی $\Delta \in \Delta_K$، یک بردار v وجود داشته باشد که نگاشت f از معادله ۱۰ را تولید کند.
ما از رمزگذاری گاوسی تعریف‌شده در معادله ۷ استفاده خواهیم کرد و فرض می‌کنیم که معادله زیر برقرار است:

$$A_{q,k} = -\alpha(\|\delta - \Delta\|^2 + c) \qquad (۱۲)$$

(ز)

حد زیر را برای هر دو حالت $\delta = \Delta$ و $\delta \neq \Delta$ حل کنید و توضیح دهید که چرا این معادله ۱۰ را برآورده می‌کند:

$$\lim_{\alpha \to +\infty} \text{softmax}(A_{q,:})_k \tag{۱۳}$$

(ح)

از تعریف رمزگذاری گاوسی استفاده کنید و تعیین کنید که برای کدام مقدار ثابت c مفروضه در معادله ۱۲ برقرار است.
با ترکیب قضیه ۱ و قضیه ۲، ما قضیه اصلی را اثبات کردیم و نشان دادیم که تحت برخی شرایط، یک لایه توجه می‌تواند یاد بگیرد که مانند یک لایه کانولوشنال رفتار کند.
در نهایت، بررسی کنید آیا این نتیجه برای هایپرپارامترهای مختلفی که معمولاً در هنگام بهینه‌سازی شبکه‌های کانولوشنی تنظیم میشوند صادق است یا خیر.

(ط)

برای کدام اندازه‌ها و مقادیر پدینگ خروجی یک لایه خودـتوجه چندسر معادل خروجی یک لایه کانولوشنال است؟

(ی)

آیا یک لایه خودـتوجه چندسر می‌تواند یک کانولوشن با dilations دلخواه را با وجود محدودیت‌های ساختاری مرتبط با تصویر ورودی بیان کند؟ دلیل خود را توضیح دهید.

(ک)

آیا یک لایه خودـتوجه چندسر می‌تواند یاد بگیرد که یک کانولوشن با stride را شبیه‌سازی کند؟ اگر بله، چرا؟ در غیر این صورت، چرا این امکان وجود ندارد؟

بخش عملی (۱۰۰ نمره)

پرسش ۱. پیش‌بینی قیمت نفت خام (۲۵ نمره)

مقدمه

پیش‌بینی سری‌های زمانی یکی از مهم‌ترین و کاربردی‌ترین مباحث در حوزه یادگیری ماشین و تحلیل داده است. در بسیاری از حوزه‌ها از جمله اقتصاد، انرژی، حمل‌ونقل، بازارهای مالی و سلامت، داده‌ها به‌صورت دنباله‌ای از مشاهدات در طول زمان ثبت می‌شوند. هدف از تحلیل سری‌های زمانی، مدل‌سازی رفتار داده‌ها در طول زمان و استفاده از آن برای پیش‌بینی مقادیر آینده است.
در این پروژه، تمرکز شما بر پیش‌بینی قیمت نفت خام خواهد بود؛ یکی از اصلی‌ترین شاخص‌های اقتصادی در جهان. نوسانات قیمت نفت تأثیر گسترده‌ای بر بودجه کشورها، نرخ ارز، تورم و حتی سیاست‌های اقتصادی دارد. بنابراین، ساخت مدلی که بتواند این قیمت را به‌صورت دقیق پیش‌بینی کند، از اهمیت بالایی برخوردار است.
شما باید با استفاده از داده‌های واقعی مربوط به نماد CL=F از پایگاه Yahoo Finance، چند مدل مختلف سری زمانی را پیاده‌سازی کرده و با یکدیگر مقایسه کنید. مدل‌هایی که در این پروژه با آن‌ها کار خواهید کرد شامل RNN، LSTM، GRU و ARIMA هستند. در نهایت باید بتوانید عملکرد این مدل‌ها را ارزیابی کرده و نتایج را با مقاله مرجع مقایسه نمایید.

مرحله اول: دریافت و آماده‌سازی داده

ابتدا باید داده‌های تاریخی قیمت نفت خام را از وب‌سایت Yahoo Finance مربوط به نماد CL=F برای بازه زمانی از سال ۲۰۱۰ تا امروز دانلود کرده و آن را به پروژه وارد کنید.
سپس از ستون Adj Close به عنوان ویژگی اصلی برای پیش‌بینی استفاده کنید. پس از آن، به‌صورت تصادفی ۱۰ درصد از مقادیر داده را حذف کرده و با یکی از روش‌های زیر، مقادیر گمشده را جایگزین کنید:

- استفاده از میانگین متحرک
- پر کردن با مقدار قبلی (forward fill)
- پر کردن با مقدار بعدی (backward fill)

در ادامه، داده‌ها را مطابق با نسبت آموزش به آزمون مشخص‌شده در مقاله (برای مثال ۷۰ درصد آموزش و ۳۰ درصد آزمون) تقسیم‌بندی کنید و سپس آن‌ها را نرمال‌سازی (Scaling) نمایید تا آماده ورود به مدل شوند.
در پایان این مرحله، باید یک هیستوگرام از توزیع قیمت‌ها رسم کنید که مشابه شکل ۶ مقاله مرجع باشد تا نمای کلی از رفتار قیمت نفت به‌دست آورید.

مرحله دوم: پیاده‌سازی مدل‌های یادگیری عمیق

در این بخش باید چهار مدل یادگیری عمیق را پیاده‌سازی و آموزش دهید. مدل‌ها شامل موارد زیر هستند:

- **RNN (Recurrent Neural Network)**
- **LSTM (Long Short-Term Memory)**
- **GRU (Gated Recurrent Unit)**

هر مدل را با استفاده از پایتون و کتابخانه‌هایی مانند TensorFlow یا PyTorch بسازید و طبق تنظیمات مقاله مرجع آن را آموزش دهید. به‌عنوان تابع هزینه از MSE (Mean Squared Error) استفاده کنید.
پس از آموزش مدل‌ها، باید پیش‌بینی آن‌ها را در کنار مقادیر واقعی قیمت نفت رسم کرده و نتایج را مقایسه نمایید.

مرحله سوم: ارزیابی مدل‌ها

در این مرحله باید عملکرد مدل‌ها را با استفاده از چهار معیار خطا ارزیابی کنید:

- **RMSE**
- **MAE**
- **MAPE**
- **R-Squared**

مقادیر این معیارها را برای هر مدل محاسبه کرده و آن‌ها را با یکدیگر و با نتایج مقاله مرجع مقایسه کنید تا متوجه شوید کدام مدل دقت بیشتری در پیش‌بینی قیمت نفت داشته است.

مرحله چهارم: مدل آماری ARIMA

در این بخش باید مدل آماری ARIMA را پیاده‌سازی کنید. ابتدا تفاوت بین ARIMA و SARIMA را به‌طور خلاصه بررسی کنید و سپس فرمول ریاضی ARIMA و نقش پارامترهای (p, d, q) را شرح دهید.
برای یافتن مقادیر بهینه این پارامترها، می‌توانید از نمودارهای ACF و PACF یا تابع auto_arima استفاده نمایید.
پس از تعیین پارامترها، مدل ARIMA را روی داده‌ها آموزش داده و نتایج آن را با سایر مدل‌های یادگیری عمیق و جدول شماره ۶ مقاله مقایسه کنید.

مرحله پایانی: جمع‌بندی

در پایان، باید به‌صورت خلاصه یافته‌های خود را گزارش دهید. بررسی کنید کدام مدل بهترین عملکرد را در پیش‌بینی قیمت نفت داشته است، دلایل آن چه بوده و مزایا و محدودیت‌های هر مدل را توضیح دهید.
این پروژه به شما کمک می‌کند تا به‌صورت عملی با مدل‌سازی سری زمانی، استفاده از یادگیری عمیق و مقایسه آن با روش‌های آماری آشنا شوید. همچنین مهارت کار با داده‌های واقعی، ارزیابی مدل‌ها و تحلیل نتایج علمی را در شما تقویت خواهد کرد.

پرسش ۲. GPT2 From Scratch (۲۵ نمره)

در این تمرین، یک نسخه‌ی بسیار کم حجم از معماری GPT-2 شرکت OpenAI را از صفر (صفر مطلق!) پیاده‌سازی خواهید کرد و سپس روی کامنت‌های اسنپ‌فود آموزش می‌دهید. هدف اصلی تمرین، ایجاد یک مدل زبانی مولد است که بتواند کامنت‌های مصنوعی فارسی با **sentiment کنترل‌پذیر** (مثبت یا منفی) تولید کند.

پرسش ۳. تنظیم دقیق بهینه (۲۵ نمره)

یکی از چالش‌های تنظیم دقیق (Fine-Tuning) مدل‌های زبانی این است که به حافظه و زمان بسیار زیادی نیاز دارند و همچنین اگر بخواهیم مدل را برای وظایف مختلفی آموزش دهیم به ازای هر وظیفه باید یک مدل کامل نگهداری و در محیط production بالا آید. برای حل این مشکلات روش‌های بهینه PEFT(Parameter Efficient Fine-Tuning) معرفی شدند که آموزش مدل‌های بزرگ را در زمان و حافظه کمتری انجام می‌دهند و در واقع وزن‌های کمتری نسبت به مدل کامل آموزش می‌دهند. در نوتبوک peft شما به بررسی این روش‌ها می‌پردازید و یکی از آن‌ها را پیاده‌سازی می‌کنید.

پرسش ۴. استدلال در LLM ها و روش‌های Inference Time Scaling (۲۵ نمره)

در این تمرین، ما درباره استدلال در مدل‌های زبانی بزرگ (LLMs) صحبت می‌کنیم و توانایی استدلال آن‌ها را با استفاده از مجموعه داده‌های ریاضی (Math benchmark) ارزیابی می‌کنیم.
ما برخی از روش‌های inference time scaling را مورد بررسی قرار می‌دهیم که عبارتند از:

- Chain of Thought: روشی که مدل را تشویق می‌کند تا به صورت مرحله به مرحله فکر کند و راه‌حل را توضیح دهد.
- Best of N: انتخاب بهترین پاسخ از میان N پاسخ تولیدشده توسط مدل.
- Beam Search: الگوریتمی برای جستجوی بهترین دنباله‌های ممکن در تولید پاسخ.
- Self-Refinement: فرآیندی که در آن مدل پاسخ خود را بازبینی و بهبود می‌دهد.