فهرست

سوال ۱. بازشناسایی تصاویر با استفاده از ترنسفورمر 4

۱-۱. آماده‌سازی دادگان (۱۰ امتیاز) 5

۱-۲. تقسیم دادگان (۱۰ امتیاز) 5

۱-۳. آموزش مدل ResNet (۱۰ امتیاز) 5

۱-۴. آموزش مدل BotNet (۴۰ امتیاز) 6

۱-۵. بررسی نتایج BotNet (۱۵ امتیاز) 6

۱-۶. آموزش با استفاده از Counterfactual Attention (امتیازی) (۵ امتیاز) 7

۱-۷. تحلیل نتایج (۱۵ امتیاز) 7

نکات 7

سوال ۲. مدل‌های زبانی بزرگ دیفیوژنی 8

۲-۱. سوال‌های تئوری (۳۰ امتیاز) 9

۲-۱-۱. تفاوت autoregressive و masked iterative generation 9

۲-۱-۲. forward masking به عنوان noise model 9

۲-۱-۳. چرا loss باید reweight شود 10

۲-۲. داده، پرامپت و ابزارهای ارزیابی (۲۰ امتیاز) 10

۲-۲-۱. ساخت pipeline داده 10

۲-۲-۲. طراحی prompt به صورت chat format 10

۲-۲-۳. SQL normalization و معیارها 12

۲-۳. بارگذاری مدل و آموزش (۳۰ امتیاز) 12

۲-۳-۱. بارگذاری مدل و توضیح مفاهیم جدید 12

۲-۳-۲. پیاده‌سازی forward masking 12

۲-۳-۳. مرز prompt/answer و ماسک کردن answer 13

۲-۳-۴. loss و training loop 13

۲-۴. تولید متن و ارزیابی (۲۰ امتیاز) 14

۲-۴-۱. پیاده‌سازی block diffusion sampling 14

۲-۴-۲. post-processing برای استخراج SQL 15

۲-۴-۳. evaluation و تحلیل خطا 15
۲-۴-۴. بخش امتیازی (۵ امتیاز) 15

شکل‌ها

شکل 1. نمونه صحیح از قرار دادن HEATMAP بر روی تصویر 6
شکل 2. نمای کلی LLADA در آموزش و تولید. 9

سوال ۱. بازشناسایی تصاویر با استفاده از ترنسفورمر

بازشناسایی (re-identification) تصاویر به معنی تحلیل تصویر و تشخیص یک نمونه از گونه‌ی جانوری است. با استفاده از این تکنیک در دامداری مدرن می‌توان به جای روش‌های سنتی (مانند گوشواره، خالکوبی و داغ زدن)، با استفاده از ویژگی‌های ظاهری دام‌ها را تشخیص داد.

برای بازشناسایی استفاده از شبکه‌های پیچشی[1] و روش‌های بر پایه ترنسفورمر رایج است و با توجه به نوع دادگان مورد استفاده ممکن است یکی بر دیگری برتری داشته باشد. در این تمرین می‌خواهیم دو شبکه عصبی با ساختارهای مختلف را برای این کار با همدیگر مقایسه کنیم.

۱-۱. آماده‌سازی دادگان (۱۰ امتیاز)

برای این تمرین شما باید دادگان مورد استفاده در dataset1 مقاله را از این لینک دریافت کنید.

پس از دریافت دادگان، چند نمونه از تصویرهای موجود را نمایش دهید. سپس تعداد داده موجود در هر کلاس را در یک نمودار نمایش دهید. توضیح دهید با توجه به توزیع کلاس‌ها چه پیش‌پردازش‌هایی ممکن است نتیجه را بهتر کند.

بعد از این که شناخت کلی نسبت به وضعیت دادگان پیدا کردید، باید دادگان را پیش‌پردازش کنید. برای این کار توصیه میشود عکس ها را نرمالایز کنید و همچنین کلاس‌هایی که کم‌تر از ۵ نمونه عکس در آن‌ها موجود است را حذف کنید. شما میتوانید پیش‌پردازش های دیگری که ممکن است برای افزایش دقت شبکه مفید باشد (مانند افزایش داده‌ها[2]) انجام بدهید و در مورد هر کدام توضیح کوتاهی ذکر کنید.

۱-۲. تقسیم دادگان (۱۰ امتیاز)

در مقاله برای تقسیم دادگان از روش Graph Sampling استفاده شده است. در مورد این روش مطالعه کنید و توضیح دهید چرا این روش می‌تواند در مورد این مقاله موثر باشد.

برای این تمرین شما تقسیم دادگان را به صورت عادی و بدون استفاده از Graph Sampling انجام خواهید داد. دادگان را با نسبت ۲/۸ به دو بخش آموزش و آزمون تقسیم کنید و تعداد داده‌ها و کلاس‌های موجود در هر بخش را نشان دهید. میزان داده در هر batch را می‌توانید به ۳۲ یا هر عدد معقولی که سخت‌افزار شما توان پردازش آن را دارد تنظیم کنید.

۱-۳. آموزش مدل ResNet (۱۰ امتیاز)

ا

بتدا در مورد مدل ResNet مطالعه کنید و در مورد ساختار و کاربردهای آن به طور خلاصه توضیح دهید.

[1] Convolutional Neural Networks
[2] Augmentation

سپس مانند مقاله، مدل ResNet50 را با مجموعه دادگان آموزش دهید. برای تابع خطا میتوانید از cross-entropy استفاده کنید.

مدل را حداقل به میزان ۲۰ ایپاک آموزش داده و میزان خطای آموزش و آزمون را در قالب یک نمودار خطی گزارش کنید.

۱-۴. آموزش مدل BotNet (۴۰ امتیاز)

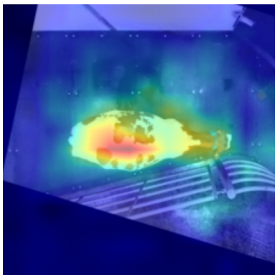
مدل BotNet50 را طبق آنچه در مقاله شرح داده شده طراحی کنید. پس از طراحی مدل از ساختار و لایه‌های آن خروجی بگیرید (در صورتی که به دلیل حجم مدل امکان نشان دادن تمام آن در گزارش نیست، خروجی باید در کد شما باشد و نوشتن تحلیل درستی ساختار و اشاره به وجود ساختار کامل در کد کفایت میکند). در طراحی مدل خروجی لایه توجه[3] را به نحوی ذخیره کنید تا در قسمت بعدی بتوانید آن را بازیابی کنید.

مدل را مانند قسمت قبلی و با همان تابع خطا آموزش دهید و میزان خطای آموزش و آزمون را به ازای هر ایپاک گزارش کنید.

۱-۵. بررسی نتایج BotNet (۱۵ امتیاز)

مشابه مقاله، heatmap خروجی لایه توجه را بر روی چند نمونه از تصاویر دادگان نشان دهید. برای این کار لازم است تصویر عادی را از حالت نرمالایز خارج کنید و خروجی لایه توجه را طبق ابعاد تصویر resize کنید. یک نمونه از خروجی درست در شکل ۱ نشان داده شده است.

با توجه به نتایج به دست آمده آیا مدل شما با استفاده از الگوهای روی پوست گاو تشخیص را انجام میدهد یا محیط اطراف؟ نمونه ای را پیدا کنید که مدل با استفاده از اطلاعات نامربوط تصویر را طبقه‌بندی کرده است.



شکل 1. نمونه صحیح از قرار دادن heatmap بر روی تصویر

[3] Attention

۱-۶. آموزش با استفاده از Counterfactual Attention (امتیازی) (۵ امتیاز)

در این بخش طبق مقاله انجام شده لازم است شما از Counterfactual Attention برای آموزش بهتر مدل استفاده کنید. روش کار این تکنیک به این شکل است که شما باید یک بار خروجی عادی از مدل بگیرید و یک بار دیگر فیچرهای مدل را تا قبل از لایه طبقه‌بندی آن دریافت کنید. سپس این فیچرها را دستکاری کنید و حدود ۳۰ الی ۵۰ درصد آن را ماسک کنید. فیچرهای دستکاری شده را به لایه طبقه‌بندی مدل بدهید و خروجی مدل را استخراج کنید. آنتروپی بین خروجی اول و دوم را حساب کنید و طبق نتیجه به دست آمده خطا را با استفاده از فرمول خطای مقاله حساب کنید و مدل را مجددا آموزش دهید.

پس از آموزش مدل بخش ۱ـ۵ را برای مدل جدید تکرار کنید و تفاوت تصاویر به دست آمده را گزارش کنید.

۱-۷. تحلیل نتایج (۱۵ امتیاز)

نتایج مدل ترنسفورمر و مدل پیچشی را با همدیگر مقایسه کنید. در این دادگان کدامیک از مدل ها بهتر عمل کرده است؟ آیا نتیجه ی به دست آمده با انتظارات شما هم‌خوانی دارد؟

توضیح دهید در چه شرایطی مدلی که در این آزمایش ضعیف ترعمل کرده می‌توانست عملکرد بهتری داشته باشد؟

در صورتی که دادگان شما متوازن نبودند، بررسی کنید که برای کلاس‌هایی که داده‌های کمتری داشتند تفاوت معنی‌داری بین دقت خروجی دو مدل مشاهده می‌شود یا خیر.

نکات

- توجه کنید که برای مقایسه درست، مدل‌های ResNet مورد استفاده در سوال را بدون وزن‌های اولیه بارگذاری کنید.
- در صورتی که آموزش به درستی انجام گیرد انتظار می‌رود مدل‌ها با دقت بالای ۹۰ درصد بتوانند عکس‌ها را بازشناسایی کنند.
- آموزش یک ایپاک هر دو مدل ResNet و BotNet با استفاده از هایپرپارامترهای پیشنهاد شده و استفاده از GPU T4 گوگل کولب، حدود ۳ الی ۵ دقیقه زمان می‌برد.

سوال ۲. مدل‌های زبانی بزرگ دیفیوژنی

تا اینجا در درس، بیشتر با مدل‌های زبانی‌ای کار کرده‌اید که متن را چپ به راست و توکن به توکن تولید می‌کنند. این خانواده را معمولاً autoregressive می‌نامند که مدل در هر قدم توکن بعدی را با توجه به توکن‌های قبلی پیش‌بینی می‌کند. در این تمرین قرار است با یک ایده‌ی متفاوت و مهم کار کنید: تولید متن با پر کردن جای خالی و اصلاح تدریجی.

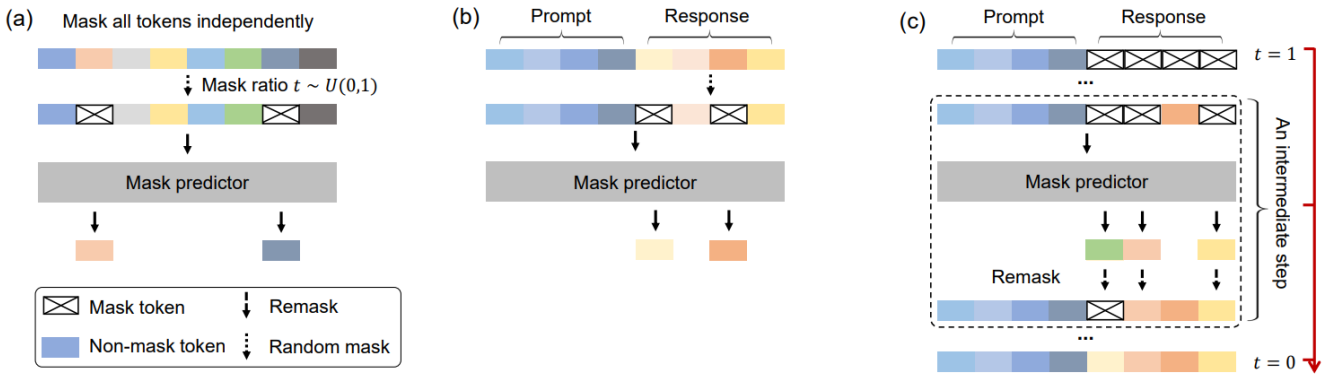
مدلی که در این تمرین از آن استفاده می‌کنیم LLaDA است؛ یک مدل زبانی که به‌جای next-token prediction، یاد می‌گیرد وقتی بعضی توکن‌ها را با [MASK] پنهان می‌کنیم، tokenهای درست را حدس بزند. در زمان inference هم به‌جای تولید خطی، از یک خروجی کاملاً ماسک شده شروع می‌کنیم و در چند مرحله به صورت تکراری توکن‌ها را پر می‌کنیم تا پاسخ نهایی ساخته شود. این تمرین diffusion را در یک نسخه‌ی گسسته و ساده می‌بینید که عملا همان masking + iterative filling است. تاکید می‌شود مقاله LLaDA و مخزن گیتهاب آن را مطالعه کرده و سپس به انجام تمرین بپردازید.

کاربرد این تمرین بر مسئله Text-to-SQL است: SQL یک زبان استاندارد برای کار با پایگاه‌داده‌های رابطه‌ای است که با آن می‌توانیم روی جدول‌ها عملیات انجام دهیم؛ مثلاً داده‌ها را جست‌وجو کنیم (SELECT)، فیلتر کنیم (WHERE)، مرتب‌سازی کنیم (ORDER BY)، یا چند جدول را با هم ترکیب کنیم (JOIN). در این تمرین معمولاً هدف این است که با توجه به ساختار جدول‌ها و ستون‌ها، یک query درست بنویسیم تا پاسخ سؤال از داخل داده‌ها استخراج شود.

در این تمرین شما به مدل یک schema از پایگاه داده و یک question می‌دهید و مدل باید فقط و فقط SQL query را تولید کند. هدف نهایی شما این است که یک pipeline کامل بسازید، از آماده سازی داده و prompt، تا آموزش (Supervised Fine-tuning) با یک loss مخصوص، تا generation با یک الگوریتم مرحله‌ای (block diffusion)، و در نهایت evaluation با معیار Exact Match .

در مسیر پیاده‌سازی، با دو مفهوم کاربردی هم آشنا می‌شوید: LoRA (از خانواده‌ی PEFT) و 4-Bit Quantization. این‌ها راه‌هایی هستند برای اینکه بتوانید یک مدل بزرگ را با GPU محدود هم fine-tune کنید. لازم نیست وارد جزئیات ریاضی این روش‌ها شوید، اما باید بفهمید چه کاری انجام می‌دهند و چگونه به کاهش هزینه کمک می‌کنند.

ابزارهای مجاز و محدودیت‌ها: می‌توانید از Python، PyTorch و کتابخانه‌های Hugging Face مثل transformers و datasets استفاده کنید. دیتاست مد نظر برای این تمرین gretelai/synthetic_text_to_sql است. همچنین در این تمرین بایستی از LoRA و 4-bit loading استفاده کنید. دیدن کد رسمی مقاله (مخصوصا فایل generate.py) برای ایده گرفتن مجاز است، اما باید نسخه‌ی خودتان را پیاده‌سازی کنید و در گزارش توضیح بدهید.



شکل 2. نمای کلی LLaDA در آموزش و تولید. (الف) در pre-training برای هر نمونه، یک عدد تصادفی t بین ۰ و ۱ انتخاب می‌شود که نشان می‌دهد تقریباً چه درصدی از توکن‌ها باید با [MASK] جایگزین شوند؛ سپس مدل (mask predictor) تلاش می‌کند توکن‌های ماسک‌شده را پیش‌بینی کند. (ب) در SFT همین ایده استفاده می‌شود، با این تفاوت که فقط توکن‌های بخش response ممکن است ماسک شوند و prompt دست‌نخورده می‌ماند. (ج) در sampling تولید از حالت کاملاً ماسک‌شده شروع می‌شود (t=1) و به‌تدریج با پیش‌بینی همزمان ماسک‌ها و remask کردن برخی توکن‌های نامطمئن، به حالت بدون ماسک می‌رسد (t=0) و خروجی نهایی ساخته می‌شود.

۲-۱. سوال‌های تئوری (۳۰ امتیاز)

هدف این بخش این است که با مطالعه دقیق مقاله مفاهیم کلیدی را بفهمید. پاسخ‌ها باید کوتاه و دقیق باشند و بیشتر بر شهود روش تکیه کنند.

۲-۱-۱. تفاوت autoregressive و masked iterative generation

در یک پاراگراف توضیح دهید یک مدل autoregressive چگونه متن را تولید می‌کند و یک مدل masked-prediction چگونه آموزش می‌بیند و چگونه تولید می‌کند. سپس در یک پاراگراف کوتاه دیگر، برای Text-to-SQL یک مزیت و یک محدودیت برای روش masked iterative generation بیان کنید. مثال بزنید و بیان بدارید فرضا اگر یک تصمیم اشتباه در ابتدای تولید رخ دهد، در هر روش چه اتفاقی می‌افتد؟

۲-۱-۲. forward masking به عنوان noise model

فرض کنید یک توالی تمیز دارید. توضیح دهید با یک احتمال ماسک کردن P چگونه نسخه‌ی نویزی آن را می‌سازید (یعنی بعضی tokenها را [MASK] می‌کنید). بعد توضیح دهید چرا خوب است P در آموزش ثابت نباشد و گاهی کم و گاهی زیاد باشد. در نهایت بگویید اگر همیشه یک مقدار ثابت مثل ۱۵٪ استفاده کنید، چه مشکل‌هایی ممکن است پیش بیاید.

۲-۱-۳. چرا loss باید reweight شود

در این تمرین برای tokenهای ماسک‌شده loss را با یک ضریب مربوط به احتمال ماسک کردن reweight می‌کنید. توضیح دهید چرا اگر این کار را نکنید، بعضی حالت‌ها (مثل زمانی که تعداد کمی token ماسک شده) کمتر در یادگیری اثر می‌گذارند. هدف این reweight کردن چیست و چه تعادلی را برقرار می‌کند؟

۲-۲. داده، پرامپت و ابزارهای ارزیابی (۲۰ امتیاز)

۲-۲-۱. ساخت pipeline داده

دیتاست gretelai/synthetic_text_to_sql را load کنید و مجموعه‌های آموزش، اعتبار سنجی و آزمایش را بسازید. اندازه‌ی هر split را گزارش کنید. همچنین میانگین (یا دیگر آماره‌ها) را برای طول schema و طول question را (بر حسب کاراکتر یا token) در train گزارش دهید. همچنین توجه کنید درمراحل بعدی پیاده سازی حداقل تعداد نمونه آموزشی 1000 و حداقل تعداد ایپاک 1 باشد. برای تعداد نمونه های اعتبارسنجی و آزمایش هم حداقل از 300 نمونه استفاده کنید.

همچنین در گزارش کوتاه توضیح دهید وقتی schema خیلی طولانی است چه روشی به‌کار می‌برید و همان روش را در کد اجرا کنید. برای این تمرین لازم نیست راه‌حل پیچیده طراحی کنید، اما یک تصمیم مشخص و قابل‌تکرار داشته باشید. ساده‌ترین کار این است که با تنظیم MAX_LEN و فعال‌کردن truncation=True اجازه دهید tokenizer متن را کوتاه کند؛ در این حالت مهم است که بررسی کنید پاسخِ SQL از انتهای نمونه حذف نشده باشد و واقعاً برای بخش پاسخ جا باقی بماند. اگر truncation ساده باعث می‌شود بخش‌های مهم schema حذف شود، می‌توانید قبل از ساخت prompt، schema را با یک قانون ساده کوتاه کنید؛ مثلاً فقط بخشی از schema را نگه دارید (مثل ابتدای آن) یا اطلاعات اضافی را حذف کنید و فقط نام جدول‌ها و ستون‌ها را باقی بگذارید. در هر حال، انتظار می‌رود در کدتان یک کنترل ساده اضافه کنید تا اگر طول prompt بیش از حد بزرگ شد و عملاً جایی برای SQL باقی نماند، آن نمونه را مدیریت کنید (مثلاً کوتاه‌سازی بیشتر، یا حذف نمونه، یا تنظیم طول‌ها) و در گزارش توضیح دهید چرا این تصمیم را گرفته‌اید.

۲-۲-۲. طراحی prompt به صورت chat format

یک قالب prompt طراحی کنید که شامل یک system message باشد تا مدل را مجبور کند «فقط SQL» خروجی بدهد و یک user message که schema و question را شامل می‌شود. در مرحله‌ی آموزش، باید یک assistant message هم اضافه کنید که داخل آن gold SQL قرار می‌گیرد. برای اینکه قالب پیام‌ها دقیقاً مطابق قالب چت مدل ساخته شود، از تابع apply_chat_template در tokenizer استفاده کنید؛ این تابع لیستی از پیام‌ها (با نقش‌های system، user و assistant) را به یک متن نهایی تبدیل می‌کند که همان چیزی است که واقعاً به مدل داده می‌شود. شما باید بعد از اعمال قالب، یک نمونه‌ی واقعی از متن نهایی را چاپ کنید (متنی که apply_chat_template تولید کرده) و در گزارش توضیح دهید چگونه مطمئن می‌شوید که SQL فقط در بخش

assistant قرار می‌گیرد؛ چون در آموزش و در تعریف مرز prompt/answer، باید دقیقاً بدانید پاسخ از کجا شروع می‌شود و قرار است فقط بخش پاسخ نویزی شود.

برای نمونه، می‌توانید ساختار پیام‌ها را شبیه زیر در نظر بگیرید. توجه کنید که این نمونه‌کد فقط یک مثال است و شما می‌توانید متن‌ها را بهتر کنید:

```
SYSTEM_PROMPT = (
    "You are a Text-to-SQL assistant. Output ONLY the SQL query. "
    "Do not add explanations."
)

user_content = (
    "Schema:\n"
    "table students(id, name, age)\n"
    "table enrollments(student_id, course_id)\n\n"
    "Question:\n"
    "List the names of students older than 20.\n\n"
)

messages_train = [
    {"role": "system", "content": SYSTEM_PROMPT},
    {"role": "user", "content": user_content},
    {"role": "assistant", "content": "SELECT name FROM students WHERE age>20;"},
]

text_final = tokenizer.apply_chat_template(
    messages_train,
    add_generation_prompt=False,
    tokenize=False,
)

print(text_final)
```

همچنین لازم است نسخه‌ی prompt بدون جواب را هم بسازید؛ یعنی همان system و user، به‌همراه هدرِ assistant اما بدون محتوای SQL. این نسخه برای این است که مرز شروع پاسخ (جایی که SQL باید تولید شود) دقیق مشخص شود و بعداً در آموزش فقط همان بخشِ پاسخ نویزی/ماسک شود. در گزارش توضیح دهید با توجه به معیارExact Match چرا system message و دستور "output only SQL" برای evaluation مهم است.

۲-۲-۳. SQL normalization و معیارها

یک تابع SQL normalization بنویسید که برای normalized Exact Match استفاده شود. حداقل باید query را lowercase کند، فاصله‌ها را یکسان کند و کاراکتر ; انتهایی را حذف کند. می‌توانید quote و backtick را هم حذف کنید. سپس دو تابع ارزیابی raw Exact Match و normalized Exact Match را پیاده‌سازی کنید. در گزارش یک مثال از محدودیت این معیارها بزنید: مثلا دو query ممکن است از نظر معنا برابر باشند اما از نظر متن برابر نباشند.

۲-۳. بارگذاری مدل و آموزش (۳۰ امتیاز)

در این بخش شما LLaDA را برای Text-to-SQL، fine-tune می‌کنید، که در این مسیر بایستی از LoRA و 4-bit loading استفاده کنید.

۲-۳-۱. بارگذاری مدل و توضیح مفاهیم جدید

مدل GSAI-ML/LLaDA-8B-Instruct و tokenizer آن را load کنید. باید [MASK] token id و EOS id را مشخص کنید و مطمئن شوید pad_token_id برابر mask_id نیست. همچنین باید use_cache=False تنظیم شود.

در گزارش با زبان ساده توضیح دهید 4-bit quantization یعنی چه و چرا به کاهش حافظه کمک می‌کند. سپس توضیح دهید LoRA چیست و چرا به جای آموزش همه‌ی پارامترها، فقط چند پارامتر اضافه آموزش می‌دهیم. در نهایت توضیح دهید KV cache چیست و چرا KV cache در این سبک generation مثل autoregressive کاربرد ندارد.

۲-۳-۲. پیاده‌سازی forward masking

تابعی بنویسید که یک batch از input_ids (یعنی همان توکن‌های تمیز) را بگیرد و از آن یک نسخه‌ی نویزی/ماسک‌شده بسازد؛ طوری که مدل مجبور شود جای خالی‌ها را پر کند. ایده این است که برای هر نمونه در batch، یک عدد تصادفی بین ۰ و ۱ انتخاب می‌کنید (به آن می‌گوییم t). این عدد تعیین می‌کند تقریباً چه سهمی از توکن‌های آن نمونه باید مخفی شوند؛ مثلا اگر t نزدیک ۰/۱ باشد یعنی تعداد کمی از توکن‌ها ماسک می‌شوند و اگر نزدیک ۰/۸ باشد یعنی بیشتر توکن‌ها ماسک می‌شوند. بعد برای هر موقعیت از توکن‌ها، به‌صورت مستقل تصمیم می‌گیرید که آیا آن توکن ماسک شود یا نه.

در خروجی این تابع باید سه چیز برگردانید: noisy_batch که همان input_ids است ولی بعضی جاهایش [MASK] شده، masked_indices که یک ماسک بولین هم‌ابعاد توکن‌هاست و نشان می‌دهد دقیقاً کدام موقعیت‌ها ماسک شده‌اند و p_mask که همان احتمال ماسک کردن است (برای هر نمونه و در عمل برای هر موقعیت هم قابل استفاده است) تا بعدا در محاسبه‌ی loss از آن برای وزن دهی استفاده کنید. هدف این است که با داشتن masked_indices دقیقا بدانید loss را کجا حساب کنید و با داشتن p_mask بتوانید اثر نمونه‌هایی که کم ماسک یا پر ماسک هستند را متعادل کنید.

۲-۳-۳. مرز prompt/answer و ماسک کردن answer

Dataset شما باید برای هر نمونه هم input_ids بدهد و هم prompt_length. روش پیشنهادی این است که دو بار tokenize کنید: یک بار فقط prompt + assistant header (بدون محتوا) و یک بار prompt + answer کامل. طول حالت اول می‌شود prompt_length. سپس در آموزش، هرچند forward_process ممکن است ابتدا همه‌جا ماسک کند، شما باید prompt را restore کنید تا prompt همیشه تمیز بماند و فقط پاسخ نویزی شود. در گزارش توضیح دهید چرا نباید خود prompt را نویزی کرد.

۲-۳-۴. loss و training loop

در این تمرین، هدف loss این نیست که مدل را مجبور کنیم همه‌ی توکن‌ها را دوباره تولید کند، بلکه فقط می‌خواهیم مدل جای خالی‌ها را پر کند. بنابراین loss شما باید فقط روی توکن‌هایی حساب شود که واقعاً ماسک شده‌اند و آن هم فقط در بخش پاسخ (response). یعنی اگر یک توکن در prompt ماسک نشده یا حتی اگر در پاسخ اصلا ماسک نشده، نباید در loss وارد شود. این دقیقاً همان چیزی است که masked_indices به شما می‌دهد: یک نقشه که نشان می‌دهد کدام موقعیت‌ها ماسک شده‌اند. بعد از اینکه prompt را دست‌نخورده نگه داشتید، باید یک ماسک نهایی بسازید که فقط ماسک‌های داخل پاسخ را نگه دارد؛ loss را هم فقط روی همین موقعیت‌ها بگیرید.

حالا نکته‌ی مهم این است که شما در forward masking، برای هر نمونه یک احتمال ماسک (مثل p_mask) دارید که می‌تواند کوچک یا بزرگ باشد. اگر p_mask کوچک باشد، تعداد توکن‌های ماسک‌شده کم می‌شود و به‌صورت طبیعی سیگنال آموزشی آن نمونه کمتر خواهد بود. اگر بدون اصلاح جلو بروید، مدل بیشتر از نمونه‌هایی یاد می‌گیرد که مقدار زیادی از توکن‌ها در آن‌ها ماسک شده‌اند و حالت‌های کم‌ماسک کم اثر می‌شوند؛ در حالی که مدل باید هم یاد بگیرد وقتی اطلاعات زیادی دارد (کم ماسک)، هم وقتی اطلاعات کمی دارد (پر ماسک). به همین دلیل loss هر توکن ماسک‌شده را با یک ضریب مرتبط با احتمال ماسک reweight می‌کنیم (به زبان ساده: توکن‌های ماسک شده در نمونه‌هایی که ماسک کردن در آن‌ها کم بوده، وزن بیشتری می‌گیرند تا اثرشان در آموزش از بین نرود). شما باید دقیقاً مشخص کنید چه reweight استفاده کرده‌اید و چرا.

بعد از محاسبه‌ی loss روی توکن‌های ماسک‌شده، باید تصمیم بگیرید چگونه آن را نرمال کنید تا مقیاس loss بین batchها قابل مقایسه و پایدار بماند. دو راه رایج وجود دارد. یک راه این است که مجموع loss را نسبت به تعداد کل توکن‌های batch نرمال کنید (یعنی چیزی شبیه میانگین‌گیری نسبت به batch_size × seq_len) تا تغییر تعداد توکن‌های ماسک‌شده باعث بالا و پایین شدن شدید loss نشود. راه دیگر این است که فقط روی تعداد توکن‌های واقعاً ماسک‌شده نرمال کنید (میانگین روی maskها) که ساده‌تر است ولی ممکن است با تغییر تعداد ماسک‌ها، مقیاس loss نوسان بیشتری داشته باشد. شما باید یکی از این دو را انتخاب کنید و در گزارش توضیح دهید چرا این انتخاب را کرده‌اید و چه اثری روی پایداری آموزش دارد.

یک نکته‌ی عملی مهم هم وجود دارد: ممکن است در یک batch، به‌طور تصادفی هیچ توکنی از پاسخ ماسک نشود (خصوصاً وقتی p_mask کوچک است یا پاسخ کوتاه است). در این حالت اگر شما مستقیم بخواهید loss را روی مجموعه خالی حساب کنید، کد ممکن است crash کند یا NaN تولید کند. بنابراین باید برای این edge case یک رفتار مشخص داشته باشید؛ مثلاً می‌توانید آن batch را رد کنید، یا دوباره ماسک‌کردن را انجام دهید، یا یک

شرط بگذارید که حداقل یک تعداد ماسک در پاسخ وجود داشته باشد. مهم این است که کدتان پایدار باشد و گزارش کنید چه راهی را انتخاب کرده‌اید.

در نهایت باید خودتان تصمیم بگیرید با padding چه کنید. چون در DataLoader معمولاً توالی‌ها را هم‌طول می‌کنیم، بخشی از توکن‌های انتهایی ممکن است فقط برای پرکردن طول باشد. در این تمرین دو تصمیم معقول وجود دارد: یا EOS padding را مثل توکن واقعی نگه دارید (به این معنا که مدل یاد بگیرد بعد از پایان پاسخ، EOSهای بعدی هم قابل پیش‌بینی‌اند)، یا padding را به طور کامل از محاسبه loss حذف کنید تا مدل صرفاً روی محتوای واقعی پاسخ تمرکز کند. هر دو قابل قبول‌اند، اما باید انتخابتان را شفاف توضیح دهید و نشان دهید در loss دقیقا چگونه این تصمیم را اعمال کرده‌اید.

توجه کنید که training loop شما باید optimizer، gradient clipping و scheduler داشته باشد. همچنین hyperparameterها را گزارش کنید و با زبان ساده بگویید هرکدام چه اثری دارد.

۲-۴. تولید متن و ارزیابی (۲۰ امتیاز)

۲-۴-۱. پیاده‌سازی block diffusion sampling

یک تابع generation بنویسید که ورودی‌اش یک prompt tokenized باشد و خروجی‌اش یک SQL تولیدشده. ایده‌ی اصلی این نوع تولید این است که مثل مدل‌های autoregressive از چپ به راست نمی‌نویسیم؛ بلکه ابتدا یک دنباله می‌سازیم که شامل خود prompt است و بعد از آن به اندازه‌ی gen_length توکن [MASK] قرار می‌دهیم. یعنی عملا پاسخ را در ابتدا کاملا نامعلوم فرض می‌کنیم و بعد قرار است در چند مرحله آن را پر کنیم. در هر مرحله مدل را روی همین دنباله اجرا می‌کنید تا برای تمام موقعیت‌ها (به‌خصوص جاهایی که هنوز [MASK] هستند) توزیع احتمال روی واژگان بدهد، و شما بتوانید برای هر موقعیت ماسک‌شده یک توکن پیشنهادی انتخاب کنید.

در ادامه باید block diffusion را پیاده‌سازی کنید تا تولید هم کنترل‌پذیرتر باشد و هم سریع‌تر. یعنی ناحیه پاسخ (همان gen_length توکن بعد از prompt) را به چند block با طول block_length تقسیم می‌کنید و تولید را block به block انجام می‌دهید. منظور از این کار این است که در هر بازه از مراحل، فقط اجازه دارید توکن‌های همان block فعلی را قطعی کنید و بقیه‌ی blockهای بعدی یا دست‌نخورده بمانند یا اصلا وارد تصمیم‌گیری نشوند. به این ترتیب مدل اول روی ساختن بخش ابتدایی پاسخ تمرکز می‌کند و بعد سراغ بخش‌های بعدی می‌رود، بدون اینکه همه چیز را از اول روی کل پاسخ هم زمان قفل کند.

در هر step داخل یک block، باید این چرخه را اجرا کنید: ابتدا برای همه‌ی موقعیت‌های ماسک‌شده توکن‌های کاندید پیشنهاد دهید (مثلاً با انتخاب بهترین توکن یا با sampling کنترل‌شده)، سپس برای هر موقعیت یک عدد به عنوان confidence بسازید که نشان بدهد مدل چقدر به پیشنهادی که داده مطمئن است. بعد به‌جای اینکه همه‌ی ماسک‌ها را یک‌باره پر کنید، فقط تعداد محدودی از موقعیت‌ها را که confidence بالاتری دارند انتخاب می‌کنید و آن‌ها را commit می‌کنید (یعنی [MASK] آن‌ها را واقعاً با توکن پیشنهادی جایگزین می‌کنید). بقیه‌ی موقعیت‌ها هنوز [MASK] باقی می‌مانند تا در stepهای بعدی، با داشتن زمینه‌ی بیشتر (توکن‌هایی که تازه commit شده‌اند)،

دوباره پیش‌بینی شوند و اصلاح شوند. این کم‌کم قطعی کردن باعث می‌شود مدل بتواند اشتباهات اولیه را بهتر جبران کند و مثل یک فرایند پالایش تدریجی عمل کند.

تعریف confidence را شما باید خودتان انتخاب کنید و در گزارش توضیح دهید چرا. یک انتخاب رایج این است که بعد از محاسبه‌ی softmax، احتمال همان توکنی را که برای یک موقعیت انتخاب کرده‌اید به عنوان confidence بگیرید (هرچه احتمال بالاتر، اطمینان بیشتر). اما می‌توانید معیارهای دیگری هم پیشنهاد دهید، به شرط اینکه روشن توضیح دهید چگونه محاسبه می‌شود و چرا فکر می‌کنید برای Text-to-SQL مناسب است. همچنین باید مشخص کنید در هر step دقیقا چند توکن را commit می‌کنید (یا اینکه این تعداد را چگونه بین stepها پخش می‌کنید) و چرا این انتخاب روی کیفیت/سرعت اثر دارد.

۲-۴-۲. post-processing برای استخراج SQL

تابعی بنویسید که خروجی متن را تمیز کند به این صورت که فقط قسمت مربوط به SQL را نگه دارد. می‌توانید متن قبل از اولین کلیدواژه‌ی SQL را حذف کنید (مثل SELECT، WITH، UPDATE)، و اگر لازم است، متن را در اولین حضور کاراکتر ; قطع کنید. در گزارش توضیح دهید چرا ممکن است مدل خروجی‌های اضافی یا junk تولید کند.

۲-۴-۳. evaluation و تحلیل خطا

روی حداقل ۲۰۰ نمونه از test، normalized EM را گزارش کنید. همچنین 10 نمونه خروجی مدل به همراه خروجی اصلی را نشان دهید و توضیح بدهید خطاهای رایج چه هستند.

۲-۴-۴. بخش امتیازی (۵ امتیاز)

پنج ایده‌ی جالب، اصولی و خلاقانه که به ذهن‌تان می‌رسد برای بهبود روش یا نتایج مطرح‌شده در این مقاله پیشنهاد دهید. برای هر ایده، یک پاراگراف کوتاه بنویسید و توضیح دهید دقیقاً چه تغییری پیشنهاد می‌کنید، چرا فکر می‌کنید می‌تواند مفید باشد، و انتظار دارید چه اثری روی کیفیت، سرعت یا پایداری مدل داشته باشد.