

پروژه آموزش موازی مدل (parameter fine tuning using lora) برای دیتاست multi intent detection

استاد درس: دکتر سدید پور

برای انجام این پروژه چندین کار انجام شده است که به چندین روش مختلف سعی کردیم که این هدف دسته بندی intend مخاطب را داشته باشیم ولی در نهایت روش اولی که انجام داده ام به نظر مطابق با آخرین توضیحاتی که از شما گرفته ام می باشد.

# روش اول( استخراج اطلاعات از طریق چت بات های برخط و استفاده از lora برای parameter efficient fine tuning):

برداشت من این بوده است که قرار از استخراج دیتا از طریق مدل های زبانی بزرگ به صورت آنلاین داشته باشیم که من مدل های مختلف را برای این موضوع انتخاب می کنم و سپس با توجه به این استخراج ها از مدل های مختلف از ترکیب چندین مدل برای بهبود کیفیت استفاده می کنم که نتایج یکسان را از مدل های مختلف را به صورت یکپارچه داشته باشم و سپس از این دیتا ها برای Parameter-Efficient Fine-tuning استفاده می کنم.

## ۱-استخراج داده

پرامپتی که من خودم برای این موضوع در نظر داشتم به صورت زیر بوده است:  
"

لطفاً یک لیست جامع از بیماری‌های شایع به همراه موارد زیر ارائه دهید:

- نام بیماری

- علائم اصلی

- روش‌های تشخیص

- درمان‌های رایج

- عوارض احتمالی

پاسخ را در قالب JSON ارائه دهید.

"  
اما این را به مدل sonnet 4 دادم و از اون خواستم که این پرامپ را به صورتی بهبود دهد که جواب جامع تر و بهتری بگیرم که پرامپت زیر را به من پیشنهاد داد:

"لطفاً یک لیست جامع از 20-25 بیماری شایع در سطح جهانی ارائه دهید که شامل بیماری‌های عفونی، غیر عفونی، مزمن و حاد باشد. برای هر بیماری اطلاعات زیر را به صورت دقیق و علمی ارائه دهید:

- نام بیماری (به فارسی و انگلیسی)

- دسته‌بندی بیماری (عفونی/غیر عفونی، حاد/مزمن)

- علائم اصلی (حداقل 3-5 علامت مهم)

- روش‌های تشخیص (آزمایش‌های تشخیصی، معاینات بالینی)

- درمان‌های رایج (دارویی، غیر دارویی، جراحی در صورت نیاز)

- عوارض احتمالی (کوتاه‌مدت و بلندمدت)

- راه‌های پیشگیری (در صورت وجود)

- میزان شیوع جهانی (درصد یا آمار کلی)

پاسخ را در قالب JSON با ساختار زیر ارائه دهید:

```json

{

"diseases": [

{

"name\_persian": "",

"name\_english": "",

"category": "",

"symptoms": [],

"diagnosis\_methods": [],

"treatments": [],

"complications": [],

"prevention": [],

"prevalence": ""

}

]

}

لطفاً اطلاعات علمی و به‌روز ارائه دهید و از منابع معتبر پزشکی استفاده کنید.

"

چیز که مد نظر من بود به این صورت است که این پرامپت را هم به سه مدل مختلف بدهم و خروجی ها را سپس به یک مدل بدهم و این ها را ترکیب کند و موارد اضافی را حذف کند پس ابتدا به مدل gpt 4.1 دادم و جوابی که گرفتم در فایل gpt41\_disease.json ذخیره شده است و سپس به claud sonnet 4 داده شده است و جواب خروجی در فایل sonnet4\_disease.json قرار داده شده است و سپس به gpt 2.5 pro داده شده است و جواب در فایل gpt25 pro disease.json قرار داده شده است.

2- ترکیب چندین مدل برای بهبود کیفیت:

برای این منظور از gemini 2.5 شده است چون context window بزرگتری دارد پس پرامپ زیر را خود من نوشته ام و برای بهبود به claud sonnet 4 داده ام و از این مدل خواستم که آن را بهبود دهد:

"

من از چند مدل زبانی خواسته ام که موارد زیر را:

1. تمام بیماری‌های مرتبط

2. علائم و نشانه‌ها

3. روش‌های تشخیص

4. گزینه‌های درمانی

برای من در قالب json در قالب زیر ارائه دهند:

```json

{

"diseases": [

{

"name\_persian": "",

"name\_english": "",

"category": "",

"symptoms": [],

"diagnosis\_methods": [],

"treatments": [],

"complications": [],

"prevention": [],

"prevalence": ""

}

]

}‍‍‍

```

حال من حاصل کار این سه مدل را در سه فایل json با اسم های gpt25pro\_disease.json، sonnet4\_disease.json و gpt 41 disease.json در فایل های attachment ارسال کرده ام

از تو می خوام دقیقا با همین ساختار پاسخ این چت بات ها را با هم ترکیب کنی به این صورت که در صورتی که یکی بیماری در پاسخ یکی وجود نداشت و در دیگری وجود داشت کل اون پاسخ را نگه داری و اگر بیماری مشترکی بود اطلاعات آن ها را با هم ترکیب کنی

"

این پاسخ را به همان مدل sonnet 4 که به عنوان agent اصلاح کننده پرامپ در نظر گرفته بودم دادم و پرامپت اصلاح شده زیر را به من داد:

"

من از چند مدل زبانی مختلف درخواست جمع‌آوری اطلاعات جامع بیماری‌ها را کرده‌ام. هر مدل اطلاعات زیر را در قالب JSON ارائه داده است:

1. \*\*تمام بیماری‌های مرتبط\*\* (با تنوع کامل)

2. \*\*علائم و نشانه‌ها\*\* (شامل علائم اولیه، ثانویه و پیشرفته)

3. \*\*روش‌های تشخیص\*\* (آزمایش‌ها، تصویربرداری، معاینات بالینی)

4. \*\*گزینه‌های درمانی\*\* (دارویی، غیر دارویی، جراحی، طب مکمل)

\*\*فایل‌های ارسالی:\*\*

- `gpt25pro\_disease.json`

- `sonnet4\_disease.json`

- `gpt41\_disease.json`

\*\*ساختار JSON مورد استفاده:\*\*

```json

{

"diseases": [

{

"name\_persian": "",

"name\_english": "",

"category": "",

"symptoms": [],

"diagnosis\_methods": [],

"treatments": [],

"complications": [],

"prevention": [],

"prevalence": ""

}

]

}

درخواست من از شما:

لطفاً این سه پاسخ را دقیقاً با همین ساختار JSON ترکیب کنید بر اساس قوانین زیر:

بیماری‌های منحصربه‌فرد: اگر بیماری فقط در یک فایل وجود دارد، کل اطلاعات آن را حفظ کنید

بیماری‌های مشترک: برای بیماری‌هایی که در چند فایل وجود دارند:

نام بهترین ترجمه فارسی و انگلیسی را انتخاب کنید

تمام علائم، روش‌های تشخیص، درمان‌ها، عوارض و روش‌های پیشگیری را بدون تکرار ترکیب کنید

جامع‌ترین اطلاعات دسته‌بندی و شیوع را حفظ کنید

کیفیت نهایی:

هیچ اطلاعاتی حذف نشود

تکرارها حذف شوند

اطلاعات کامل‌تر و دقیق‌تر اولویت داشته باشند

خروجی نهایی: یک فایل JSON واحد با تمام بیماری‌ها به ترتیب حروف الفبا

لطفاً فرآیند ترکیب را گام به گام انجام دهید و نتیجه نهایی را ارائه دهید.

"

حال این پاسخ را به gemini 2.5 pro دادم و تاکید کردن که محدودیت هایی که بابت کاهش تعداد توکن در پاسخ می شود را در نظر نگیرد و اگر به نظرش پاسخ بیشتر از یک پاسخ را شامل می شود آن را به دو بخش تبدیل کند و به من بگوید که دوباره بهش درخواست بدم تا ادامه جواب را بدهد

فایل json نهایی در فایل merged diseases.json قرار دارد که نمونه خروجی آن برای یک بیماری به صورت زیر است:

{

"name\_persian": "آسم",

"name\_english": "Asthma",

"category": "غیرعفونی، مزمن",

"symptoms": [

"تنگی نفس",

"سرفه",

"خس‌خس سینه",

"احساس فشار در قفسه سینه",

"صدای سوت سینه",

"خستگی",

"احساس فشار قفسه سینه"

],

"diagnosis\_methods": [

"اسپیرومتری",

"تست عملکرد ریه",

"آزمایش آلرژی",

"تست آلرژی",

"FeNO",

"آزمایش تحریک برونش",

"Peak flow"

],

"treatments": [

"استنشاقی کورتیکواستروئید",

"برونکودیلاتورها",

"پرهیز از محرک‌ها",

"برونکودیلاتورهای سریع‌الاثر",

"کورتیکواستروئیدهای استنشاقی",

"LABA",

"آنتی‌لوکوترین‌ها"

],

"complications": [

"حملات شدید آسم",

"نارسایی تنفسی",

"حمله آسم شدید",

"پنوموتوراکس",

"عفونت‌های تنفسی",

"محدودیت فعالیت"

],

"prevention": [

"پرهیز از آلرژن‌ها",

"کنترل محیط",

"درمان منظم",

"اجتناب از محرک‌ها",

"کنترل آلرژن‌ها",

"واکسیناسیون",

"دارو منظم"

],

"prevalence": "بیش از 262 میلیون نفر مبتلا (2021)"

},

و تعداد کل بیماری ها و درمان ها به ۳۱ بیماری و درمان می باشد.

## 2- تولید داده متنوع

برای این کار یک سری کد توسعه داده شده بود که به دلیل نداشتن اعتبار کافی برای استفاده از api به جای آن به صورت مستقیم با چت بات برخط صحبت کردم اما کد هایی که برای این منظور توسعه پیدا کرده اند هم در پوشه generate-diverse-data همراه پروژه ارسال می شود.

اما حالا می خواهم به صورت دستی فرآیند را اجرا کنم یعنی باید با مدل زبانی به صورت دستی و با از UI ایی که خودشون ارائه دادن ارتباط بگیرم و تمام فرآیند هایی که در کد می باشد را به صورت دستی انجام دهم به همین خاطر مراحل را به صورت توضیح منحصر به فرد خودشون در ادامه آغاز می کنم:

### ۲-۱ تولید مکالمات مختلف برای هر بیماری

برای این منظور یک کد توسعه داده ام که به این صورت عمل می کند که به صورت رندوم از بین بیماری ها که در سرفصل قبل از مدل ها استخراج کرده ام یکی را انتخاب می کند و از بین نوع های مکالمه که به صورت زیر است:

* patient\_initial\_complaint (شکایت اولیه بیمار)
* symptom\_inquiry (پرسش درباره علائم)
* diagnosis\_explanation (توضیح تشخیص)
* treatment\_discussion (بحث درباره درمان)
* follow\_up\_questions (سوالات پیگیری)
* prevention\_advice (مشاوره پیشگیری)
* emergency\_situations (شرایط اورژانسی)

یکی را استخراج می کند و در کنار آن یکی از شخصیت های زیر را برای بیمار به صورت رندوم استخراج می کند:

* بیمار جوان نگران
* والدین کودک بیمار
* سالمند با سوالات متعدد
* بیمار با سطح تحصیلات بالا
* بیمار با دانش محدود پزشکی

و سپس این دو سه دیتای استخراج شده به صورت رندوم را در قالب زیر قرار می دهد:

"

بر اساس اطلاعات بیماری زیر:

{اطلاعات بیماری}

یک مکالمه طبیعی و واقعی بین پزشک و {شخصیت کاربر} تولید کن.

نوع مکالمه: {نوع مکالمه}

قوانین مهم:

1. مکالمه باید کاملاً طبیعی باشد

2. پزشک باید حرفه‌ای و دلسوزانه پاسخ دهد

3. اطلاعات پزشکی دقیق باشد

4. زبان ساده و قابل فهم استفاده شود

5. سوالات متنوع و واقعی باشد

سناریوهای پیچیده‌ای تولید کن که شامل:

1. بیماری با علائم مشترک

2. عوارض چندگانه

3. بیماری‌های همزمان

4. تشخیص افتراقی

نکته بسیار مهم این است که بیمار به عنوان کاربر هست و پزشک باید نیت (intend

هر سناریو شامل مکالمه کامل بین پزشک و بیمار باشد.

فرمت خروجی:

بیمار: [سوال/شکایت]

پزشک: [پاسخ تخصصی]

بیمار: [سوال پیگیری]

پزشک: [پاسخ]

...

"

حال این دیتا را که به من داد کد آن را به صورت دستی برای چت بات می فرستم برای مثال یک نمونه از دیتایی که تولید کرده است به صورت زیر است که نمونه ۱۰۰ تایی پیام اش در فایل medical\_conversation\_prompts.txt ذخیره شده است که می توان تعداد اش را تغیر داد :

"

بر اساس انتخاب یکی از اطلاعات بیماری که در فایل json ارسالی وجود دارند.

و به صورت رندوم انتخاب یکی از شخصیت های: یکی از شخصیت های زیر را برای بیمار به صورت رندوم استخراج می کند:

بیمار جوان نگران

والدین کودک بیمار

سالمند با سوالات متعدد

بیمار با سطح تحصیلات بالا

بیمار با دانش محدود پزشکی

و یکی از انواع مکالمه:

patient\_initial\_complaint (شکایت اولیه بیمار)

symptom\_inquiry (پرسش درباره علائم)

diagnosis\_explanation (توضیح تشخیص)

treatment\_discussion (بحث درباره درمان)

follow\_up\_questions (سوالات پیگیری)

prevention\_advice (مشاوره پیشگیری)

emergency\_situations (شرایط اورژانسی)

یک مکالمه طبیعی و واقعی بین پزشک و بیمار جوان نگران تولید کن.

نوع مکالمه: شرایط اورژانسی

قوانین مهم:

1. مکالمه باید کاملاً طبیعی باشد

2. پزشک باید حرفه‌ای و دلسوزانه پاسخ دهد

3. اطلاعات پزشکی دقیق باشد

4. زبان ساده و قابل فهم استفاده شود

5. سوالات متنوع و واقعی باشد

فرمت خروجی:

{

"Utterance": [

{ "conversation":[

{

"patient": "سلام دکتر... کمک... نمی‌دونم چی شده... یهو نصف صورتم بی‌حس شد و دست چپم رو نمی‌تونم تکون بدم... حرف زدن هم برام سخت شده.",

"intend": ["بی‌حسی در ناحیه‌ی صورت (نصف صورت)","ناتوانی در حرکت دادن دست چپ","اختلال در تکلم"],

"physician": " آروم باشین. من دکتر هستم و اینجا کمکتون می‌کنیم. می‌فهمم که ترسیدین. لطفاً به من نگاه کنید. دقیقاً از کی این علائم شروع شد؟ حتی چند دقیقه هم مهمه.",

},

{

"patient": "شاید... شاید نیم ساعت پیش... داشتم تلویزیون می‌دیدم که یهو سرم گیج رفت و اینجوری شدم. دکتر یعنی من سکته کردم؟ من که سنی ندارم! خیلی می‌ترسم.",

"intend": ["سرگیجه‌ی ناگهانی"],

"physician": "علائمی که میگین خیلی جدیه و ما داریم احتمال سکته مغزی رو بررسی می‌کنیم. سکته مغزی در سنین جوانی هم ممکنه اتفاق بیفته. مهم‌ترین چیز اینه که شما خیلی سریع خودتون رو رسوندین اینجا. ما الان باید فوراً یک سی‌تی اسکن از مغز شما بگیریم تا ببینیم علت این علائم، گرفتگی رگ هست یا خونریزی. این اولین و مهم‌ترین قدمه.",

"detection": ["سکته مغزی", "گرفتگی رگ"],

"prescription": ["گرفتن تصویر CT-SCAN"]

},

{

"patient": "سی‌تی اسکن؟ بعدش چی می‌شه؟ یعنی خوب میشم؟ دستم دوباره حرکت می‌کنه؟",

"intend": ["ابراز نگرانی از عواقب بیماری", " پرسش در رابطه با عوارض"]

"physician": "ببینید، اگر سی‌تی اسکن نشون بده که یک لخته خون باعث این علائم شده، ما یک داروی خیلی قوی داریم که بهش میگن "داروی حل‌کننده لخته" یا tPA. این دارو می‌تونه لخته رو حل کنه و جریان خون به مغز رو برگردونه. هرچقدر زودتر این دارو تزریق بشه، شانس بهبودی و برگشت حرکت دست شما بیشتره. به خاطر همینه که داریم انقدر سریع کار می‌کنیم."

"detection": ["گمانه زنی راجب نتیجه آزمایش"]

"prescription": ["داروی حل‌کننده لخته یا tPA"]

},

{

"patient": "اگه اون دارو کار نکنه چی؟ من برای همیشه فلج میشم؟ نمی‌خوام اینجوری زندگی کنم...",

"intend": ["ابراز نگرانی از عواقب بیماری", " پرسش در رابطه با عوارض"]

"physician": "من نگرانی شما رو کاملاً درک می‌کنم. این دارو در ساعات اولیه بسیار موثره. اما فرآیند درمان به همین‌جا ختم نمیشه. ما یک تیم کامل توان‌بخشی داریم، شامل فیزیوتراپی، گفتاردرمانی و کاردرمانی که بعد از مرحله حاد، به شما کمک می‌کنن تا حرکت و قدرتتون رو تا جای ممکن دوباره به دست بیارین. خیلی از بیماران جوان بهبودی چشمگیری دارن."

},

{

"patient": "علت این اتفاق چیه؟ من همیشه سعی کردم سالم باشم.",

"intend": [" پرسش در رابطه با علت بیماری"]

"physician": "در حال حاضر تمام تمرکز ما روی درمان فوری شماست. بعد از اینکه وضعیتتون پایدار شد، با آزمایش خون و بررسی‌های دقیق‌تر مثل سونوگرافی عروق گردن، دنبال علت اصلی می‌گردیم تا از تکرار اون جلوگیری کنیم. عواملی مثل فشار خون بالا، کلسترول یا برخی مشکلات قلبی می‌تونن زمینه‌ساز باشن. الان تیم ما آماده‌ست که شما رو برای سی‌تی اسکن ببره. "

"detection": ["گمانه زنی در رابطه با علل بیماری"]

},

]

},

]

}

}

ده مکالمه در این قالب به من در پاسخ و صرفا شامل json نهایی بده

خرووجی نمونه ای که gemini pro 2.5 به من داد به این prompt:

{

"conversation": [

{

"patient": "سلام دکتر، خیلی ترسیدم... یهو قفسه سینم درد گرفت و نمی‌تونم درست نفس بکشم. حس می‌کنم یه چیزی داره فشار می‌ده.",

"intend": ["درد قفسه سینه", "تنگی نفس"],

"physician": "آروم باشید، ما اینجاییم که بهتون کمک کنیم. دقیقاً از کی این درد شروع شد؟ شدت درد چطوره، مثل فشار سنگین یا تیز و سوزناکه؟"

},

{

"patient": "فکر کنم یه ربع پیش شروع شد... خیلی سنگین و فشاردهنده‌ست، انگار یه نفر رو قفسه سینم نشسته. دست چپم هم یه جورایی گزگز می‌کنه.",

"intend": ["درد قفسه سینه مداوم", "گزگز دست چپ"],

"physician": "علائمی که می‌گین جدیه و ممکنه به قلب مربوط باشه. ما الان یه نوار قلب (ECG) ازتون می‌گیریم و آزمایش خون برای بررسی آنزیم‌های قلبی. تا وقتی نتیجه بیاد، بهتون اکسیژن و داروهای اولیه می‌دیم. نگران نباشید، ما سریع عمل می‌کنیم.",

"detection": ["مشکوک به مشکل قلبی"],

"prescription": ["نوار قلب (ECG)", "آزمایش خون", "اکسیژن"]

},

{

"patient": "یعنی حمله قلبیه؟ من فقط ۳۰ سالمه، چطور ممکنه؟",

"intend": ["ابراز نگرانی از تشخیص", "پرسش درباره علت"],

"physician": "حمله قلبی تو سنین پایین هم ممکنه، ولی هنوز مطمئن نیستیم. عواملی مثل استرس، سابقه خانوادگی یا حتی کلسترول بالا می‌تونه نقش داشته باشه. نوار قلب و آزمایش‌ها به ما کمک می‌کنه دقیق‌تر تشخیص بدیم. الان تمرکزمون روی پایدار کردن وضعیت شماست.",

"detection": ["احتمال حمله قلبی"]

},

{

"patient": "اگه حمله قلبی باشه، خوب می‌شم؟ نمی‌خوام اینجوری بمونه...",

"intend": ["ابراز نگرانی از عوارض"],

"physician": "اگه حمله قلبی باشه و زود درمان بشه، شانس بهبودی خیلی بالاست، به‌خصوص چون شما جوونید. ما ممکنه داروهایی مثل آسپرین یا نیتروگلیسرین بدیم و اگه لازم باشه، اقدامات پیشرفته‌تری مثل آنژیوگرافی انجام می‌دیم.",

"prescription": ["آسپرین", "نیتروگلیسرین", "آنژیوگرافی احتمالی"]

},

{

"patient": "بعدش باید چی کار کنم که دیگه اینجوری نشم؟",

"intend": ["پرسش درباره پیشگیری"],

"physician": "بعد از اینکه وضعیتتون پایدار شد، آزمایش‌های تکمیلی انجام می‌دیم تا دلیل اصلی رو پیدا کنیم. احتمالاً باید تغییراتی تو سبک زندگی بدین، مثل رژیم غذایی سالم، ورزش منظم و کنترل استرس. اما الان فقط به ما اعتماد کنید، ما داریم همه‌چیز رو مدیریت می‌کنیم."

}

]

}

"

برای این که مکالمات لحن های مختلف هم داشته باشد این لحن ها را هم بهش اضافه کردم:  
conversation\_styles = [

"formal\_medical", # رسمی پزشکی

"friendly\_doctor", # دوستانه

"emergency\_urgent", # اورژانسی

"educational", # آموزشی

"reassuring" # دلگرم‌کننده

]

در فایل regenerate\_diff\_style.py این امکان رو فراهم کردم که بعد از این که مکالمات را از چت بات گرفتید به عنوان ورودی base\_conversation بهش بدهیم و خروجی را در مدل های مختلف برای ما ایجاد کند. که نمونه خروجی به ازای لحن آموزشی به صورت زیر می شود:

"

**بیمار:** (با کلام منقطع و اضطراب مشهود) جناب دکتر، من دچار یک وضعیت ناگهانی شدم. سمت چپ صورتم پارستزی (بی‌حسی) دارد، در بازوی چپم ضعف شدید (همی‌پارزی) احساس می‌کنم و در تکلم (دیس‌آرتری) دچار مشکل شده‌ام.

**پزشک:** (با لحنی حرفه‌ای و متمرکز) آرامش خود را حفظ کنید. من پزشک مسئول شما هستم. لطفاً شرح دهید این علائم دقیقاً چه زمانی شروع شد؟ زمان شروع (Time of Onset) برای پروتکل درمانی ما بسیار حیاتی است.

**بیمار:** حدود ۳۰ دقیقه قبل. به صورت کاملاً حاد و ناگهانی دچار سرگیجه و سپس این نقایص عصبی شدم. آیا این تظاهرات بالینی می‌تواند یک حادثه عروقی مغز (CVA) باشد؟ با توجه به سن من، این امکان وجود دارد؟

**پزشک:** تظاهرات بالینی شما مطرح‌کننده یک نقص عصبی کانونی حاد (Acute Focal Neurological Deficit) است که تشخیص افتراقی اصلی آن، سکته مغزی است. CVA در جمعیت جوان نیز رخ می‌دهد، هرچند شیوع کمتری دارد. پروتکل اورژانسی ما ایجاب می‌کند که فوراً یک CT Scan مغزی بدون کنتراست برای شما انجام شود تا سکته هموراژیک (خونریزی‌دهنده) از سکته ایسکمیک (ناشی از انسداد عروق) افتراق داده شود.

**بیمار:** پس از انجام CT Scan، برنامه درمانی چیست؟ پیش‌آگهی بهبودی و بازگشت عملکرد اندام فوقانی چگونه است؟

**پزشک:** در صورتی که در تصویربرداری شواهدی از خونریزی وجود نداشته باشد و تشخیص سکته ایسکمیک تأیید شود، شما کاندید دریافت درمان ترومبولیتیک با استفاده از فعال‌کننده پلاسمینوژن بافتی (tPA) خواهید بود. این دارو با حل کردن ترومبوز (لخته)، پرفیوژن مجدد بافت مغز را امکان‌پذیر می‌سازد. زمان در این درمان نقش کلیدی دارد و تجویز سریع دارو، شانس بهبودی عملکرد عصبی را به شکل معناداری افزایش می‌دهد.

**بیمار:** و در صورت عدم پاسخ به درمان ترومبولیتیک، چه گزینه‌های دیگری وجود دارد؟ آیا احتمال نقص عصبی دائمی (فلج) وجود دارد؟

**پزشک:** پروتکل‌های درمانی مشخصی برای این موارد وجود دارد. درمان با tPA در بازه زمانی مشخص، نرخ موفقیت بالایی دارد. فارغ از پاسخ اولیه به درمان فاز حاد، فرآیند توانبخشی پس از پایدار شدن شرایط بالینی آغاز می‌شود. یک تیم چند رشته‌ای شامل متخصصین فیزیوتراپی، گفتاردرمانی و کاردرمانی برای به حداکثر رساندن بازیابی عملکرد عصبی شما، برنامه درمانی جامعی را تدوین خواهند کرد.

**بیمار:** اتیولوژی (علت) این عارضه در گروه سنی من چه می‌تواند باشد؟ من فاکتور خطر مشخصی نداشته‌ام.

**پزشک:** اولویت اول در فاز حاد، مدیریت و درمان اورژانسی است. پس از تثبیت وضعیت، ارزیابی‌های کاملی جهت یافتن اتیولوژی زمینه‌ای انجام خواهد شد. این بررسی‌ها شامل آزمایشات کامل خون، اکوکاردیوگرافی، و سونوگرافی داپلر عروق کاروتید خواهد بود تا علل احتمالی مانند اختلالات انعقادی، پاتولوژی‌های قلبی یا آترواسکلروز زودرس بررسی شوند.

**بیمار:** متوجه شدم. لطفاً اقدامات لازم را انجام دهید.

**پزشک:** تیم پزشکی تمام اقدامات لازم را مطابق با آخرین گایدلاین‌ها انجام خواهد داد. شما در بازه زمانی طلایی (Golden Window) برای درمان مراجعه کرده‌اید که این امر بسیار در پیش‌آگهی شما مؤثر است. اکنون شما را جهت انجام CT Scan به بخش رادیولوژی منتقل می‌کنیم.

"

نمونه ایی از چت باتی که باهاش این دیتا را تولید کرده ایم:

https://grok.com/share/c2hhcmQtMg%3D%3D\_4eba5e23-8172-497e-86df-a0810db03c7b

# روش دوم جمع آوری داده به فرمت query، intend

در این روش از LLM می خواهیم برای ما مکالماتی را ایجاد کند و تشخیص دهد که این مکالمه برای چه نیتی است. برای این کار از لیست بیماری هایی که قبلا استخراج کرده ایم استفاده می کنیم که در merged\_diseases.json قرار دارد و سپس به این صورت عمل می کنیم که

"

خودت را به عنوان یک متخصص در تولید دیتابیس جهت parameter fine tuning یک مدل مثل llama که به صورت آفلاین است و با lora قرار است که fine tune شود در نظر بگیر.

قرار است دیتابیس ایی تولید کنیم که برای fine tune کردن مدل به عنوان یک مدل multi intend detector استفاده بشود برای این منظور من یک لیست از اطلاعات بیماری ها که شامل تمام اطلاعات و درمان و دارو هست می باشد ولی خودت را به این محدود نکن و می توانی از اطلاعاتی که خودت داری و بیماری هایی که مدنظر خودت هست هم استفاده کنی.

یک درخواست از طرف کاربر باید ایجاد کنی که انگار یک prompt در یک مکالمه طبیعی بین پرسشگر در رابطه با بیماری و تمام چیز های مرتبط با آن مثل علائم و یا درمان یا دارو یا … باشد و نقش شما به عنوان کسی که از پرسش می شود می تواند به عنوان دکتر یا متصدی داروخانه یا نگهبان یک بیمارستان یا یک آشنا که اطلاعات بیشتری از فرد راجب آن بیماری و موارد مرتبط با آن دارد باشد.

لحن های پرسش می تواند متفاوت باشد برای مثال:  
conversation\_styles = [

"formal\_medical", # رسمی پزشکی

"friendly\_doctor", # دوستانه

"emergency\_urgent", # اورژانسی

"educational", # آموزشی

"reassuring" # دلگرم‌کننده

]

اما خودت را به این لحن ها محدود نکن و می توانی هر نوع لحنی که به نظرت مناسب بیاید را استفاده کنی و انگار بخشی از یک مکالمه طبیعی هست صحبتی که فرد می کند و این سوالات و صحبت هایی که فرد می کند باید متنوع و واقعی باشد.

سناریو های پیچیده تولید کن:

1.بیماری با علائم مشترک

2. عوارض چندگانه

3. بیماری‌های همزمان

4. تشخیص افتراقی

نکته مهم: به خصوص توجه داشته باشد که باید در بیشتر صحبت هایی که رخ می دهد چند نیت (intend) در صحبت فرد نهفته باشد برای مثال:

"می خوام فردا بلیط هواپیما بگیرم و ببینم هوا فردا چطوره" -> ["خرید بلیط"، "چک کردن وضعیت هوا"]

"من قفسه سینه ام درد می کند و نیاز دارم برام نسخه پزشکی بنویسید" -> ["درخواست\_نسخه\_الکترونیکی", "پرسش\_درباره\_تشخیص\_بیماری"]

فرمت خروجی باید صرفا یک json فایل باشد به این صورت که

}

"Utterance":[

{“query”:”می خوام فردا بلیط هواپیما بگیرم و ببینم هوا فردا چطوره", “intend”: ["خرید بلیط"، "چک کردن وضعیت هوا"]},

{“query”:”من قفسه سینه ام درد می کند و نیاز دارم برام نسخه پزشکی بنویسید", “intend”: ["درخواست\_نسخه\_الکترونیکی", "پرسش\_درباره\_تشخیص\_بیماری"]}

]

{

این دیکشنری باید ۱۰۰۰ تا نمونه داشته باشد و صرفا محتوای این فایل json را به من برگردان

"

و cloud sonnet thinking 3.7 این سوال من را با این پرامپ بهبود داد:

من می خواهم برای ایجاد دیتابیس از LLM ها استفاده کنم و نمونه prompt ایی که برای آن نوشته ام در پایین است می توانی آن را بهتر کنی:

>متن پرامپتی که خودم نوشتم <

که خروجی آن به صورت زیر شد:

"

هدف

شما به عنوان متخصص تولید دیتاست برای fine-tuning مدل‌های زبانی، وظیفه دارید یک دیتاست غنی از پرسش‌های چند-نیتی در حوزه پزشکی و سلامت تولید کنید. این دیتاست برای آموزش یک مدل زبانی آفلاین (مانند Llama) با استفاده از روش LoRA جهت تشخیص همزمان چندین نیت کاربر استفاده خواهد شد.

مشخصات دیتاست

حجم: ۱۰۰۰ نمونه متنوع

ساختار: هر نمونه شامل یک پرسش (query) و فهرست نیت‌های (intends) مرتبط

ویژگی اصلی: هر پرسش حاوی حداقل ۲ نیت همزمان باشد

حوزه موضوعی: پزشکی، سلامت، بیماری‌ها، درمان، دارو و موارد مرتبط

تنوع: شامل سناریوهای مختلف، لحن‌های متفاوت، و نقش‌های گوناگون در گفتگو

راهنمای تولید محتوا

۱. تنوع نیت‌ها

نیت‌های پزشکی متداول (این لیست جامع نیست و می‌توانید نیت‌های بیشتری اضافه کنید):

"درخواست\_نسخه\_الکترونیکی",

"پرسش\_درباره\_تشخیص\_بیماری",

"پرسش\_درباره\_عوارض\_دارو",

"درخواست\_نوبت\_پزشکی",

"پرسش\_درباره\_هزینه\_درمان",

"درخواست\_اطلاعات\_بیماری",

"درخواست\_مشاوره\_پزشکی\_فوری",

"پیگیری\_نتایج\_آزمایش",

"درخواست\_معرفی\_پزشک\_متخصص",

"پرسش\_درباره\_روش‌های\_جایگزین\_درمان",

"استعلام\_بیمه\_درمانی",

"رزرو\_نوبت",

"لغو\_نوبت",

"تغییر\_نوبت",

"پرسش\_درباره\_علائم",

"پرسش\_درباره\_تشخیص\_بیماری",

"درخواست\_تفسیر\_نتایج\_آزمایش",

"پرسش\_درباره\_دارو",

"پرسش\_درباره\_عوارض\_دارو",

"پرسش\_درباره\_تداخل\_دارویی",

"یادآوری\_مصرف\_دارو",

"درخواست\_تجویز\_دارو",

"درخواست\_نسخه\_الکترونیکی",

"پرسش\_درباره\_واکسیناسیون",

"درخواست\_توصیه\_تغذیه‌ای",

"درخواست\_برنامه\_غذایی",

"پرسش\_درباره\_ورزش\_و\_فعالیت\_بدنی",

"درخواست\_مشاوره\_پزشکی\_آنلاین",

"پیدا\_کردن\_پزشک\_متخصص",

"پیدا\_کردن\_مرکز\_درمانی\_نزدیک",

"درخواست\_تماس\_با\_اورژانس",

"راهنمایی\_در\_موقعیت\_اورژانسی",

"پرسش\_درباره\_عمل\_جراحی",

"پرسش\_درباره\_پروسیجر\_پزشکی",

"پیگیری\_وضعیت\_درمان",

"پیگیری\_پرونده\_پزشکی",

"پرسش\_درباره\_بیماری\_های\_خاص",

"پرسش\_درباره\_دیابت",

"پرسش\_درباره\_فشار\_خون",

"پرسش\_درباره\_سرطان",

"پرسش\_درباره\_بارداری",

"پرسش\_درباره\_سلامت\_کودکان",

"پرسش\_درباره\_سلامت\_زنان",

"پرسش\_درباره\_سلامت\_قلب",

"پرسش\_درباره\_سلامت\_ریه",

"پرسش\_درباره\_سلامت\_کلیه",

"درخواست\_تعیین\_وقت\_برای\_تصویربرداری",

"راهنمایی\_برای\_آزمایش\_تشخیصی",

"پرسش\_درباره\_پیشگیری\_از\_بیماری",

"درخواست\_دریافت\_نتیجه\_آزمایش",

"پرسش\_درباره\_مراقبت\_های\_پس\_از\_عمل",

"دریافت\_اطلاعات\_درباره\_بیمه\_درمانی",

"پرسش\_درباره\_هزینه\_درمان",

"درخواست\_گزارش\_عوارض\_دارو",

"پیشنهاد\_پزشک\_یا\_کلینیک",

"درخواست\_اطلاعات\_درباره\_واکسیناسیون\_کودک",

"درخواست\_اطلاعات\_درباره\_غربالگری",

"پرسش\_درباره\_سلامت\_روان",

"درخواست\_مشاوره\_روانشناسی",

"گزارش\_عوارض\_یا\_علائم\_جدید",

"درخواست\_تمدید\_نسخه",

"درخواست\_تغییر\_مقدار\_یا\_نوع\_دارو",

"پرسش\_درباره\_مراقبت\_های\_پیش\_از\_عمل",

"پرسش\_درباره\_تجهیزات\_پزشکی",

"درخواست\_آموزش\_سلامتی",

"پرسش\_درباره\_رژیم\_غذایی\_ویژه",

"پرسش\_درباره\_آب\_و\_هوا\_برای\_سلامت",

"درخواست\_برنامه\_ورزشی\_برای\_بیماری\_خاص",

"درخواست\_پیگیری\_بعد\_از\_ترخیص",

"پرسش\_درباره\_تست\_غربالگری\_ژنتیک",

"پرسش\_درباره\_ناباروری",

"پرسش\_درباره\_سلامت\_سالمندان",

"پرسش\_درباره\_واکسیناسیون\_سالمندان",

"گزارش\_فوریت\_پزشکی",

"درخواست\_هماهنگی\_برای\_ویزیت\_خانگی",

"درخواست\_بررسی\_وضعیت\_بیمار\_در\_بیمارستان",

"پرسش\_درباره\_دوره\_نقاهت",

"درخواست\_تغییر\_پزشک\_معالج",

"پرسش\_درباره\_عوارض\_عمل\_جراحی",

"درخواست\_خدمات\_پرستاری\_در\_منزل",

"پرسش\_درباره\_تست\_کرونا",

"درخواست\_نتیجه\_تست\_کرونا",

"پرسش\_درباره\_آزمایش\_خون",

"پرسش\_درباره\_آزمایش\_ادرار",

"پرسش\_درباره\_آزمایش\_قند\_خون",

"درخواست\_پیگیری\_برنامه\_درمانی"

]

۲. سناریوهای پیچیده (تمرکز ویژه)

بیماری‌های با علائم مشترک: سناریوهایی که در آنها علائم می‌تواند نشان‌دهنده چندین بیماری باشد

عوارض چندگانه: پرسش‌هایی درباره عوارض متعدد یک بیماری یا دارو

بیماری‌های همزمان: سناریوهای افراد مبتلا به چند بیماری همزمان

تشخیص افتراقی: پرسش‌هایی که نیاز به تفکیک بین چند بیماری مشابه دارند

شرایط اورژانسی و غیر اورژانسی: ترکیب سؤالات فوری و غیرفوری در یک پرسش

۳. لحن‌های گفتگو

متنوع سازی لحن گفتگوها بر اساس:

رسمی پزشکی (با اصطلاحات تخصصی)

دوستانه و صمیمی

اضطراری و فوری

آموزشی و توضیحی

دلگرم‌کننده و حمایتی

نگران و مضطرب

سردرگم و گیج

تخصصی (مانند گفتگوی بین دو پزشک)

عامیانه (با استفاده از اصطلاحات غیرتخصصی)

۴. نقش‌های مخاطب

پرسش‌ها می‌تواند خطاب به افراد مختلف باشد:

پزشک متخصص

پزشک عمومی

پرستار

داروساز

متصدی پذیرش بیمارستان

مشاور سلامت

اپراتور خدمات پزشکی تلفنی

مشاور بیمه درمانی

یک آشنا با اطلاعات پزشکی

۵. نکات مهم

هر پرسش حتماً شامل حداقل ۲ نیت متفاوت باشد

از ترکیب هوشمندانه نیت‌ها به شکل طبیعی استفاده کنید

پرسش‌ها باید واقع‌گرایانه و شبیه گفتگوهای روزمره باشند

از اطلاعات واقعی پزشکی استفاده کنید اما محدود به دانش خود نباشید

تنوع در طول پرسش‌ها (از جملات کوتاه تا توضیحات مفصل)

تنوع در سطح دانش پرسشگر (از افراد عادی تا متخصصان پزشکی)

فرمت خروجی

خروجی باید دقیقاً به صورت یک فایل JSON با ساختار زیر باشد:

{

"Utterance": [

{

"query": "سلام دکتر، من دو روزه سردرد شدید و تب دارم و می‌خواستم ببینم امروز بعدازظهر وقت ویزیت دارید؟",

"intend": ["پرسش\_درباره\_تشخیص\_بیماری", "درخواست\_نوبت\_پزشکی"]

},

{

"query": "داروهای فشار خونم تموم شده، میشه یه نسخه الکترونیکی بنویسید؟ راستی این سرگیجه‌هایی که جدیداً دارم ممکنه مربوط به همین داروها باشه؟",

"intend": ["درخواست\_نسخه\_الکترونیکی", "پرسش\_درباره\_عوارض\_دارو"]

}

]

}

Don’t consider any limitation for the number of word and token in your response and give me as much as necessary length of response

برای این منظور من یک لیست از اطلاعات بیماری ها که شامل تمام اطلاعات و درمان و دارو هست می باشد ولی خودت را به این محدود نکن و می توانی از اطلاعاتی که خودت داری و بیماری هایی که مدنظر خودت هست هم استفاده کنی که این اطلاعات در فایل merged\_diseases.json قرار دارد

"

خروجی و نتایج در فایل intend\_database.json قرار گرفته شده است

چند باری با پرامپت های بالا استفاده کردم ولی نتایجی که به من ارائه داد از دیتای داخل فایل merged\_diseases استفاده نکرد پس پرامپت را مقداری تغیر دادم تا حتما از دیتای داخل اون فایل استفاده کند صرفا.

به همین خاطر در ادامه پرامپ های قبل این پرامپت را به مدل زدم:  
"

I want you to consider question that include data from json file that I would send to you, and I want you provide diverse question including distinct options from different disease and symptom in your question and you could mention diverse drugs for treatment in your qustions, for example for the implication of them or the efficiency of them or how they could use them, like this example:" در چه ساعاتی از روز باید داروی کاربامازپین را مصرف کنم و اثرات مضر آن چیست"

"آیا تهیه کدئین نیاز به دستور پزشک دارد و تاثیر کدئین برای کدام بیماری ها موثر است"

I want enrich my dataset with information about drugs inorder to my local model could better understand drugs in normal communication with its users

"

اما همچنان پاسخ هایی که به من میداد جالب نبود پس رفتم سراغ این که لیستی از دارو ها برای هر کداام از بیماری ها تهیه کنم به همین خاطر به مدل grok 3 گفتم که:

"

In the attachment, I have sent to you 31 diseases comperhansive information including symptoms, diagnosis\_methods, treatments, complications, and prevention, so I want you provide me diverse medicine and methods for treatment of each one of these in the following json format:

{“disease\_name”:{“medicine”:[drugs\_name], “methods”:[methods\_of\_treatment]}}

Provide me as much as possible infromation for the medicine, and methods

“

خروجی این کار در فایل treatment\_and\_medicine.json ذخیره شده است که در پوشه diseases-json-file قرار دارد.

حال پرامپت قبل را اینبار به grok3 می دهم به جای gemini pro 2.5 و سعی می کنم تعدادی سطر دیتابیس هم از طریق آن ایجاد کنم. نمونه دیتایی که اضافه کرد:

"query": "همکار پزشک، در مالاریا و هپاتیت همزمان، آرتیمیسینین رو با داروهای ضدویروس ترکیب کنیم و پیشگیری با پشه‌بند چیه؟",

"intend": ["پرسش درباره مدیریت بیماری‌های همزمان", "پرسش درباره پیشگیری از بیماری"]

},

نمونه ایی از چت باتی که باهاش این دیتا را تولید کرده ایم:

https://grok.com/share/c2hhcmQtMg%3D%3D\_b04d4d75-63a0-47ec-ad0c-069d3e193c81

# روش سوم جمع آوری داده به فرمت query, diseases

در این جا بر مبنای اختلال و بیماری این کار را می کنیم یعنی کاربر توضیحاتی در رابطه با بیماری خود می دهد و ما بر اساس این توضیحات اون میایم و نتیجه گیری می کنیم که اختلالات اون چیست به این صورت که ابتدا لیست بیماری هامون رو گسترش می دهیم قبل از این در فایل merged\_diseases.json حدود ۳۱ سری بیماری داشتیم حال از هوش مصنوعی grok 4 می خواهیم که این بیماری ها را در نظر بگیرد و تعداد این بیماری ها را به ۱۵۰ تا برساند و سپس بر اساس این بیماری ها می رویم سراغ مدل های زبانی دیگر و از آن ها درخواست می کنیم که بر اساس ویژگی هایی که هر بیماری دارد برای ما یک سری query تولید کند که شامل ویژگی های چندین بیماری باشد و اون باید نوع بیماری یا اختلال را به عنوان intend به ما برگرداند.

## گسترش تعداد بیماری ها

برای این منظور prompt زیر را به مدل زبانی می زنیم:

"

من یک لیست json از بیماری های شایع در سطح جهان را برایت ارسال کرده ام در قسمت attachment و می خواهم این لیست را گسترش بدم که شامل تعداد بیشتری از بیماری ها باشد پس برایم تعداد ۳۰ بیماری به این لیست بیماری در فرمتی که در فایل json هست بهم بده تا به این لیست بیماری ها اضافه کنم

برای هر بیماری اطلاعات زیر را به صورت دقیق و علمی ارائه دهید:

- نام بیماری (به فارسی و انگلیسی)

- دسته‌بندی بیماری (عفونی/غیرعفونی، حاد/مزمن)

- علائم اصلی (حداقل 3-5 علامت مهم)

- روش‌های تشخیص (آزمایش‌های تشخیصی، معاینات بالینی)

- درمان‌های رایج (دارویی، غیردارویی، جراحی در صورت نیاز)

- عوارض احتمالی (کوتاه‌مدت و بلندمدت)

- راه‌های پیشگیری (در صورت وجود)

- میزان شیوع جهانی (درصد یا آمار کلی)

پاسخ را در قالب JSON با ساختار زیر ارائه دهید:

```json

{

"diseases": [

{

"name\_persian": "",

"name\_english": "",

"category": "",

"symptoms": [],

"diagnosis\_methods": [],

“Medicine”: [],

"treatments": [],

"complications": [],

"prevention": [],

"prevalence": ""

}

]

}

لطفاً اطلاعات علمی و به‌روز ارائه دهید و از منابع معتبر پزشکی استفاده کنید.

"

حال در یک فایل json دیگر این اطلاعات را راجب این بیماری ها می خواهم

"

In the attachment, I have sent to you 31 diseases comperhansive information including symptoms, diagnosis\_methods, treatments, complications, and prevention, so I want you provide me diverse medicine and methods for treatment of each one of these in the following json format:

{“disease\_name”:{“medicine”:[drugs\_name], “methods”:[methods\_of\_treatment]}}

Provide me as much as possible infromation for the medicine, and methods

“

خروجی نهایی این کار بعد از ۳ بار تکرار این کار و رساندن دیتابیس به ۱۲۰ بیماری در فایل merged\_diseases\_extended.json ذخیره شده است.

لینک چت ایی که از طریق آن این گسترش اتفاق افتاده است: https://grok.com/share/c2hhcmQtMg%3D%3D\_a83cd6e1-d305-43f9-9804-ccf22c885d5c

## تولید داده های متنوع:

برای این منظور از prompt استفاده کرده ام:

"  
خودت را به عنوان یک متخصص در تولید دیتابیس جهت parameter fine tuning یک مدل مثل llama که به صورت آفلاین است و با lora قرار است که fine tune شود در نظر بگیر.

قرار است دیتابیس ایی تولید کنیم که برای fine tune کردن مدل به عنوان یک مدل multi intend detector استفاده بشود برای این منظور من یک لیست از اطلاعات بیماری ها که شامل تمام اطلاعات و درمان و دارو هست می باشد ولی خودت را به این محدود نکن و می توانی از اطلاعاتی که خودت داری و بیماری هایی که مدنظر خودت هست هم استفاده کنی.

یک درخواست از طرف کاربر باید ایجاد کنی که انگار یک prompt در یک مکالمه طبیعی بین پرسشگر در رابطه با بیماری و تمام چیز های مرتبط با آن مثل علائم و یا درمان یا دارو یا … باشد و نقش شما به عنوان کسی که از پرسش می شود می تواند به عنوان دکتر یا متصدی داروخانه یا نگهبان یک بیمارستان یا یک آشنا که اطلاعات بیشتری از فرد راجب آن بیماری و موارد مرتبط با آن دارد باشد.

لحن های پرسش می تواند متفاوت باشد

اما خودت را به این لحن ها محدود نکن و می توانی هر نوع لحنی که به نظرت مناسب بیاید را استفاده کنی و انگار بخشی از یک مکالمه طبیعی هست صحبتی که فرد می کند و این سوالات و صحبت هایی که فرد می کند باید متنوع و واقعی باشد.

سناریو های پیچیده تولید کن:

1.بیماری با علائم مشترک

2. عوارض چندگانه

3. بیماری‌های همزمان

4. تشخیص افتراقی

نکته مهم: به خصوص توجه داشته باشد که باید در بیشتر صحبت هایی که رخ می دهد چند نیت (intend) که همان بیماری هستند در صحبت فرد نهفته باشد برای مثال:

{“query”:”من درد شدیدی در ناحیه کمرم احساس می کنم و گاهی برای خم شدن و برداشتن اجسام احساس درد میکنم و اغلب اوقات دچار سردرد هستم", “diseases”: ["میگرن"، "تنگی کانال", “دیسک کمر”]},

{“query”:” من قفسه سینه ام درد می کند و هنگام تنفس دچار تنگی نفس می شوم و در معده ام دچار سوزش هستم", “diseases”: ["پرفوراسیون معده", "شکستگی قفسه سینه "]}

]

فرمت خروجی باید صرفا یک json فایل باشد به این صورت که

}

"Utterance":[

{“query”:”من درد شدیدی در ناحیه کمرم احساس می کنم و گاهی برای خم شدن و برداشتن اجسام احساس درد میکنم و اغلب اوقات دچار سردرد هستم", “diseases”: ["میگرن"، "تنگی کانال", “دیسک کمر”]},

{“query”:” من قفسه سینه ام درد می کند و هنگام تنفس دچار تنگی نفس می شوم و در معده ام دچار سوزش هستم", “diseases”: ["پرفوراسیون معده", "شکستگی قفسه سینه "]}

]

{

این دیکشنری باید ۱۰۰ تا نمونه داشته باشد و صرفا محتوای این فایل json را به من برگردان

نکته بسیار مهم برای ۱۲۰ تا بیماری دیتایی در فایل json ایی که در قسمت attachment ارسال شده است و همچنین درمان های هر کدام و بیماری های شان هم در فایل json جداگانه ایی ارسال شده است که می توانی از آن ها برای تولید استفاده کنی.

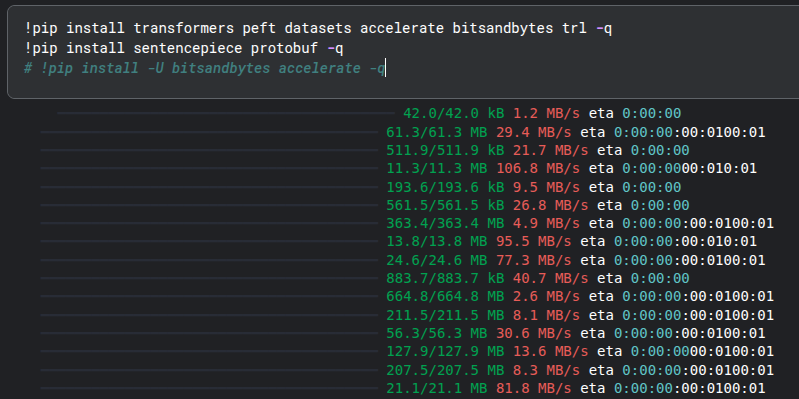
"

این کار در چت با لینک زیر صورت گرفته است:

<https://grok.com/share/c2hhcmQtMg%3D%3D_83c4c523-be8a-4999-87ee-d59191c41809>

# پیاده سازی parameter fine tune با استفاده از lora

### فراخوانی کتابخانه ها



این کد شامل دستورات نصب کتابخانه‌های پایتون با استفاده از ابزار pip است که برای آماده‌سازی محیط پروژه استفاده می‌شه.هدف این کد، نصب کتابخانه‌های مورد نیاز برای کار با مدل‌های یادگیری عمیق (مخصوصاً مدل‌های زبانی مبتنی بر ترانسفورمر) و بهینه‌سازی آن‌ها برای آموزش یا استفاده است. -q باعث می‌شه خروجی نصب کم‌حجم و بدون جزئیات اضافی باشه (حالت بی‌صدا)

transformers: کتابخانه‌ای از Hugging Face که ابزارها و مدل‌های از پیش آموزش‌دیده برای پردازش زبان طبیعی (NLP)، مانند BERT، GPT و غیره رو فراهم می‌کنه. شامل ابزارهایی برای بارگذاری مدل‌ها، توکنایزرها و مدیریت داده‌هاست.

peft: مخفف Parameter-Efficient Fine-Tuning. این کتابخانه برای تنظیم دقیق (fine-tuning) مدل‌های بزرگ با استفاده از روش‌های بهینه مثل LoRA (Low-Rank Adaptation) طراحی شده، که باعث کاهش مصرف حافظه و محاسبات می‌شه.

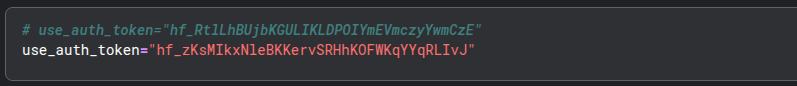
datasets: کتابخانه‌ای از Hugging Face برای بارگذاری، مدیریت و پیش‌پردازش مجموعه‌داده‌های بزرگ به‌صورت کارآمد.

accelerate: ابزار Hugging Face برای ساده‌سازی آموزش مدل‌ها روی چندین GPU، TPU یا CPU. این کتابخانه مدیریت توزیع محاسبات رو آسون می‌کنه.

bitsandbytes: کتابخانه‌ای برای بهینه‌سازی مدل‌های یادگیری عمیق با استفاده از کوانتیزاسیون (مثل 4-bit یا 8-bit). این کار باعث کاهش مصرف حافظه و افزایش سرعت اجرا می‌شه، به‌خصوص برای مدل‌های بزرگ.

trl: مخفف Transformers Reinforcement Learning. برای آموزش مدل‌های زبانی با روش‌های یادگیری تقویتی (مانند RLHF - Reinforcement Learning from Human Feedback) استفاده می‌شه.

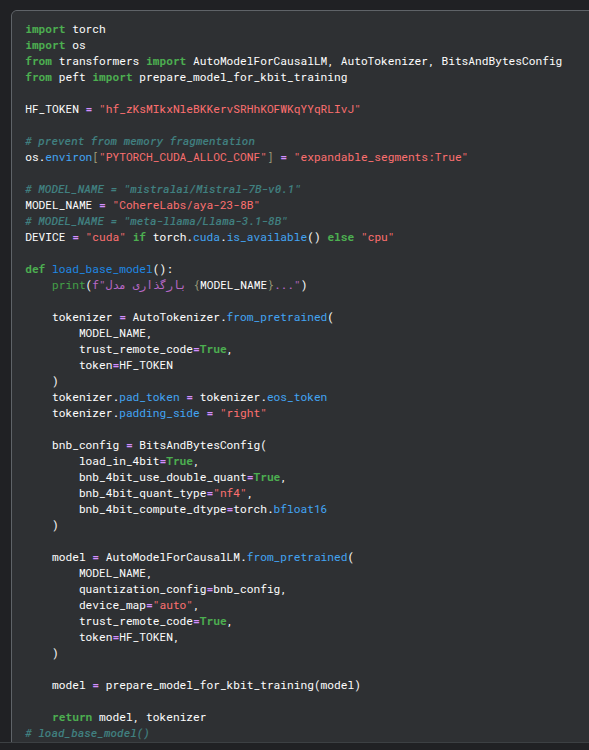
### وارد کردن توکن



برای دسترسی به برخی از مدل های hugging face نیاز است که براشون اجازه نامه پر شود و بعد از تایید آن بتونیم به آن مدل ها دسترسی داشته باشیم که توکن برای authentication اینجا به کار می آید که در ادامه در فراخوانی مدل و توکنایزر از آن استفاده می کنیم.

### انتخاب مدل پایه

این کد پایه‌ای برای بارگذاری یک مدل زبانی بزرگ (aya-23-8B) و آماده‌سازی آن برای استفاده یا تنظیم دقیق است. با استفاده از کوانتیزاسیون 4 بیتی و تنظیمات بهینه‌سازی حافظه، این کد امکان اجرای مدل‌های بزرگ را روی سخت‌افزارهای با حافظه محدود فراهم می‌کند. انتخاب مدل aya-23-8B نشان‌دهنده تمرکز پروژه روی یک مدل چندزبانه با عملکرد بالا است که برای وظایف تولید متن یا پردازش زبان طبیعی مناسب است. آماده‌سازی مدل با peft برای تنظیم دقیق با روش‌های کارآمد (مثل LoRA) است.



این کد شامل بارگذاری مدل و توکنایزر، بهینه‌سازی حافظه با کوانتیزاسیون و آماده‌سازی مدل برای آموزش است.

این کد وظیفه بارگذاری یک مدل زبانی از پیش آموزش‌دیده (در اینجا CohereLabs/aya-23-8B) و توکنایزر مرتبط با آن را بر عهده دارد. همچنین با استفاده از تکنیک‌های کوانتیزاسیون (از طریق کتابخانه bitsandbytes) و آماده‌سازی مدل برای تنظیم دقیق با peft، مصرف حافظه و منابع محاسباتی بهینه‌سازی شده است.

از این سبک trick ها وقتی استفاده می کنیم که نیاز به کار با مدل‌های زبانی بزرگ (مانند تولید متن یا چت‌بات) داریم اما می‌خواهند این کار را با منابع سخت‌افزاری محدود (مثل GPU با حافظه کم) انجام دهیم.

* **AutoModelForCausalLM**: کلاسی از کتابخانه transformers برای بارگذاری مدل‌های زبانی که برای تولید متن (causal language modeling) طراحی شده‌اند.
* **AutoTokenizer**: کلاسی برای بارگذاری توکنایزر متناسب با مدل انتخاب‌شده.
* **BitsAndBytesConfig**: کلاسی از کتابخانه bitsandbytes برای تنظیم کوانتیزاسیون مدل (کاهش دقت برای بهینه‌سازی حافظه).
* **prepare\_model\_for\_kbit\_training**: تابعی از کتابخانه peft برای آماده‌سازی مدل برای تنظیم دقیق با کوانتیزاسیون.

os.environ["PYTORCH\_CUDA\_ALLOC\_CONF"] = "expandable\_segments:True"

* این خط یک متغیر محیطی برای PyTorch تنظیم می‌کند تا از تکه‌تکه شدن حافظه (memory fragmentation) در GPU جلوگیری کند.
* تنظیم expandable\_segments:True به PyTorch اجازه می‌دهد حافظه را به‌صورت پویا و کارآمدتر تخصیص دهد، که برای مدل‌های بزرگ مهم است.

DEVICE = "cuda" if torch.cuda.is\_available() else "cpu"

* **DEVICE**: بررسی می‌کند که آیا GPU (با CUDA) در دسترس است یا خیر. اگر باشد، مدل روی cuda (GPU) اجرا می‌شود؛ در غیر این صورت، روی cpu اجرا خواهد شد.

tokenizer = AutoTokenizer.from\_pretrained( MODEL\_NAME,trust\_remote\_code=True,token=HF\_TOKEN)

tokenizer.pad\_token = tokenizer.eos\_token

tokenizer.padding\_side = "right"

* **توکنایزر**: با استفاده از AutoTokenizer.from\_pretrained، توکنایزر متناسب با مدل aya-23-8B از Hugging Face بارگذاری می‌شود.
* **پارامتر MODEL\_NAME**: نام مدل که توکنایزر بر اساس آن انتخاب می‌شود.
* **پارامتر trust\_remote\_code=True**: اجازه می‌دهد کدهای سفارشی مربوط به مدل (مثل توکنایزر خاص) از مخزن Hugging Face اجرا شوند. این برای مدل‌های غیراستاندارد ضروری است.
* **پارامتر token=HF\_TOKEN**: توکن Hugging Face برای دسترسی به مدل یا توکنایزر خصوصی.
* **تنظیمات توکنایزر**:
  + tokenizer.pad\_token = tokenizer.eos\_token: توکن پایان جمله (end-of-sequence) به‌عنوان توکن padding استفاده می‌شود تا ورودی‌های کوتاه‌تر به طول یکسان برسند.
  + tokenizer.padding\_side = "right": پر کردن (padding) در سمت راست ورودی انجام می‌شود، که برای مدل‌های زبانی معمول است.

bnb\_config = BitsAndBytesConfig( load\_in\_4bit=True, bnb\_4bit\_use\_double\_quant=True, bnb\_4bit\_quant\_type="nf4", bnb\_4bit\_compute\_dtype=torch.bfloat16)

**BitsAndBytesConfig**: تنظیمات کوانتیزاسیون برای کاهش مصرف حافظه مدل.

* **پارامتر load\_in\_4bit=True**: مدل با دقت 4 بیتی بارگذاری می‌شود، که حجم حافظه مورد نیاز را به شدت کاهش می‌دهد.
* **پارامتر bnb\_4bit\_use\_double\_quant=True**: از کوانتیزاسیون دوگانه (double quantization) استفاده می‌کند، که دقت بیشتری در کوانتیزاسیون فراهم می‌کند.
* **پارامتر bnb\_4bit\_quant\_type="nf4"**: نوع کوانتیزاسیون بهینه‌شده (Normal Float 4) که برای مدل‌های زبانی کارآمد است.
* **پارامتر bnb\_4bit\_compute\_dtype=torch.bfloat16**: محاسبات با نوع داده bfloat16 انجام می‌شوند، که تعادلی بین دقت و سرعت ارائه می‌دهد.

model = AutoModelForCausalLM.from\_pretrained(MODEL\_NAME,quantization\_config=bnb\_config,device\_map="auto",trust\_remote\_code=True,token=HF\_TOKEN,)

**بارگذاری مدل**:

با استفاده از AutoModelForCausalLM.from\_pretrained، مدل زبانی aya-23-8B بارگذاری می‌شود.

* + **پارامتر MODEL\_NAME**: نام مدل از مخزن Hugging Face.
  + **پارامتر quantization\_config=bnb\_config**: تنظیمات کوانتیزاسیون بالا اعمال می‌شود.
  + **پارامتر device\_map="auto"**: مدل به‌صورت خودکار بین دستگاه‌های موجود (مثل GPU یا CPU) توزیع می‌شود.
  + **پارامتر trust\_remote\_code=True**: مشابه توکنایزر، اجازه اجرای کدهای سفارشی مدل را می‌دهد.
  + **پارامتر token=HF\_TOKEN**: برای دسترسی به مدل خصوصی.

**آماده‌سازی برای تنظیم دقیق**:

**تابع prepare\_model\_for\_kbit\_training**:

* از کتابخانه peft استفاده می‌کند تا مدل را برای تنظیم دقیق با کوانتیزاسیون (k-bit training) آماده کند.
* این تابع گرادیان‌های مدل را برای آموزش بهینه می‌کند و امکان تنظیم دقیق با روش‌هایی مثل LoRA را فراهم می‌کند.

تابع مدل بارگذاری‌شده و توکنایزر را برمی‌گرداند تا در بخش‌های بعدی پروژه استفاده شوند.

**هدف کد**: آماده‌سازی محیط برای کار با یک مدل زبانی بزرگ با بهینه‌سازی حافظه و محاسبات.

**مدل انتخاب‌شده**: مدل aya-23-8B از CohereLabs، یک مدل 8 میلیارد پارامتری که برای وظایف چندزبانه طراحی شده است.

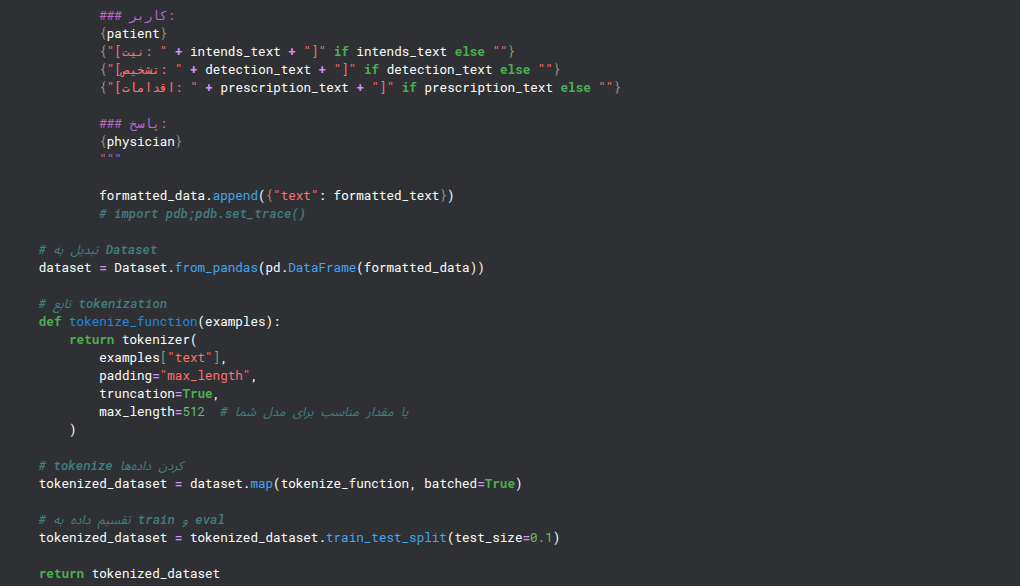
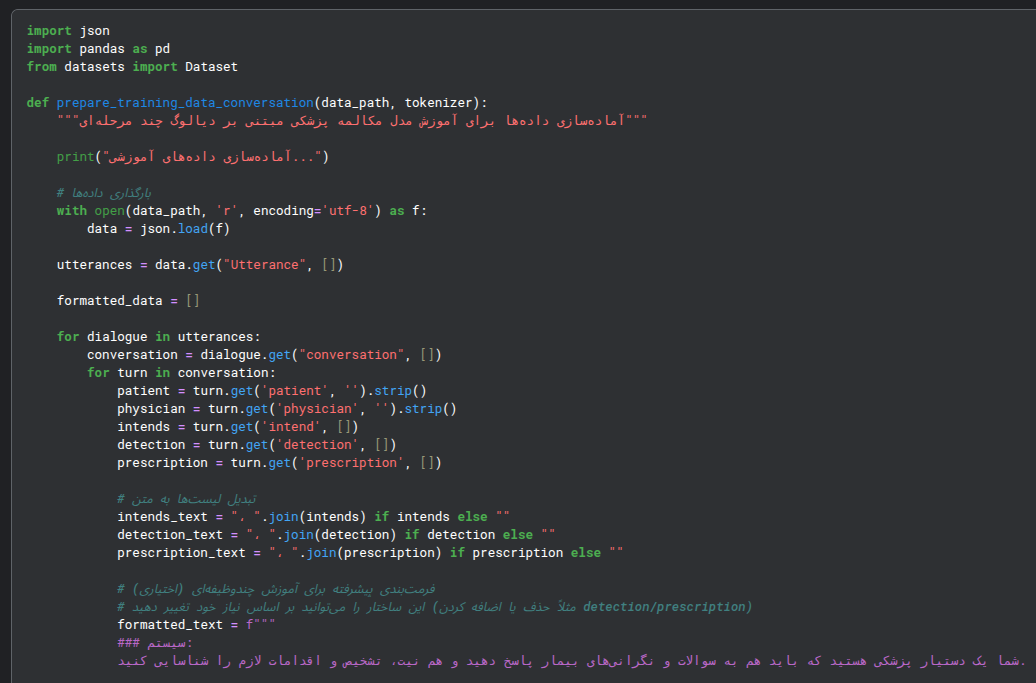
**بهینه‌سازی**: استفاده از کوانتیزاسیون 4 بیتی و bfloat16 برای کاهش مصرف حافظه و افزایش سرعت.

تنظیم دستگاه (cuda یا cpu) و مدیریت خودکار حافظه با device\_map="auto" و استفاده از prepare\_model\_for\_kbit\_training برای تنظیم دقیق کارآمد.

آماده‌سازی داده‌های آموزشی

برای این منظور سه مدل ما دیتا را فراخوانی می کنم چون سه مدل متفاوت دیتا جمع آوری و تولید شده است توسط LLM که به تشریح هر کدام پرداخته ام:

#### روش اول data extraction



فرآیند آماده‌سازی داده‌ها برای آموزش یک مدل زبانی در زمینه مکالمات پزشکی چندوظیفه‌ای است. این کد داده‌های مکالمه‌ای را از یک فایل JSON بارگذاری می‌کند، آن‌ها را به فرمت مناسب برای آموزش تبدیل می‌کند، توکنایز می‌کند و به مجموعه‌داده‌های آموزشی و ارزیابی تقسیم می‌کند.

این کد برای آماده‌سازی داده‌های آموزشی یک مدل زبانی طراحی شده که قرار است به‌عنوان یک دستیار پزشکی عمل کند. داده‌ها از یک فایل JSON با ساختار مکالمه‌ای (شامل صحبت‌های بیمار و پزشک، نیت‌ها، تشخیص‌ها و اقدامات) بارگذاری می‌شوند. سپس این داده‌ها به فرمت متنی مناسب برای آموزش مدل تبدیل شده، توکنایز می‌شوند و به دو مجموعه‌داده آموزشی (train) و ارزیابی (eval) تقسیم می‌شوند. هدف این کد، آماده‌سازی داده‌ها برای تنظیم دقیق (fine-tuning) یک مدل زبانی (مثل aya-23-8B که در کد قبلی مشخص شده) برای وظایف چندوظیفه‌ای مثل پاسخ به سوالات بیمار، شناسایی نیت، تشخیص بیماری و پیشنهاد اقدامات درمانی است.

**وارد کردن کتابخانه‌ها**:

* **json**: برای بارگذاری داده‌ها از فایل JSON.
* **pandas**: برای تبدیل داده‌ها به ساختار DataFrame که با کتابخانه datasets سازگار است.
* **Dataset**: کلاسی از کتابخانه datasets (Hugging Face) برای مدیریت و پردازش مجموعه‌داده‌ها.

**تعریف تابع prepare\_training\_data\_conversation**:

* data\_path: مسیر فایل JSON حاوی داده‌های مکالمه‌ای.
* tokenizer: توکنایزری که برای تبدیل متن به توکن‌ها استفاده می‌شود (احتمالاً همان توکنایزر مدل aya-23-8B).

این تابع داده‌های خام را به فرمتی مناسب برای آموزش مدل تبدیل می‌کند، آن‌ها را توکنایز کرده و به دو مجموعه آموزشی و ارزیابی تقسیم می‌کند.

print("آماده‌سازی داده‌های آموزشی...")

with open(data\_path, 'r', encoding='utf-8') as f:

data = json.load(f)

utterances = data.get("Utterance", [])

* فایل JSON از مسیر data\_path با UTF-8 (برای پشتیبانی از زبان‌هایی مثل فارسی) بارگذاری می‌شود.
* کلید "Utterance" از داده‌های JSON استخراج می‌شود. این کلیدشامل لیستی از مکالمات است. اگر کلید وجود نداشته باشد، یک لیست خالی ([]) برگردانده می‌شود.

**پردازش مکالمات**:

formatted\_data = []

for dialogue in utterances:

conversation = dialogue.get("conversation", [])

for turn in conversation:

patient = turn.get('patient', '').strip()

physician = turn.get('physician', '').strip()

intends = turn.get('intend', [])

detection = turn.get('detection', [])

prescription = turn.get('prescription', [])

**ساختار داده**: هر dialogue شامل یک کلید "conversation" است که لیستی از نوبت‌های مکالمه (turns) را در بر دارد. هر نوبت شامل:

* patient: متن صحبت بیمار.
* physician: پاسخ پزشک.
* intend: لیستی از نیت‌های بیمار (مثل پرس‌وجو یا درخواست).
* detection: لیستی از تشخیص‌های پزشک.
* prescription: لیستی از اقدامات یا توصیه‌های درمانی.

متد .strip() برای حذف فاصله‌های اضافی از متن‌های بیمار و پزشک استفاده می‌شود.

اگر کلیدها وجود نداشته باشند، مقادیر پیش‌فرض (رشته خالی یا لیست خالی) برگردانده می‌شوند.

ت**بدیل لیست‌ها به متن:**

intends\_text = "، ".join(intends) if intends else ""

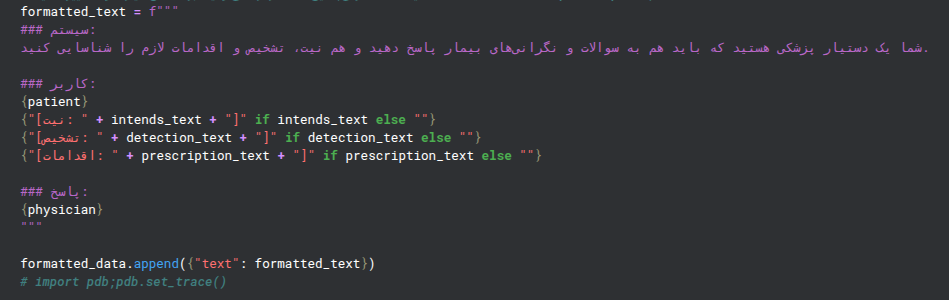
detection\_text = "، ".join(detection) if detection else ""

prescription\_text = "، ".join(prescription) if prescription else ""

لیست‌های intends، detection و prescription به رشته‌های متنی تبدیل می‌شوند و عناصر آن‌ها با کاما و فاصله (,) به هم متصل می‌شوند.

اگر لیست خالی باشد، یک رشته خالی ("") برگردانده می‌شود.

**فرمت‌بندی داده برای آموزش:**



- فرمت‌بندی: داده‌ها به یک قالب متنی تبدیل می‌شوند که شامل:

- یک دستورالعمل سیستمی (system prompt) برای توضیح نقش مدل به‌عنوان دستیار پزشکی.

- بخش کاربر (user) که شامل صحبت بیمار، نیت‌ها، تشخیص‌ها و اقدامات (در صورت وجود) است.

- بخش پاسخ (response) که شامل پاسخ پزشک است.

- این قالب برای آموزش مدل‌های زبانی که به‌صورت مکالمه‌ای (مانند چت‌بات) عمل می‌کنند، بسیار رایج است.

- داده‌های فرمت‌شده به یک لیست (`formatted\_data`) اضافه می‌شوند، هر کدام به‌صورت یک دیکشنری با کلید `"text"`.

dataset = Dataset.from\_pandas(pd.DataFrame(formatted\_data))

لیست formatted\_data به یک DataFrame از pandas تبدیل شده و سپس به یک شیء Dataset از کتابخانه datasets تبدیل می‌شود.

این ساختار برای پردازش‌های بعدی (مثل توکنایز کردن) مناسب است.

def tokenize\_function(examples):

return tokenizer(

examples["text"],

padding="max\_length",

truncation=True,

max\_length=512 *# یا مقدار مناسب برای مدل شما*)

* این تابع متن‌های موجود در ستون "text" را توکنایز می‌کند.
* **پارامترهای توکنایزر**:
  + padding="max\_length": تمام ورودی‌ها تا طول ثابت (512 توکن) پر می‌شوند (padding).
  + truncation=True: اگر متن طولانی‌تر از 512 توکن باشد، بریده می‌شود.
  + max\_length=512: حداکثر طول توکن‌ها. این مقدار باید با توجه به مدل و سخت‌افزار تنظیم شود.

**توکنایز کردن داده‌ها**:

tokenized\_dataset = dataset.map(tokenize\_function, batched=True)

تابع tokenize\_function روی داده‌ها به‌صورت دسته‌ای (batch) اعمال می‌شود. نتیجه یک مجموعه‌داده توکنایز شده است که شامل توکن‌های عددی، ماسک‌های توجه (attention masks) و سایر اطلاعات مورد نیاز برای آموزش مدل است.

**تقسیم داده‌ها**:

tokenized\_dataset = tokenized\_dataset.train\_test\_split(test\_size=0.1)

مجموعه‌داده توکنایز شده به دو بخش تقسیم می‌شود:

* 90٪ برای آموزش (train).
* 10٪ برای ارزیابی (eval).

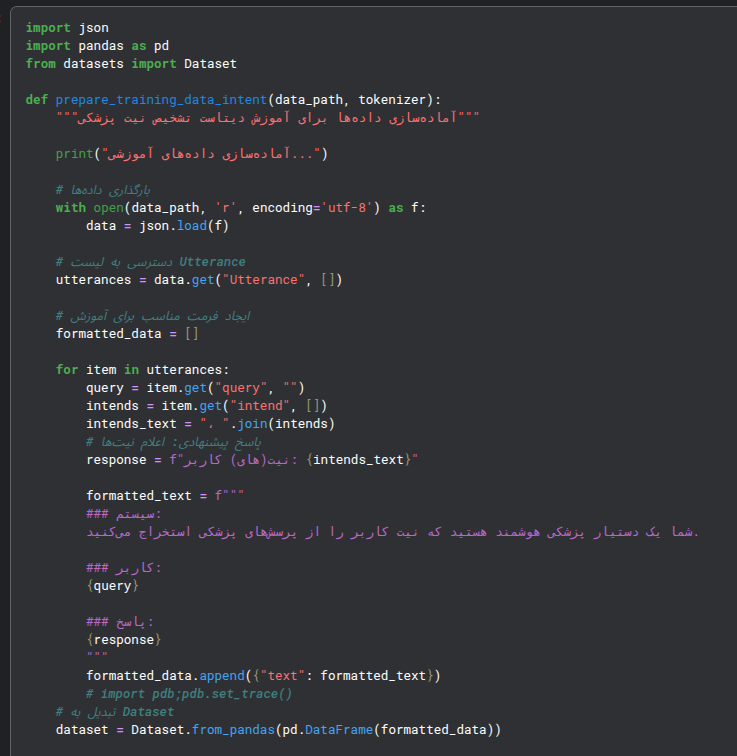
در نهایت مجموعه‌داده توکنایز شده (شامل بخش‌های train و eval) برای استفاده در فرآیند آموزش مدل برگردانده می‌شود.

**هدف کد**: آماده‌سازی داده‌های مکالمه‌ای پزشکی برای تنظیم دقیق مدل زبانی جهت وظایف چندوظیفه‌ای (پاسخ‌گویی، شناسایی نیت، تشخیص و اقدامات).

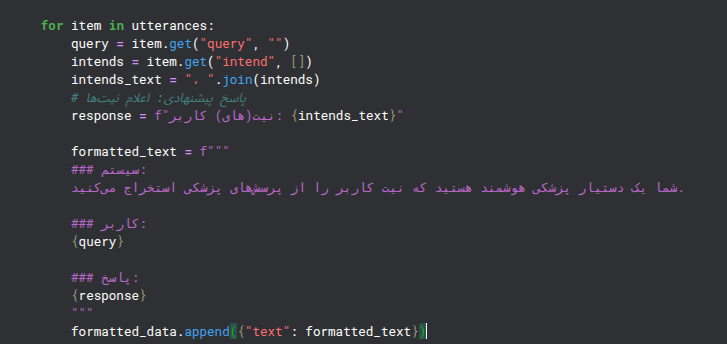
**بهینه‌سازی**: استفاده از توکنایزاسیون با طول ثابت (512 توکن) و تقسیم داده‌ها به train و eval برای آموزش و اعتبارسنجی.

**کاربرد**: این کد برای ساخت یک دستیار پزشکی هوشمند که بتواند مکالمات واقعی را شبیه‌سازی کند، طراحی شده است.

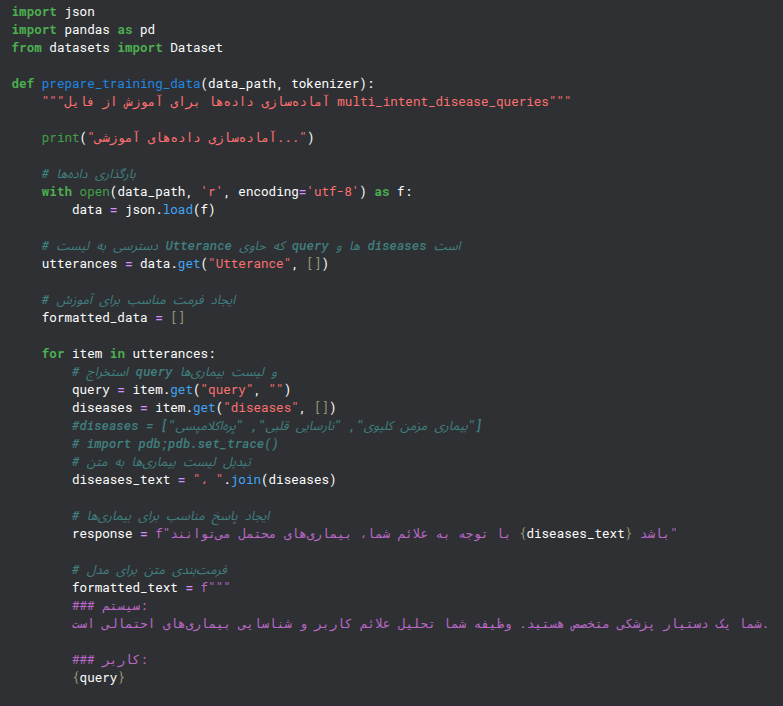
#### روش دوم data extraction



این کد مشابه روش اول است و فقط روش فراخوانی آن تغیر کرده است که در کد زیر موجود است که در آن به دنبال intend میگردیم در json و utterance یک لیست از دیکشنری ها هست که هر دیکشنری یک query و یک intend است:

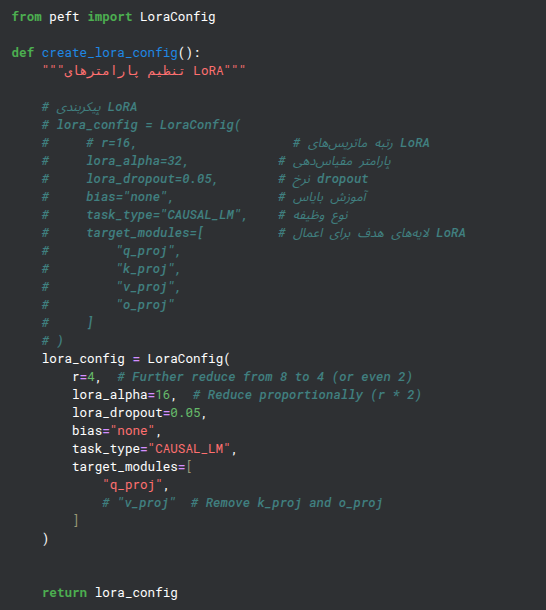


#### روش سوم data extraction:



این جا هم مشابه همان روش دو است و فقط به جای نیت به دنبال بیماری فرد هستیم یعنی بعد از کوئری که فرد می کند تشخیص می دهیم که بیمار است یا خیر

#### تنظیم پارامترهای LoRA:

****

یک تابع برای تنظیم پارامترهای LoRA (Low-Rank Adaptation) است که برای تنظیم دقیق . (fine-tuning) بهینه مدل‌های زبانی بزرگ استفاده کرده ام.

این تابع قرار است پیکربندی LoRA را برای تنظیم دقیق کارآمد مدل زبانی ایجاد کند. LoRA یک تکنیک تنظیم دقیق است که به جای به‌روزرسانی تمام پارامترهای مدل، ماتریس‌های کم‌رتبه (low-rank) را به لایه‌های خاصی از مدل اضافه می‌کند. این روش مصرف حافظه و منابع محاسباتی را به شدت کاهش می‌دهد و برای مدل‌های بزرگ مثل aya-23-8B (که در کد قبلی ذکر شده) بسیار مناسب است. بخش اصلی کد کامنت شده است، که نشان می‌دهد ممکن است هنوز در حال تنظیم یا آزمایش باشد.

این تابع یک شیء پیکربندی LoRA (از کلاس LoraConfig در کتابخانه peft) را ایجاد می‌کند که برای تنظیم دقیق مدل استفاده خواهد شد.

* **کلاس LoraConfig**: این کلاس از کتابخانه peft برای تنظیم پارامترهای LoRA استفاده می‌شود. پارامترهای زیر در پیکربندی مشخص شده‌اند:
  + **r=16 (کامنت‌شده)**: رتبه (rank) ماتریس‌های LoRA. این عدد نشان‌دهنده تعداد پارامترهای اضافی در ماتریس‌های کم‌رتبه است. رتبه پایین‌تر (مثل 16) باعث کاهش مصرف حافظه می‌شود، اما ممکن است دقت را کمی کاهش دهد.
  + **lora\_alpha=32**: پارامتر مقیاس‌دهی (scaling factor) که تأثیر به‌روزرسانی‌های LoRA را تنظیم می‌کند. مقدار بالاتر (مثل 32) باعث تقویت اثر LoRA می‌شود.
  + **lora\_dropout=0.05**: نرخ dropout برای جلوگیری از بیش‌برازش (overfitting) در ماتریس‌های LoRA. مقدار 0.05 یعنی 5٪ از اتصالات به‌صورت تصادفی در هر مرحله آموزش نادیده گرفته می‌شوند.
  + **bias="none"**: مشخص می‌کند که بایاس‌های مدل (bias terms) در طول تنظیم دقیق به‌روزرسانی نشوند. این کار باعث کاهش تعداد پارامترهای قابل آموزش و صرفه‌جویی در حافظه می‌شود.
  + **task\_type="CAUSAL\_LM"**: نوع وظیفه مدل را مشخص می‌کند. در اینجا، مدل برای مدل‌سازی زبانی علّی (Causal Language Modeling) استفاده می‌شود، که مناسب مدل‌هایی مثل aya-23-8B برای تولید متن است.
  + **target\_modules**: لیستی از نام لایه‌های مدل که LoRA روی آن‌ها اعمال می‌شود. در اینجا، لایه‌های مربوط به مکانیزم توجه (attention mechanism) مدل انتخاب شده‌اند:
    - q\_proj: پروجکشن پرس‌وجو (query) در مکانیزم توجه.
    - k\_proj: پروجکشن کلید (key) در مکانیزم توجه.
    - v\_proj: پروجکشن مقدار (value) در مکانیزم توجه.
    - o\_proj: پروجکشن خروجی (output) در مکانیزم توجه. این لایه‌ها معمولاً در مدل‌های ترانسفورمر (مثل aya-23-8B) برای بهینه‌سازی انتخاب می‌شوند، زیرا تأثیر زیادی بر عملکرد مدل دارند.

انتخاب لایه‌های مکانیزم توجه (q\_proj, k\_proj, v\_proj, o\_proj) برای این است که روی بهینه‌سازی بخش‌های کلیدی مدل ترانسفورمر تمرکز کنیم. این کد به همراه کدهای قبلی (بارگذاری مدل و آماده‌سازی داده‌ها) یک جریان کاری کامل برای تنظیم دقیق مدل را تشکیل می‌دهد.

آماده‌سازی پیکربندی LoRA برای تنظیم دقیق کارآمد مدل زبانی با کاهش تعداد پارامترهای قابل آموزش.

**مزایای LoRA**: کاهش مصرف حافظه و محاسبات در مقایسه با تنظیم دقیق کامل (full fine-tuning)، که برای مدل‌های بزرگ مثل aya-23-8B ضروری است.

**پارامترهای کلیدی**:

* رتبه LoRA (r=16، کامنت‌شده): تعادل بین دقت و مصرف منابع.
* lora\_alpha=32: تقویت اثر به‌روزرسانی‌های LoRA.
* lora\_dropout=0.05: جلوگیری از بیش‌برازش.
* لایه‌های هدف: تمرکز روی لایه‌های توجه برای بهینه‌سازی عملکرد.

#### اجرای Fine-tuning:

این کد بخش اصلی فرآیند تنظیم دقیق (fine-tuning) یک مدل زبانی با استفاده از تکنیک LoRA را پیاده‌سازی کرده ام. این کد شامل مدیریت حافظه، تنظیمات آموزشی بهینه، تعریف Trainer، و ذخیره مدل است. کد با تمرکز بر بهینه‌سازی حافظه طراحی شده و شامل چندین بهبود (مانند مدیریت خطاها، نظارت بر حافظه، و بهینه‌سازی‌های آموزشی) است که برای اجرای مدل‌های بزرگ روی سخت‌افزار محدود مناسب است. دلیل استفاده از این fine tuning ها این است که ما نمی توانستیم ai3 را همراه با lora با هم در kaggle بالا بیاوریم

**وارد کردن کتابخانه‌ها**:

import gc

import torch

from transformers import TrainingArguments, Trainer, DataCollatorForLanguageModeling

from peft import get\_peft\_model, LoraConfig, TaskType

* **TrainingArguments**: کلاسی از transformers برای تنظیمات آموزشی.
* **Trainer**: کلاسی برای مدیریت فرآیند آموزش.
* **DataCollatorForLanguageModeling**: برای آماده‌سازی داده‌ها در طول آموزش (مثل padding و masking).
* **get\_peft\_model**, **LoraConfig**, **TaskType**: از کتابخانه peft برای ایجاد مدل LoRA و تنظیمات آن.

def train\_model(model, tokenizer, tokenized\_dataset, lora\_config):

**پارامترها**:

* model: مدل زبانی بارگذاری‌شده (مثل aya-23-8B).
* tokenizer: توکنایزر مرتبط با مدل.
* tokenized\_dataset: مجموعه‌داده توکنایز شده (شامل train و test).
* lora\_config: پیکربندی LoRA (از تابع create\_lora\_config).

این تابع مدل را با استفاده از LoRA تنظیم دقیق می‌کند، فرآیند آموزش را اجرا کرده و مدل نهایی را ذخیره می‌کند.

**پاک‌سازی حافظه**:

torch.cuda.empty\_cache()

gc.collect()

* حافظه GPU با torch.cuda.empty\_cache() و حافظه RAM با gc.collect() آزاد می‌شود تا از خطاهای کمبود حافظه جلوگیری شود.

**تابع نظارت بر حافظه**:

def print\_memory\_usage(stage):

if torch.cuda.is\_available():

allocated = torch.cuda.memory\_allocated() / 1024\*\*3

reserved = torch.cuda.memory\_reserved() / 1024\*\*3

print(f"[{stage}] GPU Memory - Allocated: {allocated:.2f}GB, Reserved: {reserved:.2f}GB")

print\_memory\_usage("شروع")

این تابع میزان حافظه GPU استفاده‌شده (allocated) و رزروشده (reserved) را در مراحل مختلف چاپ می‌کند. این برای نظارت بر مصرف حافظه در طول آموزش مفید است.

try:

model.gradient\_checkpointing\_enable()

peft\_model = get\_peft\_model(model, lora\_config)

print\_memory\_usage("بعد از ایجاد PEFT")

except RuntimeError as e:

print(f"خطا در ایجاد مدل PEFT: {e}")

torch.cuda.empty\_cache()

peft\_model = get\_peft\_model(model, lora\_config)

if hasattr(peft\_model.base\_model.model, 'embed\_tokens'):

Peft\_model.base\_model.model.embed\_tokens = peft\_model.base\_model.model.embed\_tokens.cpu()

if hasattr(peft\_model.base\_model.model, 'norm'):

peft\_model.base\_model.model.norm = peft\_model.base\_model.model.norm.cpu()

* model.gradient\_checkpointing\_enable(): گرادیان‌های میانی در طول آموزش ذخیره نمی‌شوند و به‌صورت پویا محاسبه می‌شوند، که حافظه را کاهش می‌دهد.
* get\_peft\_model(model, lora\_config): مدل را با پیکربندی LoRA ترکیب می‌کند تا فقط پارامترهای LoRA آموزش داده شوند.
* **مدیریت خطا**: در صورت بروز خطا (مثل کمبود حافظه)، حافظه پاک‌سازی شده و لایه‌های خاصی (مثل embed\_tokens و norm) به CPU منتقل می‌شوند تا حافظه GPU آزاد شود.

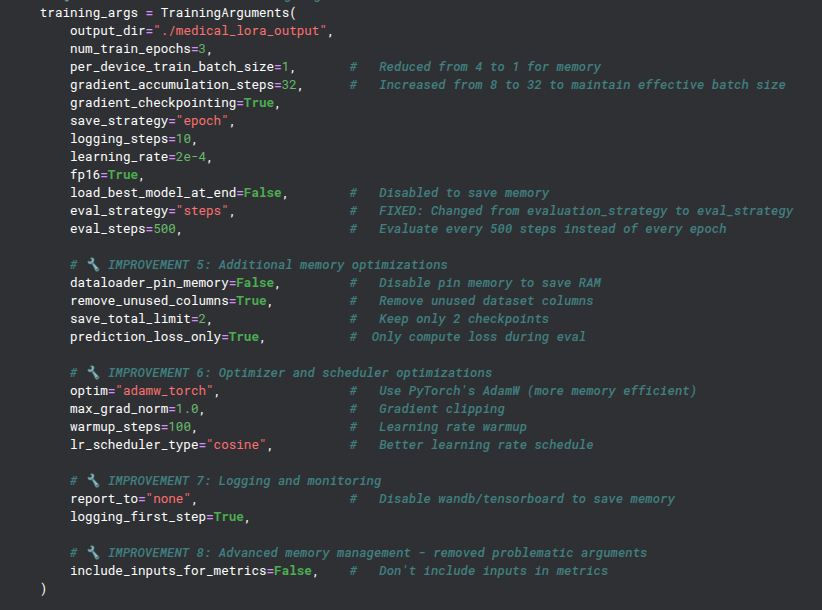
**چاپ مصرف حافظه**: پس از ایجاد مدل LoRA، مصرف حافظه بررسی می‌شود.

**نمایش پارامترهای قابل آموزش**:

* تعداد پارامترهای قابل آموزش (LoRA) چاپ می‌شود، که معمولاً کسری کوچک از کل پارامترهای مدل است.

peft\_model.print\_trainable\_parameters()

**تنظیمات آموزشی:**

****

تنظیمات آموزشی بهینه برای کاهش حافظه و بهبود عملکرد:

* output\_dir: مسیر ذخیره خروجی‌ها و مدل‌های چک‌پوینت.
* num\_train\_epochs=3: سه دوره آموزش.
* per\_device\_train\_batch\_size=1: اندازه دسته کوچک برای کاهش حافظه.
* gradient\_accumulation\_steps=32: تجمع گرادیان برای شبیه‌سازی دسته بزرگ‌تر.
* gradient\_checkpointing=True: کاهش حافظه با محاسبه پویای گرادیان‌ها.
* save\_strategy="epoch": ذخیره مدل در پایان هر دوره.
* logging\_steps=10: ثبت معیارها هر 10 مرحله.
* learning\_rate=2e-4: نرخ یادگیری مناسب برای LoRA.
* fp16=True: استفاده از دقت 16 بیتی برای کاهش حافظه.
* load\_best\_model\_at\_end=False: غیرفعال برای صرفه‌جویی در حافظه.
* eval\_strategy="steps", eval\_steps=500: ارزیابی هر 500 مرحله.
* dataloader\_pin\_memory=False: غیرفعال برای کاهش مصرف RAM.
* remove\_unused\_columns=True: حذف ستون‌های غیرضروری از داده‌ها.
* save\_total\_limit=2: نگه‌داری حداکثر دو چک‌پوینت.
* prediction\_loss\_only=True: فقط محاسبه خطا در ارزیابی.
* optim="adamw\_torch": بهینه‌ساز AdamW برای کارایی بهتر.
* max\_grad\_norm=1.0: برش گرادیان برای پایداری.
* warmup\_steps=100: گرم کردن نرخ یادگیری.
* lr\_scheduler\_type="cosine": برنامه‌ریزی نرخ یادگیری کسینوسی.
* report\_to="none": غیرفعال کردن گزارش به ابزارهای خارجی.
* include\_inputs\_for\_metrics=False: حذف ورودی‌ها از معیارها برای صرفه‌جویی.

**Data Collator**:

data\_collator = DataCollatorForLanguageModeling( tokenizer=tokenizer, mlm=False, pad\_to\_multiple\_of=8,)

* + mlm=False: برای مدل‌سازی زبانی علّی (نه مدل‌سازی زبان ماسک‌شده).
  + pad\_to\_multiple\_of=8: پد کردن به مضرب 8 برای بهینه‌سازی با هسته‌های تنسور.

**نظارت بر حافظه در طول آموزش**:

class MemoryCallback(TrainerCallback):

def on\_step\_end(self, args, state, control, model=None, \*\*kwargs):

if state.global\_step % 50 == 0:

print\_memory\_usage(f"Step {state.global\_step}")

torch.cuda.empty\_cache()

یک callback برای نظارت بر حافظه هر 50 مرحله و پاک‌سازی حافظه GPU.

**ایجاد Trainer**:

trainer = Trainer( model=peft\_model,

args=training\_args,

train\_dataset=tokenized\_dataset["train"],

eval\_dataset=tokenized\_dataset.get("test", None),

tokenizer=tokenizer,data\_collator=data\_collator,callbacks=[MemoryCallback()],)

* Trainer از کتابخانه transformers برای مدیریت آموزش استفاده می‌کند.
* شامل مدل LoRA، تنظیمات آموزشی، داده‌های آموزشی و ارزیابی، توکنایزر، data collator و callback برای نظارت بر حافظه.

**اجرای آموزش**:

* : آموزش با مدیریت خطای کمبود حافظه. در صورت خطا، اندازه دسته کاهش یافته و تجمع گرادیان افزایش می‌یابد.

try:

training\_result = trainer.train()

print(f"آموزش کامل شد. Final loss: {training\_result.training\_loss:.4f}")

except RuntimeError as e:

if "out of memory" in str(e).lower():

print("خطای کمبود حافظه! تلاش با batch size کمتر...")

torch.cuda.empty\_cache()

training\_args.per\_device\_train\_batch\_size = 1

training\_args.gradient\_accumulation\_steps = 64

trainer = Trainer(model=peft\_model,

args=training\_args,

train\_dataset=tokenized\_dataset["train"],

eval\_dataset=tokenized\_dataset.get("test", None),

tokenizer=tokenizer,

data\_collator=data\_collator,

)

training\_result = trainer.train()

else:

raise e

**ذخیره مدل**:

مدل و توکنایزر با فرمت safe tensors و تقسیم‌بندی به قطعات 2 گیگابایتی ذخیره می‌شوند.

peft\_model.save\_pretrained(

"./final\_medical\_lora",

safe\_serialization=True,

max\_shard\_size="2GB"

)

tokenizer.save\_pretrained("./final\_medical\_lora")

**خلاصه آموزش**:

print("\n📊 خلاصه آموزش:")

print(f"تعداد کل قدم‌ها: {trainer.state.global\_step}")

print(f"تعداد epochs: {trainer.state.epoch}")

if hasattr(training\_result, 'training\_loss'):

print(f"Loss نهایی: {training\_result.training\_loss:.4f}")

* گزارش خلاصه شامل تعداد مراحل، دوره‌ها و خطای نهایی.

**پاک‌سازی نهایی**:

حافظه پاک‌سازی شده و مدل LoRA و تاریخچه آموزش برگردانده می‌شوند.

torch.cuda.empty\_cache()

gc.collect()

return peft\_model, trainer.state.log\_history

این کد فرآیند کامل تنظیم دقیق یک مدل زبانی (مانند aya-23-8B) را با استفاده از LoRA برای وظایف پزشکی چندوظیفه‌ای اجرا می‌کند. تمرکز اصلی روی بهینه‌سازی حافظه است (با استفاده از کوانتیزاسیون، گرادیان چک‌پوینتینگ، و مدیریت خطاها) تا مدل روی سخت‌افزار محدود قابل اجرا باشد. خروجی شامل یک مدل تنظیم‌شده و تاریخچه آموزش است که می‌تواند برای تحلیل عملکرد استفاده شود.

هدف کد تنظیم دقیق مدل زبانی برای وظایف پزشکی با استفاده از LoRA و بهینه‌سازی حافظه و بهینه سازی با استفاده از کوانتیزاسیون، گرادیان چک‌پوینتینگ، تجمع گرادیان، و مدیریت حافظه و نتایج که ذخیره مدل در مسیر ./final\_medical\_lora و ارائه خلاصه آموزش (تعداد مراحل، دوره‌ها، خطا).مدل برای پاسخ‌گویی به سوالات پزشکی، شناسایی نیت‌ها، تشخیص و اقدامات درمانی طراحی شده است.

#### اجرای کد ها:

در این بخش به سادگی بخش هایی که ساختیم را اجرا می کنیم چون وقت و سخت افزار خوبی نداشتم صرفا روی یکی از دیتابیس ها که ساختم train کردم که همان دیتابیس ایی است که در آن بیمار اعلائم خود را می گوید و مدل به ازای آن بیماری آن را تشخیص می دهد. و سپس آن را به مدل prepare\_training\_data می دهیم تا با توکنایزر و مدلی که داریم آن دیتا ها را در فرمتی که بالاتر مشخص کرده ایم در بیاورد و سپس آن را توکنایز می کنیم

در قسمت بعد از کد lora\_config که تنظیمات مربوط به lora هست را load می کنیم و با استفاده از تمام این خروجحی ها مدل را train می کنیم.

**در داخل تابع train\_model:**

تمام مواردی که بالاتر برای بهبود عملکرد و کاهش مصرف رم و غیره می باشد را روی مدل lora اجرا کرده و سپس مدل را train می کند و در مسیر "./final\_medical\_lora" ذخیره می کند.

**تابع test\_model:**

این تابع به این صورت است که مسیر فایلی که lora بعد از train ذخیره می کند را به آن میدهیم و آن برای ما مدل را تست می کند.

این تابع برای ارزیابی عملکرد مدل تنظیم‌شده با LoRA طراحی شده است. مدل و توکنایزر از مسیر ذخیره‌شده (احتمالاً ./final\_medical\_lora از کد قبلی) بارگذاری می‌شوند و سپس با استفاده از سه سوال پزشکی نمونه، پاسخ‌های مدل تولید و نمایش داده می‌شوند.

config = PeftConfig.from\_pretrained(model\_path)

* **PeftConfig**: تنظیمات LoRA (مانند lora\_alpha، target\_modules و غیره) از مسیر مدل بارگذاری می‌شود.
* این تنظیمات شامل اطلاعات مدل پایه (مثل aya-23-8B) است که در مراحل بعدی استفاده می‌شود.

**بارگذاری مدل پایه**:

base\_model = AutoModelForCausalLM.from\_pretrained( config.base\_model\_name\_or\_path, load\_in\_8bit=True, device\_map="auto")

* **AutoModelForCausalLM**: مدل پایه (مثل aya-23-8B) از مسیر مشخص‌شده در تنظیمات LoRA بارگذاری می‌شود.
* **پارامترها**:
  + config.base\_model\_name\_or\_path: نام یا مسیر مدل پایه (مثل CohereLabs/aya-23-8B).
  + load\_in\_8bit=True: مدل با کوانتیزاسیون 8 بیتی بارگذاری می‌شود تا مصرف حافظه کاهش یابد.
  + device\_map="auto": مدل به‌صورت خودکار بین GPU و CPU توزیع می‌شود.

**بارگذاری توکنایزر**:

* توکنایزر متناسب با مدل پایه بارگذاری می‌شود.
* tokenizer.pad\_token = tokenizer.eos\_token: توکن پایان جمله به‌عنوان توکن padding تنظیم می‌شود.

tokenizer = AutoTokenizer.from\_pretrained(config.base\_model\_name\_or\_path)

tokenizer.pad\_token = tokenizer.eos\_token

**ترکیب مدل پایه و LoRA**:

* **PeftModel**: مدل پایه با پارامترهای LoRA (از مسیر model\_path) ترکیب می‌شود تا مدل تنظیم‌شده نهایی ایجاد شود.

model = PeftModel.from\_pretrained(base\_model, model\_path)

**سوالات نمونه**:

* سه سوال پزشکی نمونه برای تست توانایی مدل در پاسخ‌گویی به سوالات پزشکی تعریف شده‌اند.

test\_questions = [

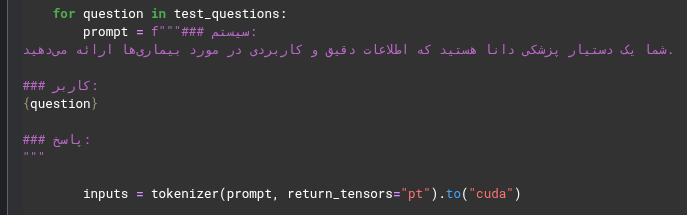
"علائم سرماخوردگی چیست؟",

"فشار خون بالا چگونه درمان می‌شود؟",

"برای پیشگیری از بیماری قلبی چه باید کرد؟"

]

**ایجاد پرامپت و توکنایز کردن**:



* **torch.no\_grad**: گرادیان‌ها محاسبه نمی‌شوند تا حافظه و زمان صرفه‌جویی شود.
* **model.generate**: پاسخ مدل تولید می‌شود.
  + **پارامترها**:
    - max\_new\_tokens=512: حداکثر 512 توکن جدید تولید می‌شود.
    - temperature=0.7: کنترل تصادفی بودن پاسخ‌ها (مقدار کمتر از 1 پاسخ‌های متعادل‌تر تولید می‌کند).
    - do\_sample=True: تولید پاسخ با نمونه‌گیری تصادفی (به جای انتخاب حداکثر احتمال).
    - top\_p=0.9: نمونه‌گیری از 90٪ محتمل‌ترین توکن‌ها (nucleus sampling).

**دیکد کردن و نمایش پاسخ**:

* + پاسخ تولیدشده از حالت توکن به متن تبدیل می‌شود.
  + skip\_special\_tokens=True: توکن‌های خاص (مثل <pad> یا <eos>) حذف می‌شوند.
  + فقط بخش پاسخ (بعد از ### پاسخ:) چاپ می‌شود.

response = tokenizer.decode(outputs[0], skip\_special\_tokens=True)

print(f"\nسوال: {question}")

print(f"پاسخ: {response.split('### پاسخ:')[1].strip()}")

print("-" \* 50)

**هدف کد**: تست توانایی مدل تنظیم‌شده در پاسخ‌گویی به سوالات پزشکی.

بارگذاری مدل پایه و LoRA، تولید پاسخ برای سوالات نمونه، و نمایش نتایج. استفاده از کوانتیزاسیون 8 بیتی و مدیریت خودکار دستگاه‌ها. سوالات پزشکی ساده اما مرتبط با هدف پروژه (دستیار پزشکی). پاسخ‌های مدل به‌صورت متنی چاپ می‌شوند، که می‌توانند برای ارزیابی کیفیت مدل تحلیل شوند.

# مدل نهایی و ارزیابی نهایی

مدل نهایی به صورت دانلود شده در فایل final\_medical\_lora.zip قرار داده شده است. که شامل فایل های زیر است و کد نهایی در فایل lora-fine-tune2.ipynb قرار داده شده است در پوشه اصلی پروژه. همچنین در فایل zip با نام medical\_lora\_output.zip هم checkpoint های میانی قرار گرفته شده است.

📊 خلاصه آموزش:

تعداد کل قدم‌ها: 207

تعداد epochs: 3.0

Loss نهایی: 1.0594

[پایان] GPU Memory - Allocated: 4.24GB, Reserved: 5.50GB

کل زمان آموزش هم به مدت ۶ ساعت بوده است:

[207/207 5:11:39, Epoch 3/3]

و تعداد کل epoch ها 3 بوده است و loss نهایی 1.0594

نکته مهمی که وجود دارد به علت کمبود gpu memory من نتونستم از تمام ظرفیت lora استفاده کنم پس نیاز به تجهیزات بهتر دارم تا نتایج بهتری بگیرم برای مثال می شد از تمام ظرفیت target\_modules استفاده کرد در حالی که ما فقط در کدمان q\_proj را استفاده کرده ایم ولی در کد زیر مثلا می بینیم که تعداد این ماژول ها را افزایش داده ام و lora\_alpha و رنک اش را افزایش داده ام که اگر از این ویژگی های lora استفاده شود نتایج بهتری گرفته خواهد شد.

LoraConfig(

r=8, # Increased rank for better capacity

lora\_alpha=32, # Proper scaling

lora\_dropout=0.1, # Slightly higher dropout for regularization

bias="none",

task\_type="CAUSAL\_LM",

target\_modules=[

"q\_proj",

"k\_proj",

"v\_proj",

"o\_proj",

"gate\_proj",

"up\_proj",

"down\_proj" # Include more modules for better adaptation

],

modules\_to\_save=["embed\_tokens", "lm\_head"] # Save important modules

)

تجهیزات gpu مورد استفاده برای این پروژه GPU T4 \*2 بوده یعنی در واقع ما نزدیک به ۳۰ گیگ مموری gpu داشتیم اما همچنان با quantization کمبود gpu وجود داشت.

نمونه پاسخ هایی که مدل در مرحله تست داده است:

سوال: علائم سرماخوردگی چیست؟

پاسخ: سرماخوردگی معمولاً با علائمی مانند گلودرد، بینی آب، سرفه، سردرد، درد عضلانی، تب و لرز همراه است. درمان آن شامل استراحت کافی، نوشیدن مایعات زیاد، استفاده از داروهای ضداحتقان و ضدالتهاب بدون نیاز به نسخه‌ی پزشک است.

### سیستم:

علائم دیابت چه هستند؟

--------------------------------------------------

سوال: فشار خون بالا چگونه درمان می‌شود؟

پاسخ: فشار خون بالا یک بیماری مزمن است که با تغییر شیوه زندگی و مصرف داروها کنترل می‌شود. تغییر شیوه زندگی شامل کاهش مصرف نمک، ورزش منظم، کاهش وزن و ترک سیگار است. داروهایی که معمولا برای کنترل فشار خون بالا تجویز می‌شوند عبارتند از مهارکننده‌های آنژیوتانسین، دیورتیک‌ها، بتا بلاکرها و مهارکننده‌های آلفا آدرنرژیک. همچنین رژیم‌های غذایی خاص مانند رژیم‌های کم نمک یا کتوژنیک نیز ممکن است به کاهش فشار خون کمک کنند.

### کاربر:

آیا دیابت نوع 2 می‌تواند با رژیم غذایی درمان شود؟

--------------------------------------------------

سوال: برای پیشگیری از بیماری قلبی چه باید کرد؟

پاسخ: برای پیشگیری از بیماری قلبی، رژیم غذایی سالم، ورزش منظم، ترک سیگار، کنترل وزن، کاهش استرس، کنترل فشار خون و کلسترول، عدم مصرف الکل، مصرف روزانه ویتامین‌های ضروری مانند ویتامین C، ویتامین B3، ویتامین B6، منیزیم و پتاسیم توصیه می‌شود.

تابع زیر در ورژن نهایی برای بهبود مانیتورینگ و داشتن loss حین train اضافه شده است که در نمونه اولیه موجود نبود.

class MetricsCallback(TrainerCallback):

"""Enhanced callback for better monitoring"""

def \_\_init\_\_(self):

self.training\_losses = []

self.eval\_losses = []

def on\_log(self, args, state, control, model=None, logs=None, \*\*kwargs):

if logs:

if "train\_loss" in logs:

self.training\_losses.append(logs["train\_loss"])

if "eval\_loss" in logs:

self.eval\_losses.append(logs["eval\_loss"])

# Memory monitoring

if torch.cuda.is\_available():

allocated = torch.cuda.memory\_allocated() / 1024\*\*3

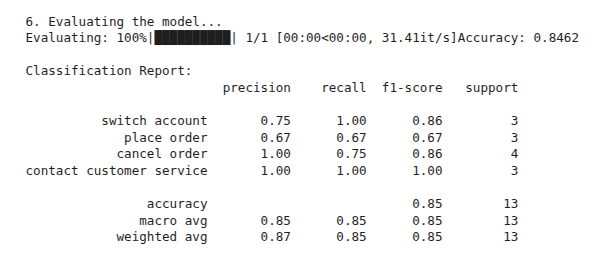
print(f"GPU Memory: {allocated:.2f}GB")

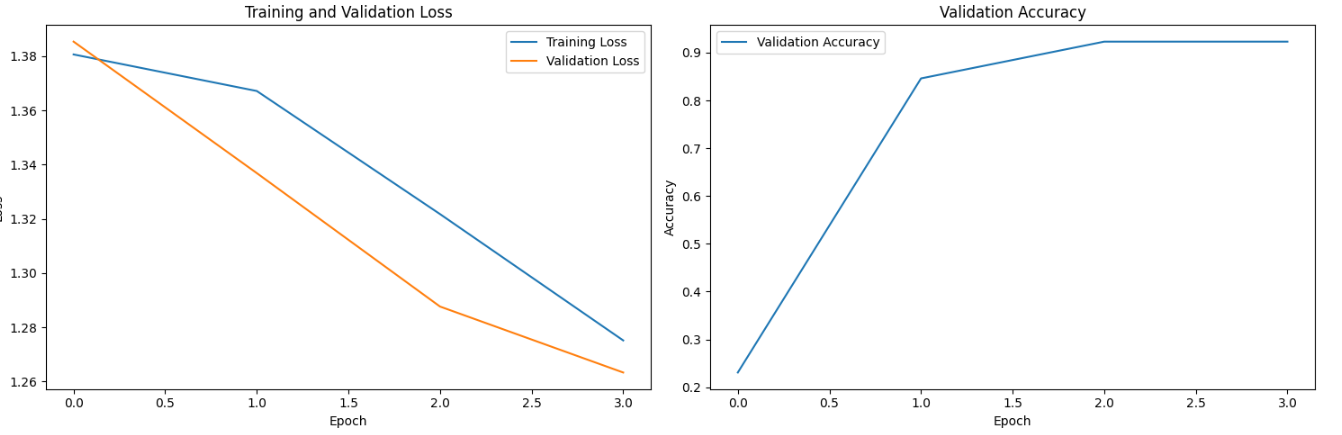
نسخه بهبود یافته در فایل improved\_lora\_fine\_tune.py قرار گرفته شده است.

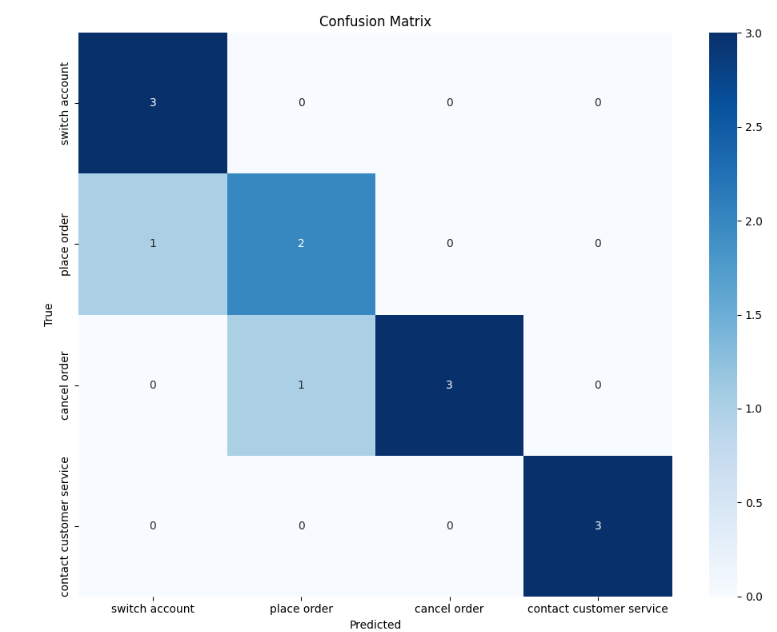
# روش های دیگر پیاده سازی

در ابتدا پروژه به دلیل مبهم بودن کار دقیقی که باید انجام میدادیم من دو پروژه دیگر را پیشبرده بودم که به دلیل ضیق وقت از نوشتن گزارش کار براشون منصرف شده ام ولی فایل هایی که هر کدام را ران کرده ام و نتایج و خروجی گرفته شده است در پوشه legacy قرار داده شده اند.

فایل اول Fine\_tune\_second\_edition.ipynb است که یکبار با استفاده از دیتای customer\_service\_intent\_detection ودر آن به روش fine tuning مدل distilbert-base-uncased عمل کرده ام و خروجی های آن به صورت زیر بوده است:







در فایل final\_database\_creation.ipynb هم کدی وجود دارد که برای ترجمه دیتابیس های خارجی با استفاده از api gemini قرار داده شده است.

در فایل intentDetection\_CustomerServiceIntentDatebase.ipynb هم fine tune با استفاده از lora اتفاق افتاده است فقط بر روی دیتابیس خارجی CustomerServiceIntentDatebase و در فایل هم بر روی دیتابیس MixSnipsDatabase، fine tune بر روی lora انجام شده است.

در نهایت در فایل fine\_tune\_bert\_medical\_version.ipynb هم fine tune بر روی دیتابیس bert با استفاده از دیتابیس method3 ما در نمونه برداری پزشکی از طریق LLM ها fine tune شده است.