

Fine-Tuning Mistral Model On Trec Dataset

Bahar Khouban

990122681002

در این پروژه، پس از انتخاب و آماده‌سازی داده‌های متنی، مدل Mistral با بهره‌گیری از تکنیک LoRA فاین‌تیون می‌شود. هدف از این مرحله، بهبود یادگیری مدل برای تطبیق بهتر با داده‌های جدید و خاص است. به جای تنظیم همه پارامترهای مدل، LoRA تنها تغییرات کوچکی در زیرمجموعه‌ای از پارامترها ایجاد می‌کند که منجر به سرعت بخشیدن به فرآیند آموزش و کاهش مصرف حافظه می‌شود.

مراحل اصلی این پروژه شامل پیش‌پردازش داده‌های متنی، تنظیم مدل Mistral با داده‌های آموزشی و تکنیک LoRA، و ارزیابی مدل فاین‌تیون شده است. در گام نخست، داده‌ها باید به‌طور دقیق و مناسب پیش‌پردازش شوند تا هر گونه نویز یا اطلاعات نامرتب حذف گردد. سپس مدل آموزش دیده و با کمک LoRA تنظیم می‌شود تا به بهترین شکل ممکن با داده‌های جدید تطبیق یابد. در نهایت، عملکرد مدل بر روی مجموعه داده‌های تست بررسی و تحلیل می‌شود تا میزان دقت و کارایی آن در طبقه‌بندی متن‌ها مشخص شود.

نتایج به دست آمده از این پروژه نه تنها به بهبود عملکرد مدل‌های زبان بزرگ مانند Mistral در وظایف طبقه‌بندی متن کمک می‌کند، بلکه می‌تواند به توسعه مدل‌های هوشمندتر و کارآمدتر برای انجام وظایف تخصصی‌تر در حوزه‌های مختلف نیز منجر شود. با استفاده از این تکنیک‌ها، قابلیت‌های مدل‌های زبان بزرگ در مواجهه با داده‌های متنوع و پیچیده ارتقا یافته و امکان اجرای وظایف پیچیده‌تر با دقت بالاتر فراهم می‌گردد.

```
!pip install transformers peft datasets scikit-learn matplotlib  
huggingface_hub bitsandbytes
```

ابتدا پکیج‌های مورد نیاز را که در طول پروژه قرار است از آن استفاده کنیم را نصب می‌کنیم.

transformers: این کتابخانه توسط Hugging Face توسعه یافته و یکی از پرکاربردترین ابزارها برای کار با مدل‌های پیشرفته زبان مانند Mistral، GPT، BERT و غیره است. این کتابخانه امکاناتی را برای load train و استفاده از مدل‌های pre train شده ارائه می‌دهد.

peft: مخفف "Parameter-Efficient Fine-Tuning"، این کتابخانه روش‌های پیشرفته‌ای مانند LoRA را برای تنظیم پارامترهای مدل‌های بزرگ با کارایی بیشتر فراهم می‌کند. هدف این روش‌ها کاهش پیچیدگی محاسباتی و منابع مورد نیاز برای تنظیم دقیق مدل‌ها است.

datasets: کتابخانه‌ای دیگر از Hugging Face که برای مدیریت و کار با مجموعه داده‌ها طراحی شده است. این ابزار امکان دسترسی آسان به مجموعه داده‌های مختلف و همچنین ایجاد و مدیریت مجموعه داده‌ها را فراهم می‌کند.

scikit-learn: یکی از پرکاربردترین کتابخانه‌ها برای یادگیری ماشین در پایتون است که الگوریتم‌های مختلفی برای classification، regression و clustering ارائه می‌دهد. این کتابخانه همچنین شامل ابزارهایی برای ارزیابی مدل‌ها و پیش‌پردازش داده‌ها است.

matplotlib: کتابخانه‌ای قدرتمند برای تجسم داده‌ها که امکان ایجاد انواع نمودارها و گراف‌ها را فراهم می‌کند. این ابزار به خصوص برای تحلیل داده‌ها و نمایش نتایج به صورت بصری بسیار مفید است.

huggingface_hub: این کتابخانه امکان ارتباط با سرورهای Hugging Face را فراهم می‌کند. با استفاده از این ابزار، می‌توانید مدل‌ها، مجموعه داده‌ها و سایر منابع را به راحتی بارگذاری و به اشتراک بگذارید.

bitsandbytes: کتابخانه‌ای است که به بهینه‌سازی مدل‌های یادگیری ماشین کمک می‌کند. این ابزار مخصوصاً برای کار با مدل‌های بزرگ و پیچیده که نیاز به بهینه‌سازی محاسباتی دارند، مفید است.

```
from datasets import load_dataset  
dataset = load_dataset('cogcomp/trec')
```

سپس دیتاست 'cogcomp/trec' را لود می کنیم.

این دیتاست شامل حدودا ۷۰۰۰ رکورد است که حاوی تکست و دو لیبل coarse_label و fine_label می باشد.
به فرمت زیر:

```
{  
  'text': 'How did serfdom develop in and then leave Russia ?',  
  'coarse_label': 2,  
  'fine_label': 26  
}
```

- text (str): Text of the question.
- coarse_label (ClassLabel): Coarse class label.
 - 'ABBR' (0): Abbreviation.
 - 'ENTY' (1): Entity.
 - 'DESC' (2): Description and abstract concept.
 - 'HUM' (3): Human being.
 - 'LOC' (4): Location.
 - 'NUM' (5): Numeric value.
- fine_label (ClassLabel): Fine class label. Possible values are:
 - ABBREVIATION:
 - 'ABBR:abb' (0): Abbreviation.
 - 'ABBR:exp' (1): Expression abbreviated.
 - ENTITY:
 - 'ENTY:animal' (2): Animal.
 - 'ENTY:body' (3): Organ of body.
 - 'ENTY:color' (4): Color.
 - 'ENTY:cremat' (5): Invention, book and other creative piece.
 - 'ENTY:currency' (6): Currency name.
 - 'ENTY:dismed' (7): Disease and medicine.
 - 'ENTY:event' (8): Event.
 - 'ENTY:food' (9): Food.
 - 'ENTY:instru' (10): Musical instrument.
 - 'ENTY:lang' (11): Language.

- 'ENTY:letter' (12): Letter like a-z.
- 'ENTY:other' (13): Other entity.
- 'ENTY:plant' (14): Plant.
- 'ENTY:product' (15): Product.
- 'ENTY:religion' (16): Religion.
- 'ENTY:sport' (17): Sport.
- 'ENTY:substance' (18): Element and substance.
- 'ENTY:symbol' (19): Symbols and sign.
- 'ENTY:techmeth' (20): Techniques and method.
- 'ENTY:termeq' (21): Equivalent term.
- 'ENTY:veh' (22): Vehicle.
- 'ENTY:word' (23): Word with a special property.
- DESCRIPTION:
 - 'DESC:def' (24): Definition of something.
 - 'DESC:desc' (25): Description of something.
 - 'DESC:manner' (26): Manner of an action.
 - 'DESC:reason' (27): Reason.
- HUMAN:
 - 'HUM:gr' (28): Group or organization of persons
 - 'HUM:ind' (29): Individual.
 - 'HUM:title' (30): Title of a person.
 - 'HUM:desc' (31): Description of a person.
- LOCATION:
 - 'LOC:city' (32): City.
 - 'LOC:country' (33): Country.
 - 'LOC:mount' (34): Mountain.
 - 'LOC:other' (35): Other location.
 - 'LOC:state' (36): State.
- NUMERIC:
 - 'NUM:code' (37): Postcode or other code.
 - 'NUM:count' (38): Number of something.
 - 'NUM:date' (39): Date.
 - 'NUM:dist' (40): Distance, linear measure.

- 'NUM:money' (41): Price.
- 'NUM:ord' (42): Order, rank.
- 'NUM:other' (43): Other number.
- 'NUM:period' (44): Lasting time of something
- 'NUM:perc' (45): Percent, fraction.
- 'NUM:speed' (46): Speed.
- 'NUM:temp' (47): Temperature.
- 'NUM:volsize' (48): Size, area and volume.
- 'NUM:weight' (49): Weight.

```
import pandas as pd

df_train = pd.DataFrame(dataset['train'])
df_train.head()
```

سپس قسمت train را از دیتاست اصلی جدا میکنیم و داخل دیتافریم جدیدی میریزیم.

	fine_label	coarse_label	text	
	26	2	...How did serfdom develop in and then leave Russ	0
	5	1	? What films featured the character Popeye Doyle	1
	26	2	...How can I find a list of celebrities ' real na	2
	2	1	...What fowl grabs the spotlight after the Chines	3
	1	0	? What is the full form of .com	4

```
coarse_label_mapping = {
    0: 'ABBR',
    1: 'ENTY',
    2: 'DESC',
    3: 'HUM',
```

```

    4: 'LOC',
    5: 'NUM'
}

df_train['coarse_label'] =
df_train['coarse_label'].map(coarse_label_mapping)
df_train['fine_label'] = df_train['fine_label'].map(fine_label_mapping)

df_train.head()

```

سپس هر کدام از لیبل‌ها مپ میشود به تکست مرتبط با خود.

```

import pandas as pd
from datasets import load_dataset

(['df_test = pd.DataFrame(dataset['test

} = coarse_label_mapping
, 'ABBR' :0
, 'ENTY' :1
, 'DESC' :2
, 'HUM' :3
, 'LOC' :4
, 'NUM' :5
{
df_test['coarse_label'] =
(df_test['coarse_label'].map(coarse_label_mapping
(df_test['fine_label'] = df_test['fine_label'].map(fine_label_mapping

df_test.head() # Display the first few rows of the DataFrame

```

سپس به طور مشابه برای test set این کار انجام می‌شود.

```
df_train = df_train.sample(frac=1, random_state=42).reset_index(drop=True)
```

```

validation_size = int(0.1 * len(df_train))
df_valid = df_train[:validation_size]
df_train = df_train[validation_size:]

print(f"Training set size: {len(df_train)}")
print(f"Validation set size: {len(df_valid)}")

print("Training set sample:")
print(df_train.head())

print("Validation set sample:")
display(df_valid.head())

```

سپس ابتدا داده ها رو به صورت رندوم قاطی می‌کنیم و ۹۰٪ داده ها رو به عنوان مجموعه train در نظر می‌گیریم. این مجموعه داده برای ترین مدل استفاده می‌شود. مدل از این داده‌ها برای یادگیری الگوها و روابط بین ویژگی‌ها و لیبل‌ها استفاده می‌کند. هرچه مجموعه داده‌های ترین بزرگ‌تر و متنوع‌تر باشد، مدل بهتر می‌تواند الگوهای پیچیده را یاد بگیرد.

سپس حدود ۱۰٪ از ترین ست اولیه را به عنوان ولیدیشن ست انتخاب می‌کنیم. این مجموعه برای تنظیم مدل و ارزیابی عملکرد آن در طول فرآیند ترینینگ استفاده می‌شود. با استفاده از مجموعه validation می‌توان عملکرد مدل را روی داده‌هایی که در فرآیند train شرکت نکرده‌اند، ارزیابی کرد. این به جلوگیری از overfitting کمک می‌کند و به تنظیم هایپرامترها کمک می‌کند.

Test set: مجموعه test به صورت جداگانه از مجموعه‌های train و valid نگه داشته می‌شود و برای ارزیابی نهایی مدل به کار می‌رود. در این کد، به مجموعه آزمایشی train نشده است، اما معمولاً در فرآیندهای یادگیری ماشین، ۱۰٪ تا ۲۰٪ از کل داده‌ها به عنوان مجموعه test جدا می‌شوند. این مجموعه داده برای ارزیابی نهایی مدل استفاده می‌شود. پس از اینکه مدل بر روی مجموعه train آموزش دید و با استفاده از مجموعه valid تنظیم شد، مجموعه test برای بررسی عملکرد واقعی مدل بر روی داده‌هایی که هرگز دیده نشده‌اند، استفاده می‌شود. این ارزیابی نهایی، میزان تعمیم‌پذیری مدل را تعیین می‌کند.

```

Training set size: 4907
Validation set size: 545
Training set sample:

```

```

                                text coarse_label \
545 Who was credited with saying : `` I never met ...      HUM
546 Whose special bear 's creator was born on Janu...      HUM

```

547	Who commanded the French forces at the Battle ...	HUM
548	What was Mark Johnson referring to when he sai...	ENTY
549	What is the abbreviation for Original Equipmen...	ABBR

```

        fine_label
545     HUM:ind
546     HUM:ind
547     HUM:ind
548  ENTY:termeq
549     ABBR:abb

```

تبدیل دیتاست:

```

def generate_prompt(data_point):
    return f"""
        [INST]Classify the text enclosed in square brackets into one of
the following coarse class labels: 'ABBR' (abbreviation), 'ENTY' (entity),
'DESC' (description or abstract concept), 'HUM' (human being), 'LOC'
(location), or 'NUM' (numeric value). Return the answer as the full
corresponding class label, exactly as it appears in this list, without any
abbreviations.[/INST]

        [{data_point["text"]}]= {data_point["coarse_label"]}
    """.strip()

```

مدل‌های LLM مانند Mistral، GPT و غیره، بر اساس پردازش توالی‌های متنی tarin شده‌اند. این مدل‌ها به صورت طبیعی به یک ورودی متنی پاسخ می‌دهند و وظایف مختلفی را با درک و تولید زبان انجام می‌دهند. در نتیجه، برای اینکه یک مدل LLM بتواند از مجموعه داده‌ای مانند TREC استفاده کند، داده‌ها باید به قالب prompt تبدیل شوند تا مدل بتواند با این داده‌ها تعامل کند.

هدف از تبدیل مجموعه داده TREC به قالب پرامت این است که مدل LLM بتواند تسکی مثل classification سوالات را به صورت طبیعی انجام دهد، به این صورت که به جای دریافت داده‌های ساختاریافته به شکل جدول، داده‌ها را به صورت یک پرامت درک کند و سپس پاسخ دهد.

نحوه کارکرد پرامت:

ورودی: مدل LLM یک پرامت را دریافت می‌کند که در آن از او خواسته شده تا یک متن خاص را به یکی از دسته‌های مشخص شده تخصیص دهد.

پردازش: مدل بر اساس دانشی که از قبل دارد و اطلاعاتی که از متن دریافت می‌کند، تصمیم می‌گیرد که متن متعلق به کدام دسته هست.

خروجی: مدل یک پاسخ تولید می‌کند که باید مطابق با labelهای داده شده در دستورالعمل باشد.

```
def generate_prompt_test(data_point):  
    return f"""  
        [INST]Classify the text enclosed in square brackets into one of  
the following coarse class labels: 'ABBR' (abbreviation), 'ENTY' (entity),  
'DESC' (description or abstract concept), 'HUM' (human being), 'LOC'  
(location), or 'NUM' (numeric value). Return the answer as the full  
corresponding class label, exactly as it appears in this list, without any  
abbreviations.[/INST]  
  
        [{data_point["text"]}]= """.strip()
```

مشابه قبل هستش، تفاوت در این است که در پرامت تست ست نام label ذکر نمیشود و قسمت
حذف شده است. {"data_point["coarse_label"]}

```
y_true = df_test[["coarse_label"]]  
y_true
```

در ادامه، قبل از تبدیل کردن داده تست ست به فرمت پرامت، قبل از حذف کردن لیبل‌های سوالات را که پاسخ درست دارند را ذخیره می‌کنیم.

```
df_train = pd.DataFrame(df_train.apply(generate_prompt, axis=1),  
                        columns=["text"])  
df_valid = pd.DataFrame(df_valid.apply(generate_prompt, axis=1),  
                        columns=["text"])  
  
df_test = pd.DataFrame(df_test.apply(generate_prompt_test, axis=1),  
                       columns=["text"])
```

سپ توابع پرامت نویسی را برای همه رکوردها اعمال می‌کنیم که برخی از نمونه‌های آن به شکل زیر می‌باشد:
داده train:

545 [INST]Classify the text enclosed in square brackets into one of the following coarse class labels: 'ABBR' (abbreviation), 'ENTY' (entity), 'DESC' (description or abstract concept), 'HUM' (human being), 'LOC' (location), or 'NUM' (numeric value). Return the answer as the full corresponding class label, exactly as it appears in this list, without any abbreviations. [/INST]\n [Who was credited with saying : `` I never met a man I did n't like " ?] = HUM

546 [INST]Classify the text enclosed in square brackets into one of the following coarse class labels: 'ABBR' (abbreviation), 'ENTY' (entity), 'DESC' (description or abstract concept), 'HUM' (human being), 'LOC' (location), or 'NUM' (numeric value). Return the answer as the full corresponding class label, exactly as it appears in this list, without any abbreviations. [/INST]\n [Whose special bear 's creator was born on January 18 , 1779 ?] = HUM

547 [INST]Classify the text enclosed in square brackets into one of the following coarse class labels: 'ABBR' (abbreviation), 'ENTY' (entity), 'DESC' (description or abstract concept), 'HUM' (human being), 'LOC' (location), or 'NUM' (numeric value). Return the answer as the full corresponding class label, exactly as it appears in this list, without any abbreviations. [/INST]\n [Who commanded the French forces at the Battle of Orleans ?] = HUM

548 [INST]Classify the text enclosed in square brackets into one of the following coarse class labels: 'ABBR' (abbreviation), 'ENTY' (entity), 'DESC' (description or abstract concept), 'HUM' (human being), 'LOC' (location), or 'NUM' (numeric value). Return the answer as the full corresponding class label, exactly as it appears in this list, without any abbreviations. [/INST]\n [What was Mark Johnson referring to when he said : `` I still can 't believe it- we beat the Russians ? "] = ENTY

549 [INST]Classify the text enclosed in square brackets into one of the following coarse class labels: 'ABBR' (abbreviation), 'ENTY' (entity), 'DESC' (description or abstract concept), 'HUM' (human being), 'LOC' (location), or 'NUM' (numeric value). Return the answer as the full corresponding class label, exactly as it appears in this list, without any abbreviations. [/INST]\n [What is the abbreviation for Original Equipment Manufacturer ?] = ABBR

... ..

5447 [INST]Classify the text enclosed in square brackets into one of the following coarse class labels: 'ABBR' (abbreviation), 'ENTY' (entity), 'DESC' (description or abstract concept), 'HUM' (human being), 'LOC' (location), or 'NUM' (numeric value). Return the answer as the full corresponding class label, exactly as it appears in this list, without any abbreviations. [/INST]\n [How much Coca Cola is drunk in one day in the world ?] = NUM

5448 [INST]Classify the text enclosed in square brackets into one of the following coarse class labels: 'ABBR' (abbreviation), 'ENTY' (entity), 'DESC' (description or abstract concept), 'HUM' (human being), 'LOC' (location), or 'NUM' (numeric value). Return the answer as the full corresponding class label, exactly as it appears in this list, without any abbreviations. [/INST]\n [What cathedral was Thomas Becket murdered in ?] = LOC

5449 [INST]Classify the text enclosed in square brackets into one of the following coarse class labels: 'ABBR' (abbreviation), 'ENTY' (entity), 'DESC' (description or abstract concept), 'HUM' (human being), 'LOC' (location), or 'NUM' (numeric value). Return the answer as the full corresponding class label, exactly as it appears in this list, without any abbreviations. [/INST]\n [What character in The Beverly Hillbillies has the given names Daisy Moses ?] = HUM

5450 [INST]Classify the text enclosed in square brackets into one of the following coarse class labels: 'ABBR' (abbreviation), 'ENTY' (entity), 'DESC' (description or abstract concept), 'HUM' (human being), 'LOC' (location), or 'NUM' (numeric value). Return the answer as the full corresponding class label, exactly as it appears in this list, without any abbreviations. [/INST]\n [What does the River Seine empty into ?] = LOC

5451 [INST]Classify the text enclosed in square brackets into one of the following coarse class labels: 'ABBR' (abbreviation), 'ENTY' (entity), 'DESC' (description or abstract concept), 'HUM' (human being), 'LOC' (location), or 'NUM' (numeric value). Return the answer as the full corresponding class label, exactly as it appears in this list, without any abbreviations. [/INST]\n [What U.S. Congressman said : `` Keep the faith , baby " .] = HUM

4907 rows × 1 columns

:test داده

0 [INST]Classify the text enclosed in square brackets into one of the following coarse class labels: 'ABBR' (abbreviation), 'ENTY' (entity), 'DESC' (description or abstract concept), 'HUM' (human being), 'LOC' (location), or 'NUM' (numeric value). Return the answer as the full corresponding class label, exactly as it appears in this list, without any abbreviations. [/INST]\n\n [How far is it from Denver to Aspen ?] =

1 [INST]Classify the text enclosed in square brackets into one of the following coarse class labels: 'ABBR' (abbreviation), 'ENTY' (entity), 'DESC' (description or abstract concept), 'HUM' (human being), 'LOC' (location), or 'NUM' (numeric value). Return the answer as the full corresponding class label, exactly as it appears in this list, without any abbreviations. [/INST]\n\n [What county is Modesto , California in ?] =

2 [INST]Classify the text enclosed in square brackets into one of the following coarse class labels: 'ABBR' (abbreviation), 'ENTY' (entity), 'DESC' (description or abstract concept), 'HUM' (human being), 'LOC' (location), or 'NUM' (numeric value). Return the answer as the full corresponding class label, exactly as it appears in this list, without any abbreviations. [/INST]\n\n [Who was Galileo ?] =

3 [INST]Classify the text enclosed in square brackets into one of the following coarse class labels: 'ABBR' (abbreviation), 'ENTY' (entity), 'DESC' (description or abstract concept), 'HUM' (human being), 'LOC' (location), or 'NUM' (numeric value). Return the answer as the full corresponding class label, exactly as it appears in this list, without any abbreviations. [/INST]\n\n [What is an atom ?] =

4 [INST]Classify the text enclosed in square brackets into one of the following coarse class labels: 'ABBR' (abbreviation), 'ENTY' (entity), 'DESC' (description or abstract concept), 'HUM' (human being), 'LOC' (location), or 'NUM' (numeric value). Return the answer as the full corresponding class label, exactly as it appears in this list, without any abbreviations. [/INST]\n\n [When did Hawaii become a state ?] =

...

495 [INST]Classify the text enclosed in square brackets into one of the following coarse class labels: 'ABBR' (abbreviation), 'ENTY' (entity), 'DESC' (description or abstract concept), 'HUM' (human being), 'LOC' (location), or 'NUM' (numeric value). Return the answer as the full corresponding class label, exactly as it appears in this list, without any abbreviations. [/INST]\n\n [Who was the 22nd President of the US ?] =

496 [INST]Classify the text enclosed in square brackets into one of the following coarse class labels: 'ABBR' (abbreviation), 'ENTY' (entity), 'DESC' (description or abstract concept), 'HUM' (human being), 'LOC' (location), or 'NUM' (numeric value). Return the answer as the full corresponding class label, exactly as it appears in this list, without any abbreviations. [/INST]\n\n [What is the money they use in Zambia ?] =

497 [INST]Classify the text enclosed in square brackets into one of the following coarse class labels: 'ABBR' (abbreviation), 'ENTY' (entity), 'DESC' (description or abstract concept), 'HUM' (human being), 'LOC' (location), or 'NUM' (numeric value). Return the answer as the full corresponding class label, exactly as it appears in this list, without any abbreviations. [/INST]\n\n [How many feet in a mile ?] =

498 [INST]Classify the text enclosed in square brackets into one of the following coarse class labels: 'ABBR' (abbreviation), 'ENTY' (entity), 'DESC' (description or abstract concept), 'HUM' (human being), 'LOC' (location), or 'NUM' (numeric value). Return the answer as the full corresponding class label, exactly as it appears in this list, without any abbreviations. [/INST]\n\n [What is the birthstone of October ?] =

499 [INST]Classify the text enclosed in square brackets into one of the following coarse class labels: 'ABBR' (abbreviation), 'ENTY' (entity), 'DESC' (description or abstract concept), 'HUM' (human being), 'LOC' (location), or 'NUM' (numeric value). Return the answer as the full corresponding class label, exactly as it appears in this list, without any abbreviations. [/INST]\n\n [What is e-coli ?] =

500 rows × 1 columns

```
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix
import numpy as np

def evaluate(y_true, y_pred):
```

```

labels = ['ABBR', 'ENTY', 'DESC', 'HUM', 'LOC', 'NUM']
mapping = {'ABBR': 0, 'ENTY': 1, 'DESC': 2, 'HUM': 3, 'LOC': 4, 'NUM':
5, 'none': 6}

def map_func(x):
    return mapping.get(x, 6)

y_true = np.vectorize(map_func)(y_true)
y_pred = np.vectorize(map_func)(y_pred)

accuracy = accuracy_score(y_true=y_true, y_pred=y_pred)
print(f'Accuracy: {accuracy:.3f}')

unique_labels = np.unique(y_true)

for label in unique_labels:
    label_indices = np.where(y_true == label)[0]
    label_y_true = y_true[label_indices]
    label_y_pred = y_pred[label_indices]
    label_accuracy = accuracy_score(label_y_true, label_y_pred)
    print(f'Accuracy for label {labels[label]}: {label_accuracy:.3f}')

class_report = classification_report(y_true=y_true, y_pred=y_pred,
target_names=labels + ['none'])
print('\nClassification Report:')
print(class_report)

conf_matrix = confusion_matrix(y_true=y_true, y_pred=y_pred,
labels=list(range(7)))
print('\nConfusion Matrix:')
print(conf_matrix)

```

کدی که ارائه شده، برای ارزیابی عملکرد مدل استفاده می‌شود. مجموعه‌ای از معیارها و گزارش‌ها را تولید می‌کند که به تحلیل عملکرد مدل کمک می‌کند. این معیارها شامل Accuracy، Classification Report و Confusion Matrix هستند.

labels: این لیست شامل شش label اصلی classification در مجموعه داده TREC است.

mapping: این دیکشنری هر label را به یک عدد تبدیل می‌کند تا برای پردازش‌های بعدی مناسب باشد.

Accuracy: معیاری است که نشان می‌دهد چه درصدی از پیش‌بینی‌های مدل درست بوده‌اند. سپس این محاسبات برای هر label به صورت جداگانه می‌فتمد.

Classification Report: این گزارش شامل معیارهای مختلفی مانند Precision، بازخوانی Recall و مقدار F1 برای هر label است. این معیارها برای ارزیابی عملکرد مدل در label classification‌های مختلف بسیار مفید هستند.

Confusion Matrix: یک جدول است که نشان می‌دهد مدل به چه تعداد نمونه را به درستی classify کرده و به چه تعداد اشتباه کرده است. این جدول کمک می‌کند تا متوجه شویم مدل در کدام دسته‌بندی‌ها بهتر عمل کرده و در کدام دسته‌ها نیاز به بهبود دارد.

```
!pip install huggingface_hub
from huggingface_hub import notebook_login
```

```
from huggingface_hub import login
token = "hf_dxNjfPgOwjxvYkpnDcGQyetyhfvcWQhfQ"

# Authenticate
login(token)
```

برای دسترسی به برخی از مدل‌های هاگینگ فیس نیاز به دسترسی می‌باشد. برای این کار توکن مورد نظر را توسط اکانت هاگینگ فیس وارد می‌کنیم و لاگین می‌شویم.

```
from transformers import AutoModelForCausalLM, AutoTokenizer
from transformers import BitsAndBytesConfig
import torch
```

```
base_model_id = "mistralai/Mistral-7B-Instruct-v0.2"
```

```
bnb_config = BitsAndBytesConfig(
    load_in_4bit=True,
    bnb_4bit_quant_type="nf4",
    bnb_4bit_compute_dtype=torch.bfloat16,
```



```
)
```

```
tokenizer.pad_token = tokenizer.eos_token
```

توکنایزر مسئول تبدیل متن به توکن‌هایی است که مدل زبانی می‌تواند با آن‌ها کار کند.

```
from tqdm import tqdm
from transformers import pipeline

def predict_coarse(X_test, model, tokenizer):
    y_pred = []
    for i in tqdm(range(len(X_test))):
        prompt = X_test.iloc[i]["text"]
        pipe = pipeline(task="text-generation",
                        model=model,
                        tokenizer=tokenizer,
                        max_new_tokens = 5,
                        temperature = 0.0,
                        )

        result = pipe(prompt, pad_token_id=pipe.tokenizer.eos_token_id)
        answer = result[0]['generated_text'].split("=")[-1]
        print(result)
        print('ans', answer)
        if "ABBR" in answer:
            y_pred.append("ABBR")
        elif "ENTY" in answer:
            y_pred.append("ENTY")
        elif "DESC" in answer:
            y_pred.append("DESC")
        elif "HUM" in answer:
            y_pred.append("HUM")
        elif "LOC" in answer:
            y_pred.append("LOC")
        elif "NUM" in answer:
            y_pred.append("NUM")
        else:
            y_pred.append("none")
    return y_pred
```


یک تابع را تعریف می‌کند که با استفاده از یک LLM، متون را به دسته‌بندی‌های کلی (ABBR، ENTY، DESC، HUM، LOC، NUM) دسته‌بندی می‌کند. این کار با تولید متن توسط مدل انجام می‌شود و سپس از خروجی مدل برای تعیین classification استفاده می‌شود.

result: خروجی تولید شده توسط مدل که به‌عنوان یک لیست از دیکشنری‌ها بازگردانده می‌شود. سپس بررسی می‌کند که کدام دسته‌بندی در پاسخ تولید شده وجود دارد و آن را به لیست y_pred اضافه می‌کند. اگر هیچ کدام از دسته‌بندی‌های مشخص‌شده یافت نشود، مقدار none به لیست اضافه می‌شود. نمونه خروجی از مدل میسترال اولیه و خروجی‌های تولید شده توسط آن:

```
y_pred = predict_coarse(df_test, model, tokenizer)
y_pred
```

```
0%|          | 0/500 [00:00<?,
?it/s]/usr/local/lib/python3.10/dist-packages/transformers/generation/configuration_utils.py:540:
UserWarning: `do_sample` is set to `False`. However, `temperature` is set to `0.0` – this flag is only used in
sample-based generation modes. You should set `do_sample=True` or unset `temperature`.
warnings.warn(
0%|          | 1/500 [00:03<25:09, 3.03s/it][{'generated_text': '[INST]Classify the text enclosed in square
brackets into one of the following coarse class labels: \'ABBR\' (abbreviation), \'ENTY\' (entity), \'DESC\'
(description or abstract concept), \'HUM\' (human being), \'LOC\' (location), or \'NUM\' (numeric value).
Return the answer as the full corresponding class label, exactly as it appears in this list, without any
abbreviations. [/INST]\n\n      [How far is it from Denver to Aspen ?] = "NUM"\n      '}]
ans "NUM"
```

```
0%|          | 2/500 [00:04<16:23, 1.97s/it][{'generated_text': "[INST]Classify the text enclosed in square
brackets into one of the following coarse class labels: 'ABBR' (abbreviation), 'ENTY' (entity), 'DESC'
(description or abstract concept), 'HUM' (human being), 'LOC' (location), or 'NUM' (numeric value). Return
the answer as the full corresponding class label, exactly as it appears in this list, without any
abbreviations. [/INST]\n\n      [What county is Modesto , California in ?] = LOC\n      ['}]
ans LOC
```

```
[
1%|          | 3/500 [00:05<13:37, 1.64s/it][{'generated_text': '[INST]Classify the text enclosed in square
brackets into one of the following coarse class labels: \'ABBR\' (abbreviation), \'ENTY\' (entity), \'DESC\'
(description or abstract concept), \'HUM\' (human being), \'LOC\' (location), or \'NUM\' (numeric value).
Return the answer as the full corresponding class label, exactly as it appears in this list, without any
abbreviations. [/INST]\n\n      [Who was Galileo ?] = "HUM: Human"}]
ans "HUM: Human"
```

```
1%|          | 4/500 [00:06<12:20, 1.49s/it][{'generated_text': "[INST]Classify the text enclosed in square
brackets into one of the following coarse class labels: 'ABBR' (abbreviation), 'ENTY' (entity), 'DESC'
```

(description or abstract concept), 'HUM' (human being), 'LOC' (location), or 'NUM' (numeric value). Return the answer as the full corresponding class label, exactly as it appears in this list, without any abbreviations. [/INST]\n\n [What is an atom ?] = DESC (description or")]

ans DESC (description or

1%| | 5/500 [00:08<11:33, 1.40s/it][{'generated_text': '[INST]Classify the text enclosed in square brackets into one of the following coarse class labels: \'ABBR\' (abbreviation), \'ENTY\' (entity), \'DESC\' (description or abstract concept), \'HUM\' (human being), \'LOC\' (location), or \'NUM\' (numeric value). Return the answer as the full corresponding class label, exactly as it appears in this list, without any abbreviations. [/INST]\n\n [When did Hawaii become a state ?] = "NUM, LOC"]}

ans "NUM, LOC

1%| | 6/500 [00:09<11:05, 1.35s/it][{'generated_text': '[INST]Classify the text enclosed in square brackets into one of the following coarse class labels: \'ABBR\' (abbreviation), \'ENTY\' (entity), \'DESC\' (description or abstract concept), \'HUM\' (human being), \'LOC\' (location), or \'NUM\' (numeric value). Return the answer as the full corresponding class label, exactly as it appears in this list, without any abbreviations. [/INST]\n\n [How tall is the Sears Building ?] = "NUM"\n }]

ans "NUM"

1%| | 7/500 [00:10<10:49, 1.32s/it][{'generated_text': "[INST]Classify the text enclosed in square brackets into one of the following coarse class labels: 'ABBR' (abbreviation), 'ENTY' (entity), 'DESC' (description or abstract concept), 'HUM' (human being), 'LOC' (location), or 'NUM' (numeric value). Return the answer as the full corresponding class label, exactly as it appears in this list, without any abbreviations. [/INST]\n\n [George Bush purchased a small interest in which baseball team ?] = [Which baseball team did"]}

ans [Which baseball team did

2%| | 8/500 [00:11<10:40, 1.30s/it][{'generated_text': '[INST]Classify the text enclosed in square brackets into one of the following coarse class labels: \'ABBR\' (abbreviation), \'ENTY\' (entity), \'DESC\' (description or abstract concept), \'HUM\' (human being), \'LOC\' (location), or \'NUM\' (numeric value). Return the answer as the full corresponding class label, exactly as it appears in this list, without any abbreviations. [/INST]\n\n [What is Australia \'s national flower ?] = ["Australia" :']}]

ans ["Australia" :

2%| | 9/500 [00:13<10:30, 1.28s/it][{'generated_text': '[INST]Classify the text enclosed in square brackets into one of the following coarse class labels: \'ABBR\' (abbreviation), \'ENTY\' (entity), \'DESC\' (description or abstract concept), \'HUM\' (human being), \'LOC\' (location), or \'NUM\' (numeric value). Return the answer as the full corresponding class label, exactly as it appears in this list, without any abbreviations. [/INST]\n\n [Why does the moon turn orange ?] = "DESC"\n }]

ans "DESC"

2%| | 10/500 [00:14<10:39, 1.31s/it][{'generated_text': '[INST]Classify the text enclosed in square brackets into one of the following coarse class labels: \'ABBR\' (abbreviation), \'ENTY\' (entity), \'DESC\' (description or abstract concept), \'HUM\' (human being), \'LOC\' (location), or \'NUM\' (numeric value). Return the answer as the full corresponding class label, exactly as it appears in this list, without any abbreviations. [/INST]\n\n [What is autism ?] = "DESC"\n }]

ans "DESC"

```
evaluate(y_true, y_pred)
```

```

Accuracy: 0.650
Accuracy for label ABBR: 0.778
Accuracy for label ENTY: 0.021
Accuracy for label DESC: 0.986
Accuracy for label HUM: 0.262
Accuracy for label LOC: 0.815
Accuracy for label NUM: 0.858

```

:Classification Report

	precision	recall	f1-score	support
ABBR	1.00	0.78	0.88	9
ENTY	1.00	0.02	0.04	94
DESC	0.76	0.99	0.86	138
HUM	0.94	0.26	0.41	65
LOC	0.81	0.81	0.81	81
NUM	0.79	0.86	0.82	113
none	0.00	0.00	0.00	0
accuracy			0.65	500
macro avg	0.76	0.53	0.55	500
weighted avg	0.85	0.65	0.63	500

:Confusion Matrix

```

[[0  0  0  0  2  0  7  ]]
[31 16  9  1 35  2  0  ]
[2  0  0  0 136 0  0  ]
[39  6  1 17  2  0  0  ]
[10  4 66  0  1  0  0  ]
[9  97  5  0  2  0  0  ]
[[0  0  0  0  0  0  0  ]

```

نتایج اولیه‌ای که از مدل خام به دست آمده است، به‌طور کلی عملکرد قابل قبولی دارند اما کاملاً مناسب نیستند.
به‌عنوان مثال:

در برخی از نمونه‌ها، مدل به درستی توانسته دسته‌بندی مناسب را تشخیص دهد، مانند سوال "Who was Galileo?" که به درستی به دسته "HUM" (انسان) تخصیص یافته.
با این حال، در برخی موارد دیگر، مدل به درستی نمی‌تواند گروه صحیح را تشخیص دهد یا چندین گروه را به‌صورت همزمان پیشنهاد می‌دهد، مانند "When did Hawaii become a state?" که مدل آن را به‌صورت "NUM, LOC" دسته‌بندی کرده که منجر به مشکل در تشخیص دقیق classification می‌شود.

نتایج ارزیابی:

دقت کلی: 0.650

دقت بر اساس لیبل‌ها:

ABBR: 0.778

ENTY: 0.021

DESC: 0.986

HUM: 0.262

LOC: 0.815

NUM: 0.858

گزارش Classification:

accuracy: مدل برای برچسب‌هایی مانند "DESC" و "NUM" عملکرد خوبی داشته است، اما برای برچسب "ENTY" بسیار ضعیف عمل کرده است.

Micro avg: دقت میانگین وزنی 0.85 نشان می‌دهد که مدل برای برخی از دسته‌ها عملکرد بهتری داشته اما در دسته‌های دیگر ضعف دارد.

Confusion Matrix:

مدل در بیشتر موارد دچار confusion بین دسته‌های "ENTY" و سایر دسته‌ها شده است، که نشان‌دهنده نیاز به بهبود در تشخیص این دسته خاص است.

این نتایج نشان می‌دهد که مدل خام به‌طور کلی عملکرد متوسطی دارد و برای بهبود آن نیاز است تا مدل را با داده‌های جدید و بهینه‌تر ترین شود تا بتوانیم دقت و کارایی آن را در تشخیص دسته‌بندی‌های مختلف افزایش دهیم.

```
def print_trainable_parameters(model):
    trainable_params = 0
    all_param = 0
    for _, param in model.named_parameters():
        all_param += param.numel()
        if param.requires_grad:
            trainable_params += param.numel()
    print(
        f"trainable params: {trainable_params} || all params: {all_param}"
        || trainable%: {100 * trainable_params / all_param}"
    )
```

```
trainable params: 85041152 || all params: 3837112320 || trainable%:
2.2162799758751914
```

تابع `print_trainable_parameters` به منظور محاسبه و نمایش تعداد پارامترهای قابل آموزش و کل پارامترهای موجود در یک مدل مورد استفاده قرار می‌گیرد. این اطلاعات برای فهمیدن این‌که چه درصدی از پارامترهای مدل در طی فرآیند آموزش تغییر می‌کنند، بسیار مفید است.

تحلیل نتایج:

تعداد پارامترهای قابل آموزش: 85,041,152

تعداد کل پارامترها: 3,837,112,320

درصد پارامترهای قابل آموزش: 2.22%

این نتایج نشان می‌دهند که فقط 2.22% از کل پارامترهای مدل در طول فرآیند آموزش به‌روزرسانی می‌شوند. این درصد پایین از پارامترهای قابل آموزش، معمولاً در مدل‌هایی مشاهده می‌شود که از روش‌هایی مانند LoRA (Low-Rank Adaptation) استفاده می‌کنند. این روش‌ها به طور عمده برای کاهش نیاز به منابع محاسباتی و حافظه، بدون افت قابل‌توجه در عملکرد مدل، به کار می‌روند.

به‌طور کلی، این روش‌ها برای بهینه‌سازی مدل‌های بزرگ به کار گرفته می‌شوند تا بتوانند با منابع محدودتر نیز به خوبی عمل کنند و دقت مناسبی را حفظ کنند.

```
from peft import LoraConfig, get_peft_model
```

```
lora_config = LoraConfig(
    r=32,
    lora_alpha=16,
    target_modules=[
        "q_proj",
        "k_proj",
        "v_proj",
        "o_proj",
        "gate_proj",
        "up_proj",
        "down_proj",
        "lm_head",
    ],
```

```

bias="none",
lora_dropout=0.1, # Conventional
task_type="CAUSAL_LM",
)

```

از کتابخانه PEFT استفاده شده است تا با استفاده از تکنیک Lora، مدل را برای یک تسک خاص فاین تیون کنیم.

:LoraConfig

این کلاس LoRA configuration را برای مدل تعیین می‌کند.

$r=32$: این پارامتر نشان‌دهنده درجه تجزیه‌ی رتبه پایین است. مقدار 32 یعنی که پارامترهای مدل به ماتریس‌هایی با رتبه 32 تجزیه می‌شوند.

$\text{lora_alpha}=16$: این مقدار یک ضریب برای کنترل مقیاس LoRA است. معمولاً برای تنظیم قدرت تاثیر LoRA در طی فرآیند ترین استفاده می‌شود.

`target_modules`: لیستی از نام ماژول‌های مدلی که می‌خواهیم LoRA را بر روی آنها اعمال کنیم. در اینجا، ماژول‌هایی مانند `v_proj`, `k_proj`, و `q_proj` ... مشخص شده‌اند.

`bias="none"`: این پارامتر تعیین می‌کند که آیا در حین بهینه‌سازی، به مقدار `bias` نیز توجه شود یا خیر. در اینجا، هیچ `bias`ی بهینه‌سازی نمی‌شود.

$\text{lora_dropout}=0.1$: نرخ Dropout برای LoRA، که به جلوگیری از `overfitting` کمک می‌کند.

`task_type="CAUSAL_LM"`: این پارامتر نشان می‌دهد که این configuration برای یک مدل زبانی با ساختار Causal Language Model اعمال می‌شود.

این configuration به‌طور خاص برای کاهش تعداد پارامترهای قابل آموزش در مدل‌های بزرگ استفاده می‌شود، در حالی که سعی می‌کند دقت و عملکرد مدل را حفظ کند. تکنیک LoRA با استفاده از تجزیه‌ی رتبه پایین (low-rank decomposition) سعی می‌کند مدل را با پارامترهای کمتری آموزش دهد، که باعث کاهش نیاز به منابع محاسباتی و حافظه می‌شود.

```

from transformers import Trainer, TrainingArguments
training_arguments = TrainingArguments(
    output_dir="/content/drive/My Drive/LLMtrec",
    report_to="wandb",
    max_steps=200,
    per_device_train_batch_size=1,
    gradient_accumulation_steps=8, # 4
)

```

```

gradient_checkpointing=True,
optim="paged_adamw_32bit",
save_steps=25,
logging_steps=25,                # When to start reporting loss
logging_dir="./logs",
learning_rate=2e-4,
weight_decay=0.001,
fp16=True,
bf16=False,
max_grad_norm=0.3,
warmup_ratio=0.03,
group_by_length=True,
lr_scheduler_type="cosine",
evaluation_strategy="steps",
save_strategy="steps",
do_eval = True,
eval_steps = 25,
warmup_steps=2
)

```

از کتابخانه Transformers برای configuration فرآیند training استفاده می‌شود. پارامترهای مختلفی در TrainingArguments تنظیم شده‌اند تا فرآیند بهینه‌سازی شود و با تنظیمات دلخواه انجام شود.

:output_dir

مسیر دایرکتوری که در آن مدل‌های ترین شده و نتایج ذخیره می‌شوند.

:max_steps=200

حداکثر تعداد قدم‌های training. در اینجا، train پس از 200 قدم متوقف می‌شود.

:per_device_train_batch_size=1

اندازه batch size در هر دستگاه.

:gradient_accumulation_steps=8

تعداد قدم‌هایی که برای تجمیع گرادیان‌ها قبل از به‌روزرسانی وزن‌ها استفاده می‌شود. با این تنظیم، مدل هر 8

دسته داده را تجمیع می‌کند و سپس وزن‌ها را update می‌کند.

:gradient_checkpointing=True

فعال کردن gradient checkpointing برای کاهش مصرف حافظه در طول فرآیند training. این تنظیم باعث

می‌شود که فقط بخشی از گرادیان‌ها در حافظه ذخیره شوند و بخش‌های دیگر به صورت dynamic محاسبه

شوند.

:optim="paged_adamw_32bit

بهینه‌ساز AdamW با استفاده از دقت 32 بیت و paged optimization که برای مدل‌های بزرگ و منابع محاسباتی محدود مناسب است.

save_steps=25 و logging_steps=25:

تعیین می‌کند که مدل هر 25 step ذخیره شود و لاگ‌های training نیز هر 25 step ثبت شوند.

logging_dir="/logs":

مسیر دایرکتوری که لاگ‌های آموزشی در آن ذخیره می‌شوند.

learning_rate=2e-4:

نرخ یادگیری اولیه که سرعت به‌روزرسانی وزن‌ها را تعیین می‌کند.

weight_decay=0.001:

نرخ کاهش وزن‌ها برای جلوگیری از overfitting.

fp16=True و bf16=False:

استفاده از دقت نیمه‌تعداد اعشاری (fp16) برای کاهش مصرف حافظه و تسریع در آموزش.

max_grad_norm=0.3:

تنظیم حداکثر نرمال‌سازی گرادیان‌ها برای جلوگیری از انفجار گرادیان.

warmup_ratio=0.03 و warmup_steps=2:

نسبت یا تعداد قدم‌های warmup که طی آن نرخ یادگیری به تدریج از صفر به مقدار هدف می‌رسد.

group_by_length=True:

دسته‌بندی ورودی‌ها بر اساس طول آن‌ها برای بهبود کارایی پردازش.

lr_scheduler_type="cosine":

نوع زمان‌بندی نرخ یادگیری، در اینجا با استفاده از زمان‌بندی کسینوسی که نرخ یادگیری را به‌صورت نوسانی تنظیم می‌کند.

evaluation_strategy="steps" و eval_steps=25:

استراتژی ارزیابی مدل، که در اینجا به ازای هر 25 قدم آموزشی یک بار ارزیابی می‌شود.

```
from trl import SFTTrainer
trainer = SFTTrainer(
    model=model,
    train_dataset=train_data,
    eval_dataset=eval_data,
    dataset_text_field="text",
    tokenizer=tokenizer,
    args=training_arguments
)
```


از SFTTrainer از کتابخانه trl استفاده شده است تا فرآیند Fine-Tuning مدل به صورت دقیق و کنترل شده انجام گیرد. پارامترهای اصلی این کد شامل مدل، داده‌های train و eval، توکنایزر و آرگومان‌های train است که قبلاً تنظیم شده‌اند.

پارامترها:

:model=model

مدل از قبل تعریف شده (که می‌تواند مدل Mistral-7B یا هر مدل دیگر باشد) که قرار است train شود.

:train_dataset=train_data

مجموعه داده‌های train که مدل بر روی آن‌ها train خواهد شد. این داده‌ها شامل جملاتی هستند که مدل باید روی آن‌ها تمرین کند.

:eval_dataset=eval_data

مجموعه داده‌های eval که برای evaluation مدل استفاده می‌شوند. این مجموعه داده برای بررسی عملکرد مدل در طول فرآیند آموزش استفاده می‌شود.

:dataset_text_field="text"

این پارامتر نشان می‌دهد که فیلد متنی در مجموعه داده‌ها با نام "text" شناخته می‌شود. این فیلد شامل متون ورودی است که مدل باید روی آن‌ها پردازش انجام دهد.

:tokenizer=tokenizer

توکنایزر استفاده شده که قبلاً تعریف شده است و مسئول تبدیل متون به توکن‌های قابل فهم برای مدل است.

:args=training_arguments

آرگومان‌های ترینینگ که قبلاً تنظیم شده‌اند.

تحلیل:

SFTTrainer ابزاری است که فرآیند Fine-Tuning را برای مدل‌ها به صورت ساده‌تر و کارآمدتر انجام می‌دهد. این ابزار به طور خاص برای آموزش LLMs طراحی شده و شامل قابلیت‌هایی است که به بهبود دقت و کاهش مصرف منابع کمک می‌کند.

```
trainer.train()
trainer.model.save_pretrained("trained-model")
```

[200/200 58:33, Epoch 0/1]

Step	Training Loss	Validation Loss
25	0.370500	0.326205
50	0.201100	0.391002
75	0.346900	0.330014
100	0.203800	0.347726
125	0.326800	0.322726
150	0.193000	0.324200
175	0.308900	0.319099
200	0.186100	0.318724

خروجی‌های فرآیند training

Step: این ستون نشان‌دهنده قدم‌های مختلف در طی فرآیند آموزش مدل است. در اینجا 200 قدم آموزشی انجام شده است.

Training Loss: میزان خطای مدل در مجموعه داده‌های ترینینگ در هر قدم. به‌طور کلی، هر چه این مقدار کمتر باشد، مدل بهتر روی داده‌های train یادگیری کرده است.

Validation Loss: میزان خطای مدل در مجموعه داده‌های validation در هر قدم. این مقدار کمک می‌کند تا بررسی شود که آیا مدل به خوبی در حال یادگیری است یا دچار overfitting شده است.

































در ابتدای فرآیند training هر دو مقدار Training Loss و Validation Loss کاهش می‌یابند که نشان‌دهنده یادگیری مدل است.

در ادامه، نوساناتی در مقادیر وجود دارد که طبیعی است، زیرا مدل تلاش می‌کند بهینه‌ترین پارامترها را پیدا کند. در نهایت، Training Loss به مقدار 0.186100 و Validation Loss به مقدار 0.318724 رسیده‌اند، که نشان‌دهنده عملکرد نسبتاً مناسب مدل است.

نمونه خروجی فایل ذخیره شده مدل ترین شده:

Type People Modified

[Clean up space](#) [Get offer](#)

Name	Owner	Last modified	File size	
 adapter_config.json	me	Aug 11, 2024	751 bytes	    
 adapter_model.safetensors	me	Aug 11, 2024	824.5 MB	
 optimizer.pt	me	Aug 11, 2024	649.1 MB	
 README.md	me	Aug 11, 2024	5 KB	
 rng_state.pth	me	Aug 11, 2024	14 KB	
 scheduler.pt	me	Aug 11, 2024	1 KB	    
 special_tokens_map.json	me	Aug 11, 2024	437 bytes	
 tokenizer_config.json	me	Aug 11, 2024	2 KB	
 tokenizer.json	me	Aug 11, 2024	1.7 MB	
 tokenizer.model	me	Aug 11, 2024	482 KB	
 trainer_state.json	me	Aug 11, 2024	3 KB	
 training_args.bin	me	Aug 11, 2024	5 KB	

در این فولدر، مجموعه‌ای از فایل‌های ذخیره شده‌اند که پس از هر چک پوینت ذخیره میشوند. این فایل‌ها شامل configuration ها، وزن‌های مدل ترین شده، وضعیت بهینه‌ساز و مولد اعداد تصادفی، تنظیمات توکنایزر، و سایر پارامترهای آموزشی هستند. این فایل‌ها به گونه‌ای طراحی شده‌اند که امکان ادامه دادن فرایند ترینینگ مدل از نقطه متوقف شده را فراهم کنند و همچنین به ذخیره و بازیابی تنظیمات و وزن‌های مدل برای ارزیابی یا استفاده مجدد کمک می‌کنند.

```
y_pred = predict_coarse(df_test, model, tokenizer)
evaluate(y_true, y_pred)
```

0% | 0/500 [00:00<?, ?it/s]

/usr/local/lib/python3.10/dist-packages/transformers/generation/configuration_utils.py:540:

UserWarning: `do_sample` is set to `False`. However, `temperature` is set to `0.0` -- this flag is only used in sample-based generation modes. You should set `do_sample=True` or unset `temperature`.

warnings.warn(

0%| | 1/500 [00:02<18:22, 2.21s/it]
[{'generated_text': "[INST]Classify the text enclosed in square brackets into one of the following coarse class labels: 'ABBR' (abbreviation), 'ENTY' (entity), 'DESC' (description or abstract concept), 'HUM' (human being), 'LOC' (location), or 'NUM' (numeric value). Return the answer as the full corresponding class label, exactly as it appears in this list, without any abbreviations.[/INST]\n\n[How far is it from Denver to Aspen ?] = NUM\n ["]]
ans NUM

[
0%| | 2/500 [00:04<17:05, 2.06s/it]
[{'generated_text': "[INST]Classify the text enclosed in square brackets into one of the following coarse class labels: 'ABBR' (abbreviation), 'ENTY' (entity), 'DESC' (description or abstract concept), 'HUM' (human being), 'LOC' (location), or 'NUM' (numeric value). Return the answer as the full corresponding class label, exactly as it appears in this list, without any abbreviations.[/INST]\n\n[What county is Modesto , California in ?] = LOC\n ["]]
ans LOC

[
1%| | 3/500 [00:06<16:38, 2.01s/it]
[{'generated_text': "[INST]Classify the text enclosed in square brackets into one of the following coarse class labels: 'ABBR' (abbreviation), 'ENTY' (entity), 'DESC' (description or abstract concept), 'HUM' (human being), 'LOC' (location), or 'NUM' (numeric value). Return the answer as the full corresponding class label, exactly as it appears in this list, without any abbreviations.[/INST]\n\n[Who was Galileo ?] = HUM\n ["]]
ans HUM

[
1%| | 4/500 [00:08<16:25, 1.99s/it]
[{'generated_text': "[INST]Classify the text enclosed in square brackets into one of the following coarse class labels: 'ABBR' (abbreviation), 'ENTY' (entity), 'DESC' (description or abstract concept), 'HUM' (human being), 'LOC' (location), or 'NUM' (numeric value). Return the answer as the full corresponding class label, exactly as it appears in this list, without any abbreviations.[/INST]\n\n[What is an atom ?] = DESC\n ["]]
ans DESC

[
1%| | 5/500 [00:10<17:18, 2.10s/it]
[{'generated_text': "[INST]Classify the text enclosed in square brackets into one of the following coarse class labels: 'ABBR' (abbreviation), 'ENTY' (entity), 'DESC' (description or abstract concept), 'HUM' (human being), 'LOC' (location), or 'NUM' (numeric value). Return the answer as the full corresponding class label, exactly as it appears in this list, without any abbreviations.[/INST]\n\n[When did Hawaii become a state ?] = NUM\n ["]]
ans NUM

[
1%| | 6/500 [00:12<18:20, 2.23s/it]
[{'generated_text': "[INST]Classify the text enclosed in square brackets into one of the following coarse class labels: 'ABBR' (abbreviation), 'ENTY' (entity), 'DESC' (description or abstract concept), 'HUM' (human being), 'LOC' (location), or 'NUM' (numeric value). Return the answer as the full

corresponding class label, exactly as it appears in this list, without any abbreviations. [/INST]\n\n

[How tall is the Sears Building ?] = NUM\n [""]

ans NUM

[

1%|| | 7/500 [00:14<17:34, 2.14s/it]

[{'generated_text': "[INST]Classify the text enclosed in square brackets into one of the following coarse class labels: 'ABBR' (abbreviation), 'ENTY' (entity), 'DESC' (description or abstract concept), 'HUM' (human being), 'LOC' (location), or 'NUM' (numeric value). Return the answer as the full corresponding class label, exactly as it appears in this list, without any abbreviations. [/INST]\n\n

[George Bush purchased a small interest in which baseball team ?] = HUM\n [""]

ans HUM

[

2%|| | 8/500 [00:16<17:18, 2.11s/it]

[{'generated_text': "[INST]Classify the text enclosed in square brackets into one of the following coarse class labels: 'ABBR' (abbreviation), 'ENTY' (entity), 'DESC' (description or abstract concept), 'HUM' (human being), 'LOC' (location), or 'NUM' (numeric value). Return the answer as the full corresponding class label, exactly as it appears in this list, without any abbreviations. [/INST]\n\n

[What is Australia 's national flower ?] = ENTY\n [""]

ans ENTY

[

2%|| | 9/500 [00:18<16:58, 2.07s/it]

[{'generated_text': "[INST]Classify the text enclosed in square brackets into one of the following coarse class labels: 'ABBR' (abbreviation), 'ENTY' (entity), 'DESC' (description or abstract concept), 'HUM' (human being), 'LOC' (location), or 'NUM' (numeric value). Return the answer as the full corresponding class label, exactly as it appears in this list, without any abbreviations. [/INST]\n\n

[Why does the moon turn orange ?] = DESC\n [""]

ans DESC

[

2%|| | 10/500 [00:20<16:46, 2.05s/it]

[{'generated_text': "[INST]Classify the text enclosed in square brackets into one of the following coarse class labels: 'ABBR' (abbreviation), 'ENTY' (entity), 'DESC' (description or abstract concept), 'HUM' (human being), 'LOC' (location), or 'NUM' (numeric value). Return the answer as the full corresponding class label, exactly as it appears in this list, without any abbreviations. [/INST]\n\n

[What is autism ?] = DESC\n [""]

ans DESC

این بار تست ست را روی تابع predict اجرا میکنیم. و خروجی‌های جدید را مشاهده میکنیم که با تقریب خوبی متفاوت عمل کرده است و ساختار خروجی نیز اصلاح شده است. سپس evaluation را روی خروجی‌های جدید امتحان می‌کنیم.

```

from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix
import numpy as np

def evaluate2(y_true, y_pred):
    labels = ['ABBR', 'ENTY', 'DESC', 'HUM', 'LOC', 'NUM']
    mapping = {'ABBR': 0, 'ENTY': 1, 'DESC': 2, 'HUM': 3, 'LOC': 4, 'NUM':
5}

    def map_func(x):
        return mapping.get(x, 6)

    y_true = np.vectorize(map_func)(y_true)
    y_pred = np.vectorize(map_func)(y_pred)

    accuracy = accuracy_score(y_true=y_true, y_pred=y_pred)
    print(f'Accuracy: {accuracy:.3f}')

    unique_labels = np.unique(y_true)

    for label in unique_labels:
        label_indices = np.where(y_true == label)[0]
        label_y_true = y_true[label_indices]
        label_y_pred = y_pred[label_indices]
        label_accuracy = accuracy_score(label_y_true, label_y_pred)
        print(f'Accuracy for label {labels[label]}: {label_accuracy:.3f}')

    class_report = classification_report(y_true=y_true, y_pred=y_pred,
target_names=labels)
    print('\nClassification Report:')
    print(class_report)

    conf_matrix = confusion_matrix(y_true=y_true, y_pred=y_pred,
labels=list(range(7)))
    print('\nConfusion Matrix:')
    print(conf_matrix)

```

Accuracy: 0.960
Accuracy for label ABBR: 0.889
Accuracy for label ENTY: 0.883
Accuracy for label DESC: 0.978
Accuracy for label HUM: 0.954
Accuracy for label LOC: 0.988
Accuracy for label NUM: 0.991

Classification Report:

	precision	recall	f1-score	support
ABBR	0.89	0.89	0.89	9
ENTY	0.93	0.88	0.91	94
DESC	0.96	0.98	0.97	138
HUM	0.97	0.95	0.96	65
LOC	0.96	0.99	0.98	81
NUM	0.97	0.99	0.98	113
accuracy			0.96	500
macro avg	0.95	0.95	0.95	500
weighted avg	0.96	0.96	0.96	500

Confusion Matrix:

```
[[ 8  0  1  0  0  0  0]
 [ 0 83  4  2  2  3  0]
 [ 1  2 135  0  0  0  0]
 [ 0  2  0 62  1  0  0]
 [ 0  1  0  0 80  0  0]
 [ 0  1  0  0  0 112  0]
 [ 0  0  0  0  0  0  0]]
```

تجزیه و تحلیل نتایج پیش‌بینی

در اینجا نتایج پیش‌بینی مدل و ارزیابی آن بر اساس داده‌های تست آمده است.

"NUM": برای سوالاتی مانند "How far is it from Denver to Aspen" و "When did Hawaii become a state"

"LOC": برای سوالاتی مانند "What county is Modesto, California in"

"HUM": برای سوالاتی مانند "Who was Galileo" و "George Bush purchased a small interest in which baseball team"

"DESC": برای سوالاتی مانند "What is an atom" و "Why does the moon turn orange"

به طور کلی، مدل توانسته است پیش‌بینی‌های دقیقی برای اکثر دسته‌های پرسش‌ها ارائه دهد.

دقت مدل

Accuracy: دقت کلی مدل برابر با 0.960 است، به این معنی که مدل به طور کلی 96% از نمونه‌ها را به درستی دسته‌بندی کرده است که پیشرفت زیادی نسبت به مدل قبل fine tune شدن دارد.

دقت برای هر label

ABBR: دقت برابر با 0.889 است.

ENTY: دقت برابر با 0.883 است.

DESC: دقت برابر با 0.978 است.

HUM: دقت برابر با 0.954 است.

LOC: دقت برابر با 0.988 است.

NUM: دقت برابر با 0.991 است.

گزارش Classification Report

Precision: نشان‌دهنده دقت مدل در پیش‌بینی صحیح لیبل‌ها است.

Recall: نشان‌دهنده توانایی مدل در شناسایی تمامی نمونه‌های واقعی از هر دسته است.

F1-score: میانگین هارمونیک recall and precision، که به طور کلی عملکرد مدل را توصیف می‌کند.

برای هر برچسب، مدل عملکرد بسیار خوبی را نشان داده است، با دقت بالا و نمرات f1 خوب.

Confusion Matrix

Confusion Matrix به تفکیک تعداد پیش‌بینی‌های صحیح و نادرست برای هر label نشان می‌دهد:

ABBR: مدل به طور صحیح 8 نمونه را شناسایی کرده و 1 نمونه را به اشتباه در category دیگر قرار داده است.

ENTY: مدل به طور صحیح 83 نمونه را شناسایی کرده و تعداد کمی نمونه را به اشتباه دسته‌بندی کرده است.

DESC: مدل به طور صحیح 135 نمونه را شناسایی کرده و تنها 2 نمونه را به اشتباه دسته‌بندی کرده است.

HUM: مدل به طور صحیح 62 نمونه را شناسایی کرده و 2 نمونه را به اشتباه در category دیگر قرار داده است.

LOC: مدل به طور صحیح 80 نمونه را شناسایی کرده و تنها 1 نمونه را به اشتباه دسته‌بندی کرده است.

NUM: مدل به طور صحیح 112 نمونه را شناسایی کرده و تنها 1 نمونه را به اشتباه در category دیگر قرار داده است.

نتیجه‌گیری

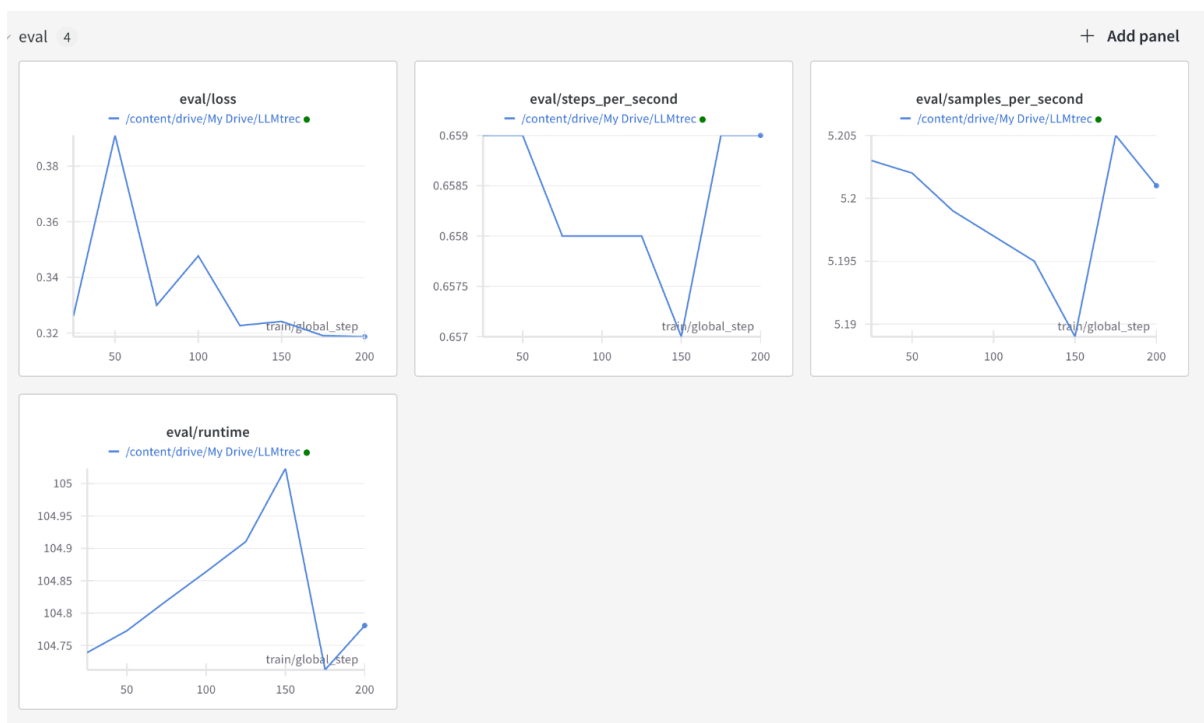
مدل در ارزیابی‌های انجام شده عملکرد بسیار خوبی داشته و دقت کلی و دقت در شناسایی هر دسته به‌طور کلی بسیار بالا است. با توجه به نتایج، مدل به خوبی توانسته classification را انجام دهد و به دقت بالایی در شناسایی لیبل‌ها دست یافته است.

```
!pip install -q wandb -U

import wandb, os
wandb.login()

wandb_project = "lora-trecdataset-llm-final"
if len(wandb_project) > 0:
    os.environ["WANDB_PROJECT"] = wandb_project
```

گزارشات wandb:



: eval/loss

این نمودار eval loss را در طی چند مرحله نشان می‌دهد. ابتدا loss به طور قابل توجهی افزایش می‌یابد و به اوج می‌رسد، سپس به سرعت کاهش می‌یابد و پایدار می‌شود، اما در نهایت دوباره کمی افزایش پیدا می‌کند.

loss در حال نوسان است، اما پس از اوج‌گیری اولیه یک روند کاهشی دارد. این نشان می‌دهد که مدل در حال یادگیری است.

:eval/steps_per_second

این نمودار تعداد قدم‌های پردازش شده در هر ثانیه را نشان می‌دهد. در حدود قدم ۵۰ کاهش پیدا می‌کند و بعد از قدم ۱۰۰ افزایش یافته و بعد از قدم ۱۵۰ دوباره ثابت می‌شود. این تغییرات ممکن است ناشی از نوسانات بار محاسباتی یا عملکرد سیستم در حین ارزیابی باشد.

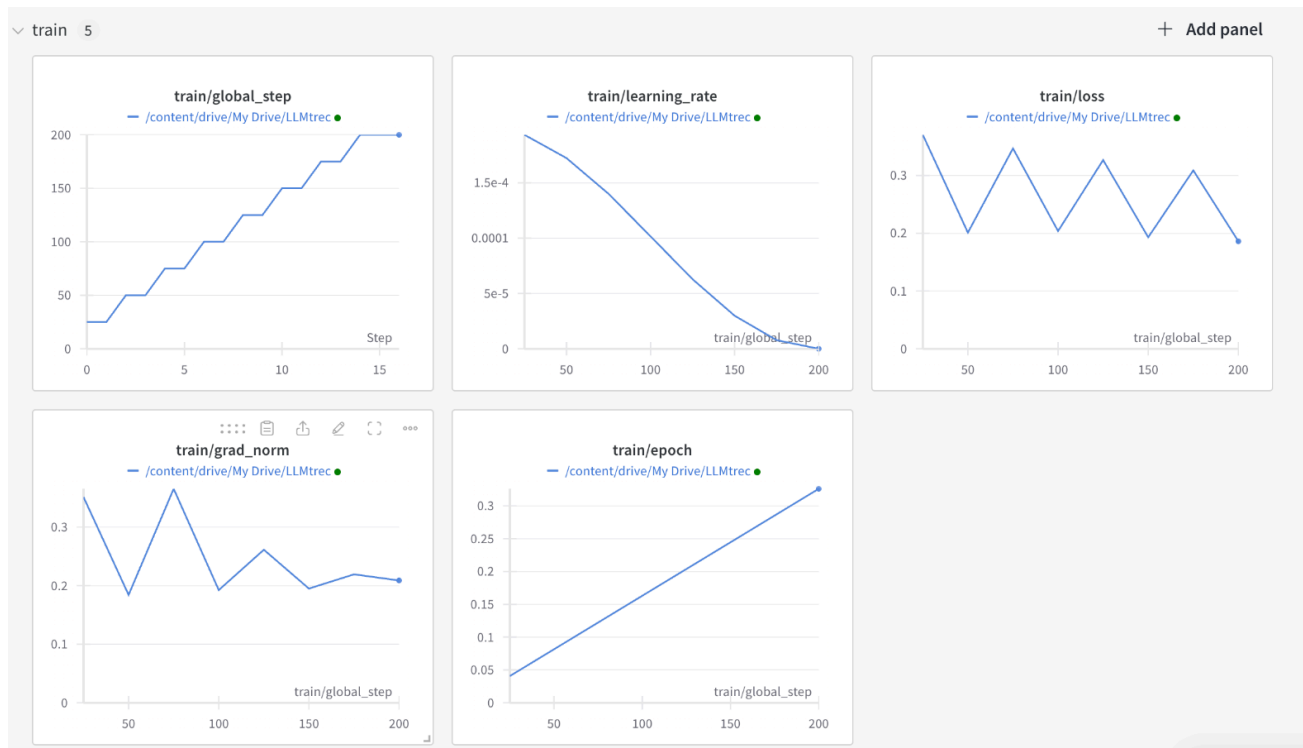
:eval/samples_per_second

این نمودار تعداد نمونه‌هایی که در هر ثانیه ارزیابی شده‌اند را نشان می‌دهد. الگوی این نمودار مشابه نمودار steps است.

در مراحل ۵۰ تا ۱۵۰ افت می‌کند و سپس دوباره افزایش می‌یابد.

:eval/runtime

این نمودار زمان اجرای فرآیند evaluation را در طی قدم‌های مختلف نشان می‌دهد. زمان اجرا به طور پیوسته تا حدود قدم ۱۵۰ افزایش می‌یابد و سپس کاهش می‌یابد.



:train/global_step

این نمودار افزایش قدم‌های کلی در طی training را نشان می‌دهد که به طور خطی افزایش می‌یابد. روند: افزایش پایدار، که نشان می‌دهد training به طور منظم پیش می‌رود.

:train/learning_rate

نرخ یادگیری به مرور زمان و با پیشرفت مراحل کاهش می‌یابد. کاهش نرخ یادگیری به صورت برنامه نزولی.

:train/loss

loss در طی فرآیند training نوسان دارد اما در کل یک روند نزولی خفیف را نشان می‌دهد. loss به طور دوره‌ای نوسان دارد که ممکن است نشانه‌ای از overfitting در برخی مراحل باشد یا اینکه مدل هنوز در حال یادگیری است.

:train/grad_norm

این نمودار نشان‌دهنده نُرم گرادیان است که میزان شیب نزول در حین train را اندازه‌گیری می‌کند. گرادیان‌ها نوسان دارند اما بزرگ نیستند، که نشانه خوبی است. نوسانات وجود دارد که انتظار می‌رود، اما ایده‌آل این است که در نهایت این نمودار صاف شود تا نشان دهد training پایدار شده است.

:train/epoch

این نمودار نشان‌دهنده پیشرفت دوره‌های training است که به طور منظم افزایش می‌یابد. روندی ثابت که نشان می‌دهد دوره‌های training به طور درستی پیش می‌روند.