به نام خدا

# تمرین سری هفتم درس مبانی پردازش زبان و گفتار دکتر مرضیه داودآبادی فراهانی

# فرزان رحمانی 99521271

## بهترین prompt

برای پیدا کردن بهترین prompt روش های گوناگونی رو تست کردیم. سپس به این نتیجه رسیدیم که برای اینکه نتایج بهتری کسب کنیم باید توازن را بین سادگی و پیچیدگی prompt حفظ کنیم. اگر مدلی به بزرگی GPT3.5 داشتیم پیچیده تر کردن prompt و افزودن جزییات بیشتر احتمالا به افزایش دقت کمک میکرد ولی در این تمرین چون هر سه مدل کوچکتر مساوی از 13B پارامتر دارند و همچنین ما مدل ها رو به صورت کوانتیزه و سبک تر (load\_in\_4bit) لود کردیم تا در حافظه free GPU کولب جا شوند افزودن جزییات بیشتر و پیچیدگی به prompt به افزایش دقت منجر نمی شد. بنابراین پس از تست کردن prompt های گوناگون به prompt های زیر برای هر سه سناریو رسیدیم. همچنین سعی کردیم که یک prompt داشته باشیم که برای ارزیابی مدل ها با هم شرایط یکسان باشد و عدالت رعایت شود.

prompt حالت Zero-shot

f"در ادامه دو سوال آمده است. " + \

"آیا این دو سوال مترادف هستند یا خیر؟ " + \

"اگر مترادف هستند فقط در یک کلمه جواب 'بله' را تولید کنید. " + \

"اگر مترادف نیستند فقط در یک کلمه جواب 'خیر' را تولید کنید. " + \

f"سوال اول: '{q1}' " + \

f"سوال دوم: '{q2}'"

prompt حالت One-shot

"در ادامه دو سوال آمده است. " + \

"آیا این دو سوال مترادف هستند یا خیر؟ " + \

"اگر مترادف هستند فقط در یک کلمه 'بله' را تولید کنید. " + \

"اگر مترادف نیستند فقط در یک کلمه 'خیر' را تولید کنید. " + \

"مثال: " + \

f"سوال اول: '{examples.iloc[0]['q1']}' " + \

f"سوال دوم: '{examples.iloc[0]['q2']}' " + \

f"جواب: '{label\_mapping[int(examples.iloc[0]['label'])]}' " + \

"حال با دیدن این مثال جواب دهید. " + \

f"سوال اول: '{q1}' " + \

f"سوال دوم: '{q2}'"

prompt حالت Five-shot

"در ادامه دو سوال آمده است. " + \

"آیا این دو سوال مترادف هستند یا خیر؟ " + \

"اگر مترادف هستند فقط در یک کلمه جواب 'بله' را تولید کنید. " + \

"اگر مترادف نیستند فقط در یک کلمه جواب 'خیر' را تولید کنید. " + \

"مثال ها: " + \

f"{examples\_text} " + \

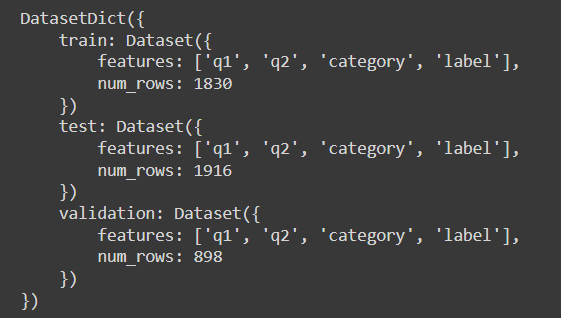
"حال با دیدن مثال ها جواب دهید. " + \

f"سوال اول: '{q1}' " + \

f"سوال دوم: '{q2}'"

## ارزیابی عملکرد هر مدل در سناریوهای مختلف آزمایش

برای تولید مثال های آموزشی one-shot و five-shot از train split دیتاست داده شده و انتخاب چندین sample به عنوان مثال آموزشی استفاده کردیم. همچنین برای ارزیابی مدل از test split دیتاست داده شده استفاده کردیم تا داده های آموزشی و ارزیابی اشتراکی نداشته باشند. همچنین چون مدل های خواسته شده بزرگ بودند و ارزیابی آنها با GPU های رایگان کولب(Tesla T4) طول می کشید از زیر مجموعه ای تصادفی از test split استفاده کردیم(برای مدل هایی که خیلی طول می کشیدند 50 نمونه برای هر سناریو و برای مدل های سریع تر 100 نمونه برای هر سناریو). از validation split نیز استفاده نکردیم چون نمیخواستیم مدل را آموزش دهیم و fine tune کنیم. همچین با استفاده از seed سمپل های یکسان برداشتیم تا قابلیت بازتولید و مقایسه عادلانه داشته باشیم.





test\_subset\_examples = test\_df.sample(n=100, random\_state=GLOBAL\_SEED) # because it takes too much to test all examples we use subset of it

one\_shot\_subset\_examples = train\_df.sample(n=1, random\_state=GLOBAL\_SEED)

# five\_shot\_subset\_examples = train\_df.sample(n=5, random\_state=GLOBAL\_SEED) # balance is not guranteed

# balance between 0 and 1

grouped = train\_df.groupby('label')

five\_shot\_subset\_examples = grouped.apply(lambda x: x.sample(n=3 if x.name == '0' else 2, random\_state=GLOBAL\_SEED)) # 2+3 = 5

five\_shot\_subset\_examples = five\_shot\_subset\_examples.reset\_index(drop=True)

همچنین توجه شود که مدل NousResearch/Meta-Llama-3-8B-Instruct یک کلمه تولید می‌کرد ولی مدل های universitytehran/PersianMind-v1.0 و ViraIntelligentDataMining/PersianLLaMA-13B بیشتر از یک کلمه با وجود اینکه در promptگفته شده بود تولید می کردند. برای اینکه بتوانیم ارزیابی داشته باشیم اولین label ای که در متن تولیدی موجود است را به عنوان label در نظر گرفتیم. کد تابع مربوطه در ادامه آمده است.

def find\_first\_label(labels, text):

    # Initialize a dictionary to store the position of each label

    positions = {label: text.find(label) for label in labels}

    # Filter out labels that are not found (position -1)

    valid\_positions = {label: pos for label, pos in positions.items() if pos != -1}

    # If no valid positions are found, return None

    if not valid\_positions:

        return None

    # Find the label with the minimum position

    first\_label = min(valid\_positions, key=valid\_positions.get)

    return first\_label

نتایج هر مدل در هر سناریو:

NousResearch/Meta-Llama-3-8B-Instruct

|  |  |  |  |
| --- | --- | --- | --- |
| Model Name | Zero-shot accuracy | One-shot accuracy | Five-shot accuracy |
| NousResearch/Meta-Llama-3-8B-Instruct | 0.66 | 0.81 | 0.39 |

universitytehran/PersianMind-v1.0

|  |  |  |  |
| --- | --- | --- | --- |
| Model Name | Zero-shot accuracy | One-shot accuracy | Five-shot accuracy |
| universitytehran/PersianMind-v1.0 | 0.17 | 0.23 | 0.27 |

ViraIntelligentDataMining/PersianLLaMA-13B

|  |  |  |  |
| --- | --- | --- | --- |
| Model Name | Zero-shot accuracy | One-shot accuracy | Five-shot accuracy |
| PersianLLaMA-13B-Instruct | 0.64 | 0.64 | 0.64 |

نتایج هر مدل در فایل های ضمیمه به شکل ‘experiment\_results\_ModelName.csv’ است. همچنین لاگ های جملات تولید شده توسط مدل، سوالات و جواب درست متناظر در فایل های ضمیمه به شکل ‘log\_df\_ModelName.csv’ آمده است.

## روند کار، پارامتر های استفاده شده، تحلیل نتایج به دست آمده و تفاوت عملکرد مدل‌ها

**روند کار:**

برای هر مدل ابتدا به Hugging face آن مدل مراجعه کردیم. هر مدل نمونه کدی برای inference گرفتن داشت. از همان استفاده نمونه کد و پارامتر های تعریف شده برای تست کردن روی یک مثال استفاده کردیم. سپس به لینک گیت هاب دیتاست خواسته شده مراجعه کردیم. از آنجا به لینک hugging face دیتاست رسیدیم. سپس دیتاست را با راهمنایی های داخل hugging face دانلود کردیم و به panda dataframe تبدیل کردیم. در ادامه تابع نوشتیم که با گرفتن نام سناریو و مثال های آموزشی prompt خواسته شده را تولید کند. سپس تابعی نوشتیم که با گرفتن prompt خروجی تمیز شده را بدهد. پس از آن با subset ای مجموعه split test و نوشتن یک for loop ساده نتایج مدل را تولید کردیم و تعداد نمونه های درست را شمردیم و بر تعداد کل تقسیم کردیم. در آخر نیز نتایج هر سناریو و مدل را در یک فایل .csv ذخیره و دانلود کردیم.

**پارامتر های استفاده شده:**

1. NousResearch/Meta-Llama-3-8B-Instruct:

template = "Answer the questions in a single word"

    messages = [

        {"role": "system", "content": template},

        {"role": "user", "content": prompt},

    ]

    prompt = pipeline.tokenizer.apply\_chat\_template(

            messages,

            tokenize=False,

            add\_generation\_prompt=True

    )

    terminators = [

        pipeline.tokenizer.eos\_token\_id,

        pipeline.tokenizer.convert\_tokens\_to\_ids("<|eot\_id|>")

    ]

    outputs = pipeline(

        prompt,

        max\_new\_tokens=max\_new\_tokens,

        eos\_token\_id=terminators,

        do\_sample=True,

        temperature=0.6,

        top\_p=0.9,

        pad\_token\_id=pipeline.tokenizer.eos\_token\_id

    )

1. universitytehran/PersianMind-v1.0:

TEMPLATE = "{context}\nYou: {prompt}\nPersianMind: "

    CONTEXT = "Answer the questions in a single word."

    PROMPT = prompt

    model\_input = TEMPLATE.format(context=CONTEXT, prompt=PROMPT)

    input\_tokens = tokenizer(model\_input, return\_tensors="pt")

    input\_tokens = input\_tokens.to(device)

    generate\_ids = model.generate(\*\*input\_tokens, max\_new\_tokens=64, do\_sample=False, repetition\_penalty=1.1, temperature=None)

    model\_output = tokenizer.batch\_decode(generate\_ids, skip\_special\_tokens=True, clean\_up\_tokenization\_spaces=False)[0]

1. ViraIntelligentDataMining/PersianLLaMA-13B:

inputs = tokenizer(prompt, return\_tensors="pt").to(device)

    outputs = model.generate(inputs["input\_ids"], max\_new\_tokens=max\_new\_tokens)

    return tokenizer.decode(outputs[0], skip\_special\_tokens=True)[len(prompt):]

**تحلیل نتایج به دست آمده و تفاوت عملکرد مدل‌ها:**

برای اینکه راحت تر بتوانیم مدل ها را با هم مقایسه کنیم نتایج را در یک نمودار به شکل بالا کشیده ایم. همان طور که مشاهده می‌کنید بهترین عملکرد میان سه مدل را در کل مدل NousResearch/Meta-Llama-3-8B-Instruct داشته است که در سناریوی one-shot learning به دقت 81% رسیده است. همچنین بدترین عملکرد را مدل universitytehran/PersianMind-v1.0 داشته است. مدل ViraIntelligentDataMining/PersianLLaMA-13B در میان سه مدل عملکردی متوسط داشته است و در سناریوی Five-shot learning حتی از مدل NousResearch/Meta-Llama-3-8B-Instruct بهتر عمل کرده است.

در سناریوی zero-shot learning مقایسه عملکرد مدل ها به شرح زیر است (از بهترین به بدترین):

1. NousResearch/Meta-Llama-3-8B-Instruct
2. ViraIntelligentDataMining/PersianLLaMA-13B
3. universitytehran/PersianMind-v1.0

در سناریوی one-shot learning مقایسه عملکرد مدل ها به شرح زیر است (از بهترین به بدترین):

1. NousResearch/Meta-Llama-3-8B-Instruct
2. ViraIntelligentDataMining/PersianLLaMA-13B
3. universitytehran/PersianMind-v1.0

در سناریوی five-shot learning مقایسه عملکرد مدل ها به شرح زیر است (از بهترین به بدترین):

1. ViraIntelligentDataMining/PersianLLaMA-13B
2. NousResearch/Meta-Llama-3-8B-Instruct
3. universitytehran/PersianMind-v1.0

همچنین می توان گفت به صورت متوسط با افزایش نمونه های یادگیری (مثلا مقایسه zero-shot learning با one-shot learning) دقت مدل ها افزایش می باید به خصوص در مدل universitytehran/PersianMind-v1.0 و همچنین در مدل NousResearch/Meta-Llama-3-8B-Instruct از zero-shot learning به one-shot learning شاهد افزایش 15 درصدی هستیم.   
در مدل ViraIntelligentDataMining/PersianLLaMA-13B دقت ثابت مانده است و شاید با تست نمونه های بیشتر مشاهده افزایش دقت بیشتر مشهود باشد. از دلایل آن میتوان استفاده از load\_in\_4bit و کوانتیزه کردن مدل باشد چرا که برای جا شدن در GPU به آن نیاز داشتیم.  
ولی در فقط یک مورد استثنا شاهد کاهش دقت از zero-shot learning به one-shot learning در مدل NousResearch/Meta-Llama-3-8B-Instruct شاهد کاهش دقت مدل هستیم. این دلیل هم میتواند ناشی از این باشد که مدل را از ریپوزیتوری اصلی Meta نتوانستیم دانلود کنیم چرا که اجازه نداشتیم و از ریپوزیتوری NousResearch دانلود کردیم. همچنین استفاده از {"torch\_dtype": torch.bfloat16, "load\_in\_4bit": True} برای لود کردن مدل می‌تواند در این رفتار غیر عادی تاثیر گذار باشد.   
  
مراجع:

<https://medium.com/@rohanvermaAI/llama-3-what-we-know-and-how-to-use-it-in-free-collab-24ec5d6058ff>   
<https://huggingface.co/docs/transformers/main/en/chat_templating#what-are-generation-prompts>   
<https://chat.openai.com/>   
<https://huggingface.co/NousResearch/Meta-Llama-3-8B-Instruct>   
[https://huggingface.co/ ViraIntelligentDataMining/ PersianLLaMA-13B](https://huggingface.co/NousResearch/Meta-Llama-3-8B-Instruct)    
<https://huggingface.co/universitytehran/PersianMind-v1.0>   
<https://github.com/persiannlp/parsinlu/tree/master/data/qqp>

پایان