به نام خدا

**دانشگاه تهران**

**دانشکده مهندسی برق و کامپیوتر**

# مدل‌های مولد عمیق

## تمرین شماره ۳

| | |
|---|---|
| نام و نام خانوادگی | علیرضا حسینی |
| شماره دانشجویی | ۸۱۰۱۰۱۱۴۲ |
| تاریخ ارسال گزارش | ۱۴۰۲/۱۱/۰۵ |

## فهرست گزارش سوالات

# سوال ۱ – LLM

## 1) مقدمه

در این سوال هدف بررسی دیتاست tweetsumm و استفاده از روش های quantization و lora جهت فاین تیون آن بر روی LLM ها میباشد.

برای انجام این بخش از پروژه از کتاب خانه های datasets ، evaluate ، transformers و ... استفاده شده است.

```
import json
import re
from pprint import pprint
import time

import pandas as pd
import torch
from datasets import Dataset, load_dataset
from huggingface_hub import notebook_login

from huggingface_hub import notebook_login
from peft import LoraConfig, PeftModel, PeftConfig
from transformers import AutoModelForCausalLM , AutoModelForSeq2SeqLM,
AutoTokenizer, GenerationConfig, TrainingArguments, Trainer , BitsAndBytesConfig

from trl import SFTTrainer

DEVICE = "cuda:1" if torch.cuda.is_available() else "cpu"
```

سخت افزار مورد استفاده برای این سوال پروژه GPU های 3090 با مشخصات زیر میباشد.

که به دلیل حافظه 48 گیگ ای آن نیازی به استفاده از bitsandbyte نشد ولی در ادامه گزارشی در مورد آن و نحوه کد زنی آن و .. امده شده است.

```
Mon Jan 22 10:27:09 2024
+---------------------------------------------------------------------------------------+
| NVIDIA-SMI 535.104.12             Driver Version: 535.104.12   CUDA Version: 12.2     |
|-----------------------------------------+----------------------+----------------------+
| GPU  Name                 Persistence-M | Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |         Memory-Usage | GPU-Util  Compute M. |
|                                         |                      |               MIG M. |
|=========================================+======================+======================|
|   0  NVIDIA GeForce RTX 3090        Off | 00000000:01:00.0 Off |                  N/A |
| 30%   52C    P0             108W / 370W |      2MiB / 24576MiB |      0%      Default |
|                                         |                      |                  N/A |
+-----------------------------------------+----------------------+----------------------+
|   1  NVIDIA GeForce RTX 3090        Off | 00000000:03:00.0 Off |                  N/A |
| 34%   39C    P0              94W / 370W |      2MiB / 24576MiB |      0%      Default |
|                                         |                      |                  N/A |
+-----------------------------------------+----------------------+----------------------+

+---------------------------------------------------------------------------------------+
| Processes:                                                                            |
|  GPU   GI   CI        PID   Type   Process name                            GPU Memory |
|        ID   ID                                                             Usage      |
|=======================================================================================|
|  No running processes found                                                           |
+---------------------------------------------------------------------------------------+
```

شکل 1 : مشخصات **GPU** های مورد استفاده برای سوال 1

## 2) لود کردن دیتاست

قبل از هر چیز لازم است تا دیتاست مدنظر را لود کرده و با ویژگی های آن آشنا شویم.

به کمک دستور زیر دیتاست را لود میکنیم.

```
dataset = load_dataset("Salesforce/dialogstudio", "TweetSumm")
```

اگر بخواهیم نگاهی به دیتاست مد نظر بیاندازیم مشخصات و سطرها و تعداد هر کدام از آن ها به صورت زیر میباشد.

```
DatasetDict({
    train: Dataset({
        features: ['original dialog id', 'new dialog id', 'dialog index',
'original dialog info', 'log', 'prompt'],
        num_rows: 879
    })
    validation: Dataset({
        features: ['original dialog id', 'new dialog id', 'dialog index',
'original dialog info', 'log', 'prompt'],
        num_rows: 110
    })
    test: Dataset({
```

```
        features: ['original dialog id', 'new dialog id', 'dialog index',
'original dialog info', 'log', 'prompt'],
        num_rows: 110
    })
})
```

حال بیاییم یکی از داده ها را باز کنیم و داخل آن را ببینیم.


```
{'original dialog id': 'b065262210783596c1fe79466b8f8985', 'new dialog id':
'TweetSumm--train--1', 'dialog index': 1, 'original dialog info': '{"summaries":
{"extractive_summaries": [[{"is_agent": false, "sentences": ["So neither my
iPhone nor my Apple Watch are recording my steps/activity, and Health
doesn\\u2019t recognise either source anymore for some reason."]}, {"is_agent":
true, "sentences": ["To start, can you tell us the software versions your iPhone
and Apple Watch are running currently?"]}, {"is_agent": false, "sentences":
["@AppleSupport My iPhone is on 11.1.2, and my watch is on 4.1."]}],
[{"is_agent": false, "sentences": ["So neither my iPhone nor my Apple Watch are
recording my steps/activity, and Health doesn\\u2019t recognise either source
anymore for some reason."]}, {"is_agent": true, "sentences": ["To start, can you
tell us the software versions your iPhone and Apple Watch are running
currently?"]}], [{"is_agent": false, "sentences": ["So neither my iPhone nor my
Apple Watch are recording my steps/activity, and Health doesn\\u2019t recognise
either source anymore for some reason."]}, {"is_agent": true, "sentences": ["Have
you tried restarting both devices since this started happening?"]}, {"is_agent":
false, "sentences": ["@AppleSupport Yes, everything seems fine, it\\u2019s just
Health and activity."]}, {"is_agent": true, "sentences": ["@135060 Let\\u2019s
move to DM and look into this a bit more."]}]], "abstractive_summaries":
[["Customer enquired about his Iphone and Apple watch which is not showing his
any steps/activity and health activities.", "Agent is asking to move to DM and
look into it."], ["The customer has a problem.", "The agent in a very
professional way tries to help the client."], ["Health and activity functions are
not working with the smartwatch and phone.", "Asks if the customer had restarted
the items, offers to take this to DM to help resolve the issue."]]}}', 'log':
[{'turn id': 1, 'user utterance': 'So neither my iPhone nor my Apple Watch are
recording my steps/activity, and Health doesn’t recognise either source anymore
for some reason. Any ideas? https://t.co/m9DPQbkftD @AppleSupport please read the
above.', 'system response': '@135060 Let’s investigate this together. To start,
can you tell us the software versions your iPhone and Apple Watch are running
currently?', 'dialog history': '', 'original user side information': '{}',
'original system side information': '{}'}, {'turn id': 2, 'user utterance':
'@AppleSupport My iPhone is on 11.1.2, and my watch is on 4.1.', 'system
response': '@135060 Thank you. Have you tried restarting both devices since this
started happening?', 'dialog history': '<USER> So neither my iPhone nor my Apple
Watch are recording my steps/activity, and Health doesn’t recognise either source
anymore for some reason. Any ideas? https://t.co/m9DPQbkftD @AppleSupport please
read the above. <SYSTEM> @135060 Let’s investigate this together. To start, can
```

```
you tell us the software versions your iPhone and Apple Watch are running
currently?', 'original user side information': '{}', 'original system side
information': '{}'}, {'turn id': 3, 'user utterance': '@AppleSupport I've
restarted both, also un-paired then re-paired the watch.', 'system response':
'@135060 Got it. When did you first notice that the two devices were not talking
to each other. Do the two devices communicate through other apps such as
Messages?', 'dialog history': '<USER> So neither my iPhone nor my Apple Watch are
recording my steps/activity, and Health doesn't recognise either source anymore
for some reason. Any ideas? https://t.co/m9DPQbkftD @AppleSupport please read the
above. <SYSTEM> @135060 Let's investigate this together. To start, can you tell
us the software versions your iPhone and Apple Watch are running currently?
<USER> @AppleSupport My iPhone is on 11.1.2, and my watch is on 4.1. <SYSTEM>
@135060 Thank you. Have you tried restarting both devices since this started
happening?', 'original user side information': '{}', 'original system side
information': '{}'}, {'turn id': 4, 'user utterance': '@AppleSupport Yes,
everything seems fine, it's just Health and activity.', 'system response':
'@135060 Let's move to DM and look into this a bit more. When reaching out in DM,
let us know when this first started happening please. For example, did it start
after an update or after installing a certain app? https://t.co/GDrqU22YpT',
'dialog history': '<USER> So neither my iPhone nor my Apple Watch are recording
my steps/activity, and Health doesn't recognise either source anymore for some
reason. Any ideas? https://t.co/m9DPQbkftD @AppleSupport please read the above.
<SYSTEM> @135060 Let's investigate this together. To start, can you tell us the
software versions your iPhone and Apple Watch are running currently? <USER>
@AppleSupport My iPhone is on 11.1.2, and my watch is on 4.1. <SYSTEM> @135060
Thank you. Have you tried restarting both devices since this started happening?
<USER> @AppleSupport I've restarted both, also un-paired then re-paired the
watch. <SYSTEM> @135060 Got it. When did you first notice that the two devices
were not talking to each other. Do the two devices communicate through other apps
such as Messages?', 'original user side information': '{}', 'original system side
information': '{}'}], 'prompt': ['']}
```


همانطور که مشاهده میشود شامل یک conversation بوده و همچنین 3 نمونه summary وجود دارد که نیاز به یک پیش پردازش میباشد تا فقط یکی از آن ها انتخاب شود.

## 3) پیش‌پردازش دیتاست

جهت پیش‌پردازش دیتاست لازم است تا مراحل زیر انجام شود.

- متن ورودی را با حذف آدرس‌های اینترنتی یا مواردی که @ هست ، نام‌های کاربر و فضاهای خالی اضافی پاک شود.

- یک پرامپ آموزشی فرمت شده ایجاد کنید که شامل اعلان سیستم، مکالمه ورودی و پاسخ مورد انتظار است.
- اجزای متن مورد نیاز برای آموزش، از جمله متن مکالمه و خلاصه آن را تولید شود.

کلاس زیر که تمامی ورودی ها و خروجی توابع و توضیحات هر کدام از آن کامنت و doc string شده است برای انجام پیش پردازش ها انجام شده است.

همچنین پرامپت مورد استفاده نیز در زیر آمده است:

```
Please provide a concise and accurate summary of the following conversation
between a human and an AI agent.
```

کد کلاس پیش پردازش :

```
# Default prompt that instructs the model to generate a summary.
DEFAULT_SYSTEM_PROMPT = """
Please provide a concise and accurate summary of the following conversation
between a human and an AI agent.
""".strip()

class DatasetProcessor:
    def __init__(self, data):
        # Initialize the processor with the dataset.
        self.data = data

    @staticmethod
    def clean_text(text):
        """
        Clean the input text by removing URLs, user mentions, and extra
whitespaces.

        Args:
            text (str): The text to be cleaned.

        Returns:
            str: The cleaned text.
        """
        text = re.sub(r"http\S+", "", text)
        text = re.sub(r"@[^\s]+", "", text)
        text = re.sub(r"\s+", " ", text)
        return re.sub(r"\^[^ ]+", "", text)

    @staticmethod
```

```
    def create_conversation_text(data_point):
        """
        Construct the conversation text from a single data point's dialog log.

        Args:
            data_point (dict): A single entry from the dataset.

        Returns:
            str: The formatted conversation text.
        """
        text = ""
        for item in data_point["log"]:
            user = DatasetProcessor.clean_text(item["user utterance"])
            text += f"user: {user.strip()}\n"

            agent = DatasetProcessor.clean_text(item["system response"])
            text += f"agent: {agent.strip()}\n"

        return text

    @staticmethod
    def generate_training_prompt(conversation: str, summary: str, system_prompt:
str = DEFAULT_SYSTEM_PROMPT):
        """
        Generate a formatted training prompt that includes the system prompt, the
input conversation, and the expected response.

        Args:
            conversation (str): The conversation text.
            summary (str): The summary of the conversation.
            system_prompt (str): Instructional text guiding the summary
generation.

        Returns:
            str: The formatted training prompt.
        """
        return f"""### Instruction: {system_prompt}

        ### Input:
        {conversation.strip()}

        """.strip()

    @staticmethod
    def generate_text(data_point):
        """
        Generate the text components required for training, including the
conversation text and its summary.
```

```
        Args:
            data_point (dict): A single entry from the dataset.

        Returns:
            dict: A dictionary containing the conversation, the summary, and the
full training text.
        """
        summaries = json.loads(data_point["original dialog
info"])["summaries"]["abstractive_summaries"]
        summary = " ".join(summaries[0])

        conversation_text = DatasetProcessor.create_conversation_text(data_point)
        return {
            "conversation": conversation_text,
            "summary": summary,
            "new_prompt":
DatasetProcessor.generate_training_prompt(conversation_text, summary),
        }

    def process(self):
        """
        Process the entire dataset by shuffling, generating text for each entry,
and removing unnecessary columns.

        Returns:
            Dataset: The processed dataset ready for use in training or
evaluation.
        """
        process_func = lambda x: DatasetProcessor.generate_text(x)
        self.data = (
            self.data.shuffle(seed=42)
            .map(process_func)
            .remove_columns(
                [
                    "original dialog id",
                    "new dialog id",
                    "dialog index",
                    "original dialog info",
                    "log",
                    "prompt",
                ]
            )
        )
        return self.data
```

در ادامه پیش پردازش را بر همه دسته های آموزش و ارزیابی و تست اعمال میکنیم.

```
# Processing the dataset
processor = DatasetProcessor(dataset["train"])
dataset["train"] = processor.process()

processor = DatasetProcessor(dataset["validation"])
dataset["validation"] = processor.process()

processor = DatasetProcessor(dataset["test"])
dataset["test"] = processor.process()
```

در نهایت فرم کلی دیتاست پس از پیش‌پردازش به صورت زیر میشود:

```
DatasetDict({
    train: Dataset({
        features: ['conversation', 'summary', 'new_prompt'],
        num_rows: 879
    })
    validation: Dataset({
        features: ['conversation', 'summary', 'new_prompt'],
        num_rows: 110
    })
    test: Dataset({
        features: ['conversation', 'summary', 'new_prompt'],
        num_rows: 110
    })
})
```

در نهایت نیز برای آنکه محتویات دیتاست را داشته باشم و این مراحل مرتب تکرار نشود تمامی داده های پیش پردازش شده را در فایل های CSV ذخیره سازی میکنیم.

```
# Create dataframes from the dataset
train_df = pd.DataFrame(dataset["train"])
validation_df = pd.DataFrame(dataset["validation"])
test_df = pd.DataFrame(dataset["test"])

# Save dataframes to CSV files
train_df.to_csv("train.csv", index=False, columns=["conversation",
"new_prompt", "summary"])
validation_df.to_csv("validation.csv", index=False,
columns=["conversation", "new_prompt", "summary"])
test_df.to_csv("test.csv", index=False, columns=["conversation",
"new_prompt", "summary"])
```

## 4) لود کردن مدل

مدل خواسته شده در این سوال stablelm-3b با بیش از 3 میلیارد پارامتر میباشد.

برای لود کردن مدل و توکنایزر از دستور زیر استفاده میکنیم. لازم به ذکر است برای جلوگیری از گرفتن حجم زیاد بر روی GPU از torch_dtype=torch.bfloat16 استفاده میشود.

```
# Load tokenizer and model

tokenizer = AutoTokenizer.from_pretrained("stabilityai/stablelm-3b-4e1t",
use_auth_token=True)
model = AutoModelForCausalLM.from_pretrained("stabilityai/stablelm-3b-4e1t",
torch_dtype=torch.bfloat16,use_auth_token=True, trust_remote_code=True)
```

## 5) in context learning

گام اول سوال از ما خواسته که بر روی مدل و با توجه به دیتا ها یک سری آزمایش ها انجام دهیم.

ابتدا نیاز به یک سری تعریف ها میباشد.

### 5-1) Zero-Shot

- Zero shot : تکنیکی است که در پردازش زبان طبیعی (NLP) و یادگیری ماشین استفاده می‌شود، به‌ویژه در مدل‌های زبان بزرگ، جایی که انتظار می‌رود مدل پرسش‌هایی را که به صراحت روی آن‌ها آموزش ندیده است، بفهمد و به آنها پاسخ دهد. برخلاف یادگیری تحت نظارت، که در آن یک مدل بر روی داده‌های برچسب‌گذاری‌شده مخصوص کار آموزش داده می‌شود، تحریک صفر به توانایی مدل برای تعمیم دانش از طیف متنوعی از منابع و زمینه‌ها به موقعیت‌های جدید متکی است. این رویکرد از درک ذاتی زبان و دانش جهانی موجود در مدل در طول مرحله قبل از آموزش آن بر روی مجموعه‌های وسیعی از متن استفاده می‌کند. مزیت اصلی دستور zero shot انعطاف پذیری و سازگاری آن است که به مدل ها اجازه می دهد تا مجموعه وسیعی از وظایف را بدون نیاز به داده های خاص کار، فاین تیون یا آموزش اضافی انجام دهند. این نه تنها کاربردهای بالقوه مدل‌های زبانی را گسترش می‌دهد، بلکه نگاهی اجمالی به درک بیشتر انسان‌ها و قابلیت‌های حل مسئله سیستم‌های هوش مصنوعی ارائه می‌دهد.

حال برای prompt های zero shot میتوانیم به صورت زیر عمل کنیم.

```
index = 100
```

```
dialogue = dataset['test'][index]['conversation']
baseline_human_summary = dataset['test'][index]['summary']

prompt = f"""
Please provide a concise and accurate summary of the following conversation
between a human and an AI agent.
{dialogue}

Summary: """

input_ids = tokenizer(prompt, return_tensors="pt")
inputs = {key: value.to("cuda:1") for key, value in input_ids.items()}

generation_config = GenerationConfig(
    max_length=2048,
    num_beams=1,
    pad_token_id=tokenizer.eos_token_id  ,
    temperature=0.7,

)

original_model_outputs = model.generate(input_ids=inputs["input_ids"],
generation_config=generation_config)
original_model_text_output = tokenizer.decode(original_model_outputs[0],
skip_special_tokens=True)


dash_line = '-'.join('' for x in range(100))
print(dash_line)
print(f'BASELINE HUMAN SUMMARY:\n{baseline_human_summary}')
print(dash_line)
print(f'ORIGINAL MODEL:\n{original_model_text_output}')
```

به عنوان نمونه داده 100 ام از مجموعه داده تست را مورد بررسی قرار میدهیم.

مکالمه و پرامپت مد نظر این داده به صورت زیر میباشد:

```
Please provide a concise and accurate summary of the following conversation
between a human and an AI agent:

user: Can you tell me how to do Red Eye Removal in Lightroom CC? I just moved to
it and don't see the Red Eye Removal tool.
agent: Hi Bob, here is a link to show you to use the Red eye removal in Lightroom
CC.
user: Does not apply to the NEW LightRoom CC. Any other suggestions?
```

```
agent: Bob, I will loop in our Lightroom expert to help you with this. The
setting may have moved to a different location.  Hi Bob, I am looping our expert
team to help answer your question. They will get back to you ASAP. Please excuse
the delay, if any. Thanks!  Hi Bob, Yes, its not there in Lightroom CC also,
refer: Thanks.
user: Thank you. I wish a list of feathers missing in Lightroom CC would have
been noted before I migrated my library. Never thought a commercial photo app
from Adobe would omit a basic feature like that. *features
agent: Hi Bob, you can report this here to alert our product teams and engineers:
Thanks!  Hi Bob, this feature is not available in Lightroom CC as of now, however
you may suggest it as a feature here:.
user: Hate to be “that guy” but this is a Photo Editing 101 feature. Where is the
list of what’s missing from the “new” Lightroom CC? Also, it would be great if
included “Lightroom CC” in its support system. Only “PhotoShop Lightroom” is
listed on that page. So if I request it, I'd probably get back an "Already
available" response.
agent: We have released Lightroom Classic CC which has all the features the old
Lightroom CC 2015.12 had, you can check this article to see the differences
betweem LR Classic &amp;amp; the new Lightroom CC:.
```

خروجی لیبل این داده به صورت زیر است :

```
Customer is asking help that  how to remove red eye in ligth room cc even he cant
find it in tool  and even customer want some new advance features. Agent is
giving details on it and then sends a link where he can get help and also asked
customer to  report a  complaint where  his engineer team will  get alert and
help him over it.
```

خروجی مدل نیز به صورت زیر میباشد :

```
The conversation was about a missing feature in Lightroom CC. The agent was
unable to provide a solution. The agent suggested the user report the missing
feature to Adobe. The user was not satisfied with the response.
```

که نشان میدهد مدل pretrained تا حد خوبی خروجی مطلوبی داده است.

### One-Shot (5-2

- One shot : یک رویکرد جذاب در زمینه پردازش زبان طبیعی (NLP) و یادگیری ماشین است، به ویژه با مدل های زبان پیشرفته. این به توانایی مدل برای درک و انجام وظایف پس از قرار گرفتن در معرض یک مثال یا نمونه واحد اشاره دارد. در این تنظیمات، مدل با یک نمونه از خروجی مورد نظر ارائه می‌شود، که اغلب در اعلان تعبیه شده است، و سپس انتظار می‌رود که الگو یا مفهوم آموخته‌شده از این مثال را برای ایجاد پاسخ‌های مناسب به ورودی‌های جدید و نادیده اعمال کند. درخواست یک‌شات تعادلی بین دستور صفر شات، که نیازی به مثال ندارد،

و یادگیری چند شات یا چند شات، که در آن از چندین مثال استفاده می‌شود، ایجاد می‌کند. نقطه قوت تلقین تک شات در توانایی آن برای انطباق سریع با وظایف یا زمینه های جدید، استفاده از دانش از قبل موجود و قابلیت های زبانی تعبیه شده در مدل در طول دوره پیش آموزش نهفته است. این روش به‌ویژه زمانی با ارزش است که با وظایف خاص یا خاص سروکار داریم که در آن داده‌های برچسب‌گذاری شده در مقیاس بزرگ ممکن است به آسانی در دسترس نباشند، و مدل را قادر می‌سازد تا پس از مشاهده تنها یک نمونه از کار در دست، با درجه‌ای از شایستگی عمل کند.

برای کد زنی این بخش یه نمونه از داده های مجموعه آموزشی را بر میداریم و به عنوان نمونه به مدل با جواب نشان میدهیم و از آن خروجی خلاصه برای داده 100 ام را میخواهیم.

```
# Select a random conversation-summary pair from the train dataset for the one-shot example
example_conversation_train = dataset['train'][0]['conversation']
example_summary_train =  dataset['train'][0]['summary']

# Select the conversation and the human-generated summary from the test dataset
index = 100
dialogue_test = dataset['test'][index]['conversation']
baseline_human_summary = dataset['test'][index]['summary']


# Prepare the one-shot prompt
prompt = f"""
### Example Conversation:
{example_conversation_train}

### Example Summary:
{example_summary_train}

---

### New Conversation:
{dialogue_test}

### Please provide a concise and accurate summary for the new conversation above:
"""

input_ids = tokenizer(prompt, return_tensors="pt")
inputs = {key: value.to("cuda:1") for key, value in input_ids.items()}

generation_config = GenerationConfig(
```

```
    max_length=1024,
    num_beams=1,
    pad_token_id=tokenizer.eos_token_id,
    temperature=0.7,
)

# Generate the summary for the new conversation
original_model_outputs = model.generate(input_ids=inputs["input_ids"],
generation_config=generation_config)
original_model_text_output = tokenizer.decode(original_model_outputs[0],
skip_special_tokens=True)

# Output the results
dash_line = '-' * 100
print(dash_line)
print(f'BASELINE HUMAN SUMMARY:\n{baseline_human_summary}')
print(dash_line)
print(f'ONE-SHOT MODEL OUTPUT:\n{original_model_text_output}')
```

پرامت در اینجا به صورت زیر میشود :

```
### Example Conversation:
user: Do you have a plan to notify passengers well in advance of pilot related
cancellations or just wait til the day before? Will you protect passengers on
other airlines if flights are cancelled b/c of pilot shortages?
agent: We're planning to fly as scheduled, Shaun.
user: HOW ABOUT ANSWERING MY QUESTION. I'm asking if you do not get enough pilots
to fly, which is a possibility, do you have a contingency plan in place on how to
get customers to their destinations &amp;amp; when will it be relayed to customers.
THE DAY BEFORE WILL NOT BE ACCEPTABLE!
agent: Our team is working hard to avoid cancellations and you'll be notified if
otherwise.
user: Your reading comprehension is terrible. WHEN WILL WE BE NOTIFIED? 3 hours
b4 our flight so all other flights r sold out? Instead of Doug Parker making
comments like I don't think we're ever going to lose money again," he should b
assuring customers we're getting home 4 XMAS
agent: As of now, flights are scheduled and we expect to avoid cancellations.
user: This is like a Seinfeld skit. I know you have flights scheduled. But what
about the pilots...are they scheduled? The answer to that?? in many cases is
probably NO. I know the AA twitter team is saying what the company tells u to
say, but AA SUCKS at communication.
agent: We’re working to address this issue. We do expect to avoid cancellations
this holiday season.
### Example Summary:
```

```
The customer is complaining that what will you do if there are no enough pilots to fly. The agent answered that as of now flights are scheduled and they have avoiding cancellations.

---

### New Conversation:
user: Can you tell me how to do Red Eye Removal in Lightroom CC? I just moved to it and don't see the Red Eye Removal tool.
agent: Hi Bob, here is a link to show you to use the Red eye removal in Lightroom CC.
user: Does not apply to the NEW LightRoom CC. Any other suggestions?
agent: Bob, I will loop in our Lightroom expert to help you with this. The setting may have moved to a different location.  Hi Bob, I am looping our expert team to help answer your question. They will get back to you ASAP. Please excuse the delay, if any. Thanks!  Hi Bob, Yes, its not there in Lightroom CC also, refer: Thanks.
user: Thank you. I wish a list of feathers missing in Lightroom CC would have been noted before I migrated my library. Never thought a commercial photo app from Adobe would omit a basic feature like that. *features
agent: Hi Bob, you can report this here to alert our product teams and engineers: Thanks!  Hi Bob, this feature is not available in Lightroom CC as of now, however you may suggest it as a feature here:.
user: Hate to be “that guy” but this is a Photo Editing 101 feature. Where is the list of what’s missing from the “new” Lightroom CC? Also, it would be great if included “Lightroom CC” in its support system. Only “PhotoShop Lightroom” is listed on that page. So if I request it, I'd probably get back an "Already available" response.
agent: We have released Lightroom Classic CC which has all the features the old Lightroom CC 2015.12 had, you can check this article to see the differences betweem LR Classic &amp;amp; the new Lightroom CC:.


### Please provide a concise and accurate summary for the new conversation above:
### Summary :
```

که خروجی مدل در این حالت به صورت زیر میباشد:

```
The customer is complaining that he is not able to find the Red Eye Removal tool in Lightroom CC. The agent answered that the feature is not available in Lightroom CC as of now.
```

که نشان میدهد که خروجی یکم بهتر شده است.

### 5-3) مقایسه zero-shot و one-shot

هر دو روش تعامل با مدل‌های زبان بزرگ (LLM) را نشان می‌دهند که هر کدام مزایا و چالش‌های منحصربه‌فردی را ارائه می‌کنند. درخواست صفر شات شامل ارائه یک سوال یا کار به LLM بدون ارائه هیچ مثال یا زمینه صریحی است. نقطه قوت این رویکرد در ارزیابی خام آن از قابلیت های از پیش آموزش دیده مدل نهفته است، و به طور موثر ارزیابی می کند که چگونه مدل درک خود را به وظایف جدید و دیده نشده تعمیم می دهد. با این حال، این می‌تواند یک محدودیت نیز باشد، زیرا فقدان راهنمایی یا زمینه خاص می‌تواند منجر به پاسخ‌های کمتر دقیق یا نادقیق شود، به ویژه در سناریوهای پیچیده‌تر یا ظریف‌تر.

در مقابل، درخواست تک شات قبل از طرح سوال یا تکلیف، یک مثال یا قسمتی از زمینه را در اختیار LLM قرار می دهد. این مثال به عنوان یک چارچوب راهنما عمل می کند و به مدل یک نقطه مرجع برای قالب خروجی مورد نظر یا نوع استدلال مورد نیاز ارائه می دهد. درخواست تک شات می تواند عملکرد مدل را در کارهای خاص به طور قابل توجهی افزایش دهد، پاسخ ها را با انتظارات انسان هماهنگ کند و دقت را در سناریوهای ظریف یا وابسته به زمینه بهبود بخشد. با این حال، کیفیت خروجی به شدت به ارتباط و وضوح مثال ارائه شده متکی است، و این خطر وجود دارد که مدل بیش از حد به ساختار یا محتوای مثال تکیه کند و به طور بالقوه خلاقیت یا سازگاری را در پاسخ‌های آن محدود کند.

### 6) عملکرد LoRA و مفهوم Efficient Fine tuning

LoRa (انطباق با رتبه پایین) یک تکنیک نوآورانه است که برای فاین تیون مدل های بزرگ از پیش آموزش دیده به طور کارآمد و موثر طراحی شده است. این تکنیک به یک چالش مهم در زمینه یادگیری عمیق می پردازد که شامل منابع محاسباتی قابل توجه و زمان مورد نیاز برای فاین تیون مدل های بزرگ مانند GPT-3 یا BERT برای کارهای خاص است. به طور سنتی، فاین تیون چنین مدل هایی نیاز به بروز رسانی تمام پارامترها دارد، فرآیندی که می تواند از نظر محاسباتی فشرده و زمان بر باشد.

LoRa یک رویکرد منحصر به فرد برای این مشکل معرفی می کند. به جای به روز رسانی تمام پارامترهای یک مدل، تجزیه با رتبه پایین را به ماتریس های وزن در لایه های ترنسفورمر مدل معرفی می کند. این روش شامل اضافه کردن دو ماتریس کوچک و قابل آموزش به هر لایه است که به‌روزرسانی‌ها را در حین فاین تیون ثبت می‌کند. ماتریس‌های وزن اصلی ثابت می‌مانند و فقط این ماتریس‌های کوچکتر به‌روزرسانی می‌شوند و تعداد پارامترهایی را که باید آموزش داده شوند به میزان قابل توجهی کاهش می‌دهند. این رویکرد نه تنها بار محاسباتی را کاهش می دهد، بلکه خطر بیش از حد برازش ( overfit ) را نیز کاهش می دهد، زیرا پارامترهای کمتری تنظیم می شوند.

مزایای استفاده از LoRa بسیار زیاد است. اولاً، امکان فاین تیون مدل‌های بزرگ در کارهایی با داده‌های محدود را فراهم می‌کند، سناریویی که در آن روش‌های فاین تیون سنتی ممکن است منجر به بیش از حد برازش شوند. ثانیاً، LoRa فاین تیون مدل‌های بزرگ را بر روی سخت‌افزار استاندارد امکان‌پذیر می‌کند و با کاهش نیاز به منابع محاسباتی گسترده، دسترسی به مدل‌های پیشرفته را دموکراتیک می‌کند. در نهایت، با کاهش تعداد پارامترهای قابل آموزش، LoRa می‌تواند به همگرایی سریع‌تر منجر شود و فرآیند فاین تیون را کارآمدتر کند.

در نتیجه، LoRa نشان دهنده یک پیشرفت قابل توجه در زمینه یادگیری عمیق، به ویژه در فاین تیون مدل های بزرگ از پیش آموزش دیده است. رویکرد آن برای کاهش بار محاسباتی و کاهش خطر بیش از حد برازش، همگی با حفظ یا حتی بهبود عملکرد مدل، یک گام اساسی به جلو است. با افزایش تقاضا برای مدل‌های شخصی‌سازی‌شده و خاص، تکنیک‌هایی مانند LoRa احتمالاً نقشی محوری در دسترسی بیشتر و کاربردی‌تر کردن این مدل‌ها برای طیف وسیعی از کاربردها ایفا می‌کنند.

روش‌های فاین تیون پارامترهای کارآمد (PEFT) انطباق کارآمد مدل‌های زبان از پیش آموزش‌دیده (PLM) را با روش هایی مانند LoRa و Q-LoRa و .. بدون فاین تیون تمام پارامترهای مدل امکان‌پذیر می‌سازد. فاین تیون PLM های مقیاس بزرگ اغلب بسیار پرهزینه است. در این راستا، روش‌های PEFT فقط تعداد کمی از پارامترهای مدل (اضافی) را تنظیم می‌کنند و در نتیجه هزینه‌های محاسباتی و ذخیره‌سازی را تا حد زیادی کاهش می‌دهند. تکنیک‌های جدید PEFT به عملکردی قابل مقایسه با فاین تیون کامل دست می‌یابند. این کار ها به کمک کتاب خانه ای با همین نام انجام شده است.

## 7) مفهوم bitbytes کوانتیزه کردن – مزیت ها و پیامد های آن

تکنیک‌های کوانتیزاسیون با نمایش وزن‌ها و فعال‌سازی‌ها با انواع داده‌های با دقت پایین‌تر مانند اعداد صحیح 8 بیتی (int8) هزینه‌های حافظه و محاسباتی را کاهش می‌دهد. این امکان بارگیری مدل‌های بزرگ‌تری را فراهم می‌کند که معمولاً نمی‌توانید آن‌ها را در حافظه جا دهید و سرعت استنتاج را افزایش می‌دهد. Transformers از الگوریتم‌های کوانتیزه‌سازی AWQ و GPTQ پشتیبانی می‌کند و از کوانتیزه‌سازی 8 بیتی و 4 بیتی با بیت و بایت پشتیبانی می‌کند.

این کار به کمک کتاب خانه ای با نام bitbyte قابل انجام است به عنوان مثال در مدل مورد نظر سوال 1 میتوان مدل را به صورت با کانفیگ 4 بیتی لود کرد و عملیات کوانتیزاسیون را انجام داد.

```
bnb_config = BitsAndBytesConfig(
        load_in_4bit=True,
        bnb_4bit_quant_type="nf4",
```

```
        bnb_4bit_compute_dtype=torch.float16,
    )

model = AutoModelForCausalLM.from_pretrained(
    "stabilityai/stablelm-3b-4e1t",
    use_safetensors=True,
    quantization_config=bnb_config,
    trust_remote_code=True,
    use_auth_token=True,
    torch_dtype=torch.bfloat16,
)
```

مزایا و پیامد ها :

استفاده از تکنیک‌های کوانتیزه‌سازی، مانند AWQ (کوانتیزه‌سازی وزن تطبیقی) و GPTQ (کوانت‌سازی فاین تیون تدریجی)، و اجرای این تکنیک‌ها با استفاده از کتابخانه‌هایی مانند بیت‌اندبایت برای تبدیل وزن‌ها و فعال‌سازی‌های مدل به انواع داده‌های با دقت پایین‌تر (به عنوان مثال، 8- بیت یا اعداد صحیح 4 بیتی) دارای طیف وسیعی از مزایا و پیامدها هستند:

مزایا:

- کاهش ردپای حافظه: با نمایش وزن ها و فعال سازی ها در 4 بیت به جای float استاندارد 32 بیتی، نیازهای حافظه مدل به طور قابل توجهی کاهش می یابد. این اجازه می دهد تا مدل های بزرگتر در حافظه بارگذاری شوند و کار با مدل های پیچیده روی سخت افزار با منابع حافظه محدود امکان پذیر است.
- استنتاج سریعتر:محاسبات با دقت کمتر معمولاً سریعتر هستند. این امر به ویژه در طول استنتاج مفید است و منجر به زمان پاسخ سریع‌تر می‌شود که در برنامه‌های بلادرنگ بسیار مهم است.
- بهره وری انرژی:کاهش نیازهای محاسباتی همچنین به معنای مصرف انرژی کمتر است و مدل های کوانتیزه را برای استقرار در دستگاه های تلفن همراه و سایر سخت افزارهای با ظرفیت توان محدود مناسب تر می کند.
- بهبود استفاده از پهنای باند: اندازه مدل کوچکتر به دلیل کوانتیزاسیون به این معنی است که اگر مدل در یک تنظیمات توزیع شده یا بر روی edge مستقر شود، داده کمتری باید منتقل شود، که منجر به استفاده بهتر از پهنای باند می شود.

عواقب:

- از دست دادن احتمالی دقت:مبادله اولیه با کوانتیزه کردن، از دست دادن بالقوه دقت مدل است. کاهش دقت وزن‌ها و فعال‌سازی‌ها می‌تواند منجر به از دست دادن توانایی مدل برای نمایش الگوهای ریز در داده‌ها شود. میزان از دست دادن دقت متفاوت است و نیاز به ارزیابی دقیق دارد.
- پیچیدگی در آموزش و تبدیل: آموزش یا فاین تیون مدل های کوانتیزه شده یا تبدیل مدل های از پیش آموزش دیده به نمونه های کوانتیزه شده خود می تواند پیچیدگی بیشتری ایجاد کند. این نیاز به درک چگونگی تأثیر کوانتیزاسیون بر رفتار مدل دارد و ممکن است شامل تنظیم فراپارامتر برای کاهش افت دقت باشد.
- سازگاری سخت افزار: همه سخت افزارها از اجرای کارآمد مدل های کوانتیزه پشتیبانی نمی کنند. در حالی که پردازنده‌ها و پردازنده‌های گرافیکی مدرن به طور فزاینده‌ای از محاسبات با دقت پایین‌تر استفاده می‌کنند، استقرار روی سخت‌افزار خاص ممکن است بدون پشتیبانی خاص برای محاسبات با دقت پایین، سرعت‌های مورد انتظار را به همراه نداشته باشد.
- چالش های پیاده سازی: اجرای کوانتیزاسیون، به‌ویژه طرح‌های تطبیقی مانند AWQ یا GPTQ، می‌تواند چالش برانگیز باشد و ممکن است به درک عمیق معماری مدل و الگوریتم کوانتیزاسیون نیاز داشته باشد. اطمینان از اینکه فرآیند کوانتیزاسیون سوگیری های قابل توجه یا سایر عوارض جانبی ناخواسته را معرفی نمی کند بسیار مهم است.

در نتیجه، در حالی که حرکت به سمت کوانتیزاسیون 4 بیتی با ابزارهایی مانند bitsandbytes می تواند مزایای قابل توجهی را از نظر کارایی و استفاده از منابع به ارمغان آورد، مدیریت دقیق مبادلات، به ویژه از نظر کاهش دقت بالقوه و پیچیدگی پیاده سازی، مهم است. نظارت بر عملکرد مدل و اعتبارسنجی مدل کوانتیزه شده در برابر سناریوهای دنیای واقعی برای اطمینان از اینکه مزایای کوانتیزه کردن بدون به خطر انداختن کاربرد مدل تحقق می‌یابد ضروری است.

## 8) عملیات فاین تیون مدل

با توجه به توضیحات فوق عملیات فاین تیون را انجام میدهیم.

### 8-1) مپ کردن داده ها

در ابتدا لازم است تا داده ها طول آن ها بررسی شود با یک حداکثر طولی فیلتر شوند و در نهایت ورودی مدل که پرامپت میباشد و خروجی نهای مدل که summary میباشد در دیتاست باقی بماند.

با توجه به لود قبلی داده ها بر روی csv از کد زیر برای لود کردن مدل استفاده میشود همچنین لازم است تا pad به توکنایزر اضافه شود.

```
# load dataset from csvs

dataset = load_dataset('csv', data_files={
    'train': 'train.csv',
    'validation': 'validation.csv',
    'test': 'test.csv'
})

# Add [PAD] as the padding token
tokenizer.add_special_tokens({'pad_token': '[PAD]'})
```

در ادامه نیز mapping به کمک تابع زیر انجام میشود و در نهایت ستون ها و نام های header آن ها حذف میشود.

```
def tokenize_prompts_and_summaries(examples):
    """
    Tokenizes the prompts and summaries from the given examples.

    This function takes a dictionary `examples` with keys 'new_prompt' and
'summary'. It tokenizes both the prompts and summaries,
    pads them to a maximum length, and truncates if necessary. The tokenized
inputs are then added to the `examples` dictionary
    under the keys 'input_ids' for the prompts and 'labels' for the summaries.

    Parameters:
    examples (dict): A dictionary containing 'new_prompt' and 'summary' keys.
'new_prompt' is a list of prompts, and 'summary' is a list of corresponding
summaries.

    Returns:
    dict: The original `examples` dictionary updated with 'input_ids' for
tokenized prompts and 'labels' for tokenized summaries.
    """

    # Define the end of the prompt token
    end_prompt_token = '\n\nSummary: '

    # Concatenate end of prompt token with each new prompt
    prompts = [new_prompt + end_prompt_token for new_prompt in
examples["new_prompt"]]

    # Tokenizing the prompts and the summaries
```

```
    tokenized_inputs = tokenizer(prompts, padding="max_length", truncation=True,
max_length=1024, return_tensors="pt")
    tokenized_labels = tokenizer(examples["summary"], padding="max_length",
truncation=True, max_length=1024, return_tensors="pt")

    # Extracting input_ids for inputs and labels
    examples['input_ids'] = tokenized_inputs.input_ids
    examples['labels'] = tokenized_labels.input_ids

    return examples


# Apply the function to the dataset
tokenized_datasets = dataset.map(tokenize_prompts_and_summaries, batched=True)
```

```
# remove columns
tokenized_datasets = tokenized_datasets.remove_columns(['conversation', 'summary'
, 'new_prompt'])
```

اگر نمیخواستیم داده ها را از csv لود کنیم میتوانستیم از دستور زیر استفاده کنیم:

```
def tokenize_function(example):
    example['input_ids'] = tokenizer(example["new_prompt"], padding="max_length",
truncation=True, return_tensors="pt").input_ids
    example['labels'] = tokenizer(example["summary"], padding="max_length",
truncation=True, return_tensors="pt").input_ids

    return example

tokenized_datasets = dataset.map(tokenize_function)
tokenized_datasets = tokenized_datasets.remove_columns(['conversation',
'new_prompt', 'summary'])
```

در نهایت فرم نهایی دیتاست به صورت زیر میشود.

```
DatasetDict({
    train: Dataset({
        features: ['input_ids', 'labels'],
        num_rows: 879
    })
    validation: Dataset({
        features: ['input_ids', 'labels'],
        num_rows: 110
    })
```

```
    test: Dataset({
        features: ['input_ids', 'labels'],
        num_rows: 110
    })
})
```

### 8-2) استفاده از PEFT

ابتدا یک تابع جهت محاسبه تعداد پارامتر ها و همچنین تعداد پارامتر های قابل آموزش جهت اطلاعات مدل نوشته میشود.

```
def count_model_parameters(model):
    """
    Counts the total and trainable parameters of the given model.

    This function iterates through all parameters of the provided model, counting the total number of parameters
    and the number of trainable parameters (parameters with requires_grad=True).

    Parameters:
    model (torch.nn.Module): The model whose parameters are to be counted.

    Returns:
    tuple: A tuple containing the number of trainable parameters and the total number of parameters.
    """
    trainable_params = sum(p.numel() for p in model.parameters() if p.requires_grad)
    total_params = sum(p.numel() for p in model.parameters())
    return trainable_params, total_params

def print_model_parameters_info(model):
    """
    Prints the number of total and trainable parameters of the given model, along with the percentage of trainable parameters.

    Parameters:
    model (torch.nn.Module): The model whose parameter information is to be printed.
    """
    trainable_params, total_params = count_model_parameters(model)
    percentage_trainable = 100 * trainable_params / total_params
    info_message = (
        f"Trainable model parameters: {trainable_params}\n"
        f"All model parameters: {total_params}\n"
        f"Percentage of trainable model parameters: {percentage_trainable:.2f}%"
```

```
    )
    print(info_message)
```

حال تعداد پارامتر های مدل اصلی بررسی میشود.

```
# print model info
print_model_parameters_info(model)

Trainable model parameters: 2795443200
All model parameters: 2795443200
Percentage of trainable model parameters: 100.00%
```

در ادامه از PEFT استفاده میکنیم.

```
lora_r = 16
lora_alpha = 64
lora_dropout = 0.1
lora_target_modules = [
    "q_proj",
    "v_proj"
]


peft_config = LoraConfig(
    r=lora_r,
    lora_alpha=lora_alpha,
    lora_dropout=lora_dropout,
    target_modules=lora_target_modules,
    bias="none",
    task_type="CAUSAL_LM",
)
```

نیاز است تا تک به تک پارامتر ها و دلیل استفاده از آن را بررسی کنیم. بیایید ابتدا بررسی کنیم این اجزا چیست و سپس به تغییرات لایه خاص و پیامدهای آنها بپردازیم.

LoRA همانطور که گفته شد یک تکنیک آموزشی با پارامتر کارآمد است که ماتریس های قابل آموزش با رتبه پایین را برای تطبیق وزن مدل های از پیش آموزش دیده مانند ترنسفورمر معرفی می کند. به جای فاین تیون همه پارامترها، LoRA بر روی چند ماتریس وزن کلیدی تمرکز می‌کند، که باعث می‌شود فرآیند کارآمدتر و کمتر مستعد بیش‌برازش شود.

**پارامترهای پیکربندی PEFT :**

- **lora_r**: این نشان دهنده رتبه ماتریس های با رتبه پایین است. «r» کوچکتر به معنای پارامترهای کمتری برای آموزش است.
- **lora_alpha**: این به روز رسانی های رتبه پایین را مقیاس می کند. «آلفا» بالاتر به این معنی است که به‌روزرسانی‌ها تأثیر بیشتری دارند.
- **`lora_dropout`**: این نرخ حذف برای لایه های LoRA است که به منظم کردن مدل (reguralization ) و جلوگیری از برازش بیش از حد کمک می کند.
- **`lora_target_modules`**: ماژول ها (یا لایه ها) را در مدل ترنسفورمر که در آن LoRA اعمال می شود، مشخص می کند. در این مورد ، «q_proj» و «v_proj» را هدف قرار داده شده است، که بخش‌هایی از مکانیسم Multi head Attention هستند.

**چرایی انتخاب لایه ها:**

`"q_proj"` و `"v_proj"`: اینها اجزای multi head attention در مدل های ترنسفورمر هستند. لایه 'q_proj' مسئول نمایش نشانه های ورودی در فضای پرس و جو است و `v_proj` توکن های ورودی را در فضای مقدار پروژکشن می دهد. این پیش بینی ها برای نحوه محاسبه توجه ترنسفورمرها اساسی هستند و بنابراین در تعیین تمرکز مدل در طول پردازش بسیار مهم هستند.

چرا خوب است اینجا را تغییر دهیم ؟

کارایی در یادگیری: با اعمال LoRA در "q_proj" و "v_proj"، مستقیماً بر نحوه محاسبه attention مدل تأثیر می‌گذاریم. از آنجایی که attention بخش مهمی از ترنسفورمرها است، حتی تغییرات کوچک و کارآمد می تواند منجر به بهبود قابل توجهی در نحوه درک و تولید متن توسط مدل شود.

کارایی پارامتر: انتخاب "lora_r" و "lora_alpha" نشان دهنده تعادل بین سازگاری و کارایی مدل است. با لمس نکردن همه پارامترها، بلکه فقط مؤلفه‌های کلیدی («q_proj» و «v_proj»)، اطمینان حاصل می‌کنید که فاین تیون‌تر و کارآمدتر است.

جلوگیری از تطبیق بیش از حد: استفاده از "lora_dropout" به منظم کردن روند آموزش کمک می کند.

ویژگی TASK: پارامتر روی «CAUSAL_LM» تنظیم شده است، که نشان می‌دهد مدل برای یک کار مدل‌سازی زبان علّی (مانند پیش‌بینی کلمه بعدی، که معمولاً در GPT- استفاده می‌شود) دقیق تنظیم می‌شود. مانند معماری ها). این ویژگی به این معنی است که تغییراتی که ایجاد می‌کنید برای بهبود عملکرد در این نوع خاص از کار طراحی شده‌اند.

پس از توضیحات فوق مدل peft جدید را تشکیل میدهیم و اطلاعات آن را میگیریم.

```
peft_model = get_peft_model(model, peft_config)

# Print peft info
print_model_parameters_info(peft_model)

Trainable model parameters: 5242880
All model parameters: 2800686080
Percentage of trainable model parameters: 0.19%
```

همانطور که مشاهده میشود از مجموع 2.8 میلیارد پارامتر فقط 0.19٪ آن آموزش داده میشود که مقدار بسیار کوچکی میباشد.

### 3-8) فرایند فاین تیون مدل

پارامتر های آموزش مدل را به صورت زیر تعیین میکنیم.

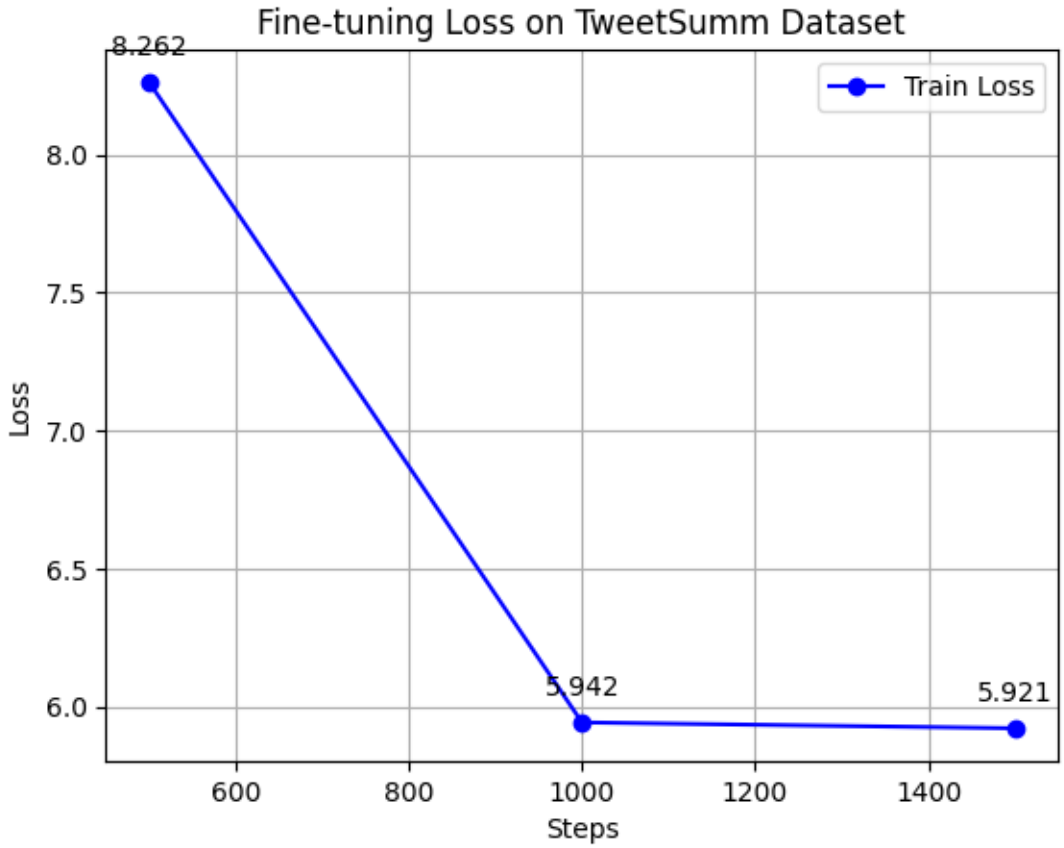
```
peft_training_args = TrainingArguments(
    output_dir='./results',
    num_train_epochs=4,
    per_device_train_batch_size=1,
    per_device_eval_batch_size=1,
    warmup_steps=500,
    save_steps=500,
    eval_steps=500,
    weight_decay=0.01,
    report_to=["tensorboard"],
    logging_dir='./logs',
)


peft_trainer = Trainer(
    model=peft_model,
    args=peft_training_args,
    train_dataset=tokenized_datasets["train"],
    eval_dataset=tokenized_datasets["validation"]
)
```

و درنهایت نیز مدل را آموزش میدهیم.

```
peft_trainer.train()
```

مدت زمان آموزش حدود 30 دقیقه ساعت بر روی 3090 طول میکشد.

منحنی loss برای ترین به شرح زیر میباشد.




شکل 2 : منحنی لاس برای فاین تیون سوال 1

که با توجه به سیر نزولی آن نشان از اندکی بهتر شدن و تیون دیتاست بر روی مدل دارد.

در نهایت نیز مدل PEFT را ذخیره میکنیم.

```
peft_model_path="./peft-model-checkpoint"

peft_trainer.model.save_pretrained(peft_model_path)
tokenizer.save_pretrained(peft_model_path)
```

## 9) ارزیابی مدل فاین تیون شده

جهت ارزیابی نیاز است تا مجددا مدل ها را لود کنیم.

```
# load model again
tokenizer = AutoTokenizer.from_pretrained("stabilityai/stablelm-3b-4e1t",
use_auth_token=True)
model = AutoModelForCausalLM.from_pretrained("stabilityai/stablelm-3b-4e1t",
torch_dtype=torch.bfloat16,use_auth_token=True, trust_remote_code=True)
```

در ادامه نیز مدل فاین تیون شده را لود میکنیم ( با همان کانفیگ LoRa )

```
from peft import LoraConfig, get_peft_model, TaskType

lora_r = 16
lora_alpha = 64
lora_dropout = 0.1
lora_target_modules = [
    "q_proj",
    "v_proj"
]


peft_config = LoraConfig(
    r=lora_r,
    lora_alpha=lora_alpha,
    lora_dropout=lora_dropout,
    target_modules=lora_target_modules,
    bias="none",
    task_type="CAUSAL_LM",
)

peft_model = get_peft_model(model, peft_config)

# Load PEFT Fine Tuned Model

peft_model = PeftModel.from_pretrained(peft_model,
                                       './peft-model-checkpoint-new/',
                                       torch_dtype=torch.bfloat16,
                                       is_trainable=False)
```

در ادامه نیز مدل را برا تست بر روی GPU میبریم.

```
peft_model.to("cuda:1")
peft_model.eval()
print("peft_model loaded")
```

در نهایت نیز میتوانیم به راحتی خروجی مدل را بر روی مدل اصلی و مدل تیون شده بگیریم و مقایسه کنیم.

```
index = 100
dialogue = dataset['test'][index]['conversation']
baseline_human_summary = dataset['test'][index]['summary']

prompt = f"""
Summarize the following conversation.

{dialogue}

Summary: """

input_ids = tokenizer(prompt, return_tensors="pt")
inputs = {key: value.to("cuda:1") for key, value in input_ids.items()}

generation_config = GenerationConfig(
    max_length=512,
    num_beams=1,
    pad_token_id=tokenizer.eos_token_id  ,
    temperature=0,

)

peft_model_outputs = peft_model.generate(input_ids=inputs["input_ids"],
generation_config=generation_config)
peft_model_text_output = tokenizer.decode(peft_model_outputs[0],
skip_special_tokens=True)

dash_line = '-'.join('' for x in range(100))
print(dash_line)
print(f'BASELINE HUMAN SUMMARY:\n{baseline_human_summary}')
print(dash_line)
print(f'PEFT MODEL: {peft_model_text_output}')
```

خروجی به عنوان مثال در این حالت به صورت زیر میباشد.

```
BASELINE HUMAN SUMMARY:
Customer is asking help that  how to remove red eye in ligth room cc even he cant
find it in tool  and even customer want some new advance features. Agent is
giving details on it and then sends a link where he can get help and also asked
customer to  report a  complaint where  his engineer team will  get alert and
help him over it.
---------------------------------------------------------------------------------
------------------
```

```
PEFT MODEL:
Summarize the following conversation.

user: Can you tell me how to do Red Eye Removal in Lightroom CC? I just moved to
it and don't see the Red Eye Removal tool.
agent: Hi Bob, here is a link to show you to use the Red eye removal in Lightroom
CC.
user: Does not apply to the NEW LightRoom CC. Any other suggestions?
agent: Bob, I will loop in our Lightroom expert to help you with this. The
setting may have moved to a different location.  Hi Bob, I am looping our expert
team to help answer your question. They will get back to you ASAP. Please excuse
the delay, if any. Thanks!  Hi Bob, Yes, its not there in Lightroom CC also,
refer: Thanks.
user: Thank you. I wish a list of feathers missing in Lightroom CC would have
been noted before I migrated my library. Never thought a commercial photo app
from Adobe would omit a basic feature like that. *features
agent: Hi Bob, you can report this here to alert our product teams and engineers:
Thanks!  Hi Bob, this feature is not available in Lightroom CC as of now, however
you may suggest it as a feature here:.
user: Hate to be “that guy” but this is a Photo Editing 101 feature. Where is the
list of what’s missing from the “new” Lightroom CC? Also, it would be great if
included “Lightroom CC” in its support system. Only “PhotoShop Lightroom” is
listed on that page. So if I request it, I'd probably get back an "Already
available" response.
agent: We have released Lightroom Classic CC which has all the features the old
Lightroom CC 2015.12 had, you can check this article to see the differences
betweem LR Classic &amp;amp; the new Lightroom CC:.


Summary:
The conversation is about a missing feature in Lightroom CC. The user is
frustrated that the feature is missing. The agent is trying to help the user by
providing a link to a page that shows how to use the feature. The user is not
satisfied with the response and asks for a list of missing features. The agent is
not sure what the user is asking for and asks for help from a Lightroom expert.
The user is not satisfied with the response and asks for a list of missing
features. The agent is not sure what
```

که مشاهده از اندکی بهتر شدن مدل فاین تیون شده دارد در نهایت نیز میتوانیم از معیار هایی مانند ROUGE برای ارزیابی استفاده کنیم.

اما لازم است کمی در مورد این معیار گفته شود.

ROUGE که مخفف عبارت Recall-Oriented Understudy for Gisting Evaluation است، مجموعه‌ای از معیارهایی است که معمولاً در پردازش زبان طبیعی (NLP) برای ارزیابی سیستم‌های تولید

متن استفاده می‌شود. ROUGE که در اصل برای کارهای خلاصه سازی توسعه یافته بود، کاربرد گسترده ای در کارهای مختلف NLP، از جمله خلاصه سازی متن، ترجمه ماشینی، و تولید متن پیدا کرده است.

هدف اصلی معیارهای ROUGE اندازه گیری شباهت بین متن تولید شده و متن مرجع (تولید شده توسط انسان) است. این ارزیابی شباهت برای ارزیابی کیفیت و اثربخشی سیستم‌های تولید خودکار متن بسیار مهم است. معیارهای ROUGE بر جنبه‌هایی مانند دقت، recall و F1 تمرکز می‌کنند تا همپوشانی بین متن تولید شده و متن مرجع را بر حسب n گرم (توالی‌های پیوسته از n مورد، معمولاً کلمات) کمّی کنند.

در اینجا برخی از مؤلفه‌های کلیدی معیارهای ROUGE که معمولاً در زمینه اعتبارسنجی تولید متن استفاده می‌شوند، آمده است:

- ROUGE-N (N-gram Overlap): این متریک همپوشانی n-گرم بین متن تولید شده و متن مرجع را ارزیابی می کند. به طور معمول، ROUGE-N برای مقادیر مختلف N محاسبه می‌شود. دقت، فراخوانی و F1 بر اساس تعداد n-گرم های همپوشانی محاسبه می شود.
- ROUGE-L (طولانی ترین دنباله مشترک): ROUGE-L طولانی ترین دنباله مشترک بین متن تولید شده و متن مرجع را اندازه گیری می کند. ارزیابی انعطاف‌پذیرتری ارائه می‌دهد، زیرا شباهت دنباله‌های کلمات را به جای تطابق دقیق در نظر می‌گیرد. دقت، فراخوانی و امتیاز F1 بر اساس طول طولانی‌ترین دنباله مشترک محاسبه می‌شود.
- ROUGE-W (همپوشانی وزنی): این متریک وزن هایی را به n-گرم ها بر اساس فراوانی وقوع آنها در متن مرجع اختصاص می دهد. هدف آن اهمیت دادن بیشتر به کلمات یا عبارات مهم است. مشابه ROUGE-N، دقت، فراخوانی و امتیاز F1 را با همپوشانی وزنی n گرم محاسبه می کند.
- ROUGE-S (Skip-bigram Overlap): ROUGE-S همپوشانی skip-bigrams را ارزیابی می کند، که n-gram هستند که تعداد معینی از کلمات میانی را امکان پذیر می کنند. این به ویژه برای گرفتن شباهت معنایی بین متون تولید شده و مرجع مفید است.
- ROUGE-P (دقت) این متریک نسبت همپوشانی n-گرم را در متن تولید شده نسبت به متن مرجع اندازه گیری می کند. بینشی در مورد دقت محتوای تولید شده ارائه می دهد.
- ROUGE-R (recall):ROUGE-R نسبت همپوشانی n-گرم در متن تولید شده را نسبت به متن مرجع کمیت می کند. روی جنبه یادآوری تمرکز می کند و نشان می دهد که متن تولید شده چقدر اطلاعات موجود در متن مرجع را به خوبی دریافت می کند.

برای این کار ابتدا 10 نمونه از تست را میگیریم.

```
dialogues = dataset['test'][0:10]['conversation']
human_baseline_summaries = dataset['test'][0:10]['summary']

peft_model_summaries = []
generation_config = GenerationConfig(
    max_length=512,
    num_beams=1,
    pad_token_id=tokenizer.eos_token_id  ,
    temperature=0,

)
for idx, dialogue in enumerate(dialogues):
    prompt = f"""
Summarize the following conversation.

{dialogue}

Summary: """

    input_ids = tokenizer(prompt, return_tensors="pt")
    inputs = {key: value.to("cuda:1") for key, value in input_ids.items()}


    human_baseline_text_output = human_baseline_summaries[idx]


    peft_model_outputs = peft_model.generate(input_ids=inputs["input_ids"],
generation_config=generation_config)
    peft_model_text_output = tokenizer.decode(peft_model_outputs[0],
skip_special_tokens=True)
    last_summary_part = peft_model_text_output.split('Summary: ')[-1]  # Get the
last part of the summary

    peft_model_summaries.append(last_summary_part)
```

در ادامه مقدار rouge را با استفاده از کتابخانه evaluate محاسبه میکنیم.

```
rouge = evaluate.load('rouge')


peft_model_results = rouge.compute(
    predictions=peft_model_summaries,
    references=human_baseline_summaries[0:len(peft_model_summaries)],
    use_aggregator=True,
```

```
    use_stemmer=True,
)

print('PEFT MODEL:')
print(peft_model_results)
```

که بدین ترتیب مقدار آن را مقایسه میکنیم.

در این ازمایش مقدار rouge مدل اولیه و مدل Fine Tune شده در جدول زیر با هم مقایسه شده است.

جدول 1: مقایسه **Rouge** برای مدل های مختلف

| Model | Rouge1 | Rouge2 | RougeL | RougeLsum |
|---|---|---|---|---|
| Base | 0.1166 | 0.0370 | 0.10592 | 0.10665 |
| PEFT FineTune | 0.12804 | 0.0387 | 0.10721 | 0.10683 |

همانطور که مشاهده میشود تنها اندکی بهتر شده است. این میتواند به خاطر پارامتر های کم مدل که تغییر کرده و low rank شده است میباشد و همچنین تعداد step های کم و مدت زمان ترین فقط 30 دقیقه میتواند باعث آن باشد ( البته لاس روند نزولی ای دارد و میتواند ادامه دار باشد.)

## 10) ارزیابی مدل فاین تیون شده از نظر Zero Shot و few show

به کمک دستور زیر میتوان ارزیبای مدل را از نظر zeroshot و one shot با مدل جدید مقایسه کنیم.

در اینجا اول کد ها آمده است و در ادامه هم نتایج نهایی few shot و zero shot در مدل تیون شده و مدل base line و label آن با هم مقایسه شده.

کد zero shot :

```
index = 100
dialogue = dataset['test'][index]['conversation']
baseline_human_summary = dataset['test'][index]['summary']

prompt = f"""
Please provide a concise and accurate summary of the following conversation
between a human and an AI agent.
{dialogue}
```

```
Summary: """

input_ids = tokenizer(prompt, return_tensors="pt")
inputs = {key: value.to("cuda:1") for key, value in input_ids.items()}

generation_config = GenerationConfig(
    max_length=2048,
    num_beams=1,
    pad_token_id=tokenizer.eos_token_id  ,
    temperature=0.7,

)

peft_model_outputs = peft_model.generate(input_ids=inputs["input_ids"],
generation_config=generation_config)
peft_model_text_output = tokenizer.decode(peft_model_outputs[0],
skip_special_tokens=True)


dash_line = '-'.join('' for x in range(100))
print(dash_line)
print(f'BASELINE HUMAN SUMMARY:\n{baseline_human_summary}')
print(dash_line)
print(f'Peft Model Zero shot MODEL:\n{peft_model_text_output}')
```

کد One shot :

```
# Select a random conversation-summary pair from the train dataset for the one-
shot example
example_conversation_train = dataset['train'][0]['conversation']
example_summary_train =  dataset['train'][0]['summary']

# Select the conversation and the human-generated summary from the test dataset
index = 100
dialogue_test = dataset['test'][index]['conversation']
baseline_human_summary = dataset['test'][index]['summary']


# Prepare the one-shot prompt
prompt = f"""
### Example Conversation:
{example_conversation_train}

### Example Summary:
{example_summary_train}
```

```
---

### New Conversation:
{dialogue_test}

### Please provide a concise and accurate summary for the new conversation above:
"""

input_ids = tokenizer(prompt, return_tensors="pt")
inputs = {key: value.to("cuda:1") for key, value in input_ids.items()}

generation_config = GenerationConfig(
    max_length=1024,
    num_beams=1,
    pad_token_id=tokenizer.eos_token_id,
    temperature=0.7,
)

# Generate the summary for the new conversation
peft_model_outputs = peft_model.generate(input_ids=inputs["input_ids"],
generation_config=generation_config)
peft_model_text_output = tokenizer.decode(peft_model_outputs[0],
skip_special_tokens=True)

# Output the results
dash_line = '-' * 100
print(dash_line)
print(f'BASELINE HUMAN SUMMARY:\n{baseline_human_summary}')
print(dash_line)
print(f'ONE-SHOT MODEL OUTPUT:\n{peft_model_text_output}')
```

نتیجه در این حالت برای Zero Shot داریم :

```
The conversation was about a missing feature in Lightroom CC. The
agent was unable to provide a solution. The agent suggested the user
report the missing feature to Adobe. The user was not satisfied with
the response.
```

نتیجه در حالت Oneshot :

```
The customer is complaining that he is not able to find the Red Eye
Removal tool in Lightroom CC. The agent answered that the feature is
not available in Lightroom CC as of now.
```

نتایج انچنان تغییر نکرده است و در کل یکم هم معیار rouge بیشتر شده است که این میتواند به دلایلی همچئن لود کردن 8 بیتی مدل از ابتدا و آموزش تنها 0.19 درصد پارامتر ها میباشد.

راهکار ها برای بهتر شدن :

- مدت زمان آموزش بیشتر و بهینه سازی پارامتر های بهینه سازی آموزش مدل
- اضافه کردن درصد پارامتر های قابل آموزش در مدل
- به جای fine tune از روش هایی مانند RAG استفاده شود و از دیتابیس آموزش tweet sum استفاده شود.

## 11) منابع استفاده شده

منابع استفاده شده برای سوال 1 :

https://github.com/Ryota-Kawamura/Generative-AI-with-LLMs/blob/main/Week-2/Lab_2_fine_tune_generative_ai_model.ipynb

# سوال ۲ – Prompt Engineering

## 1) بررسی مقاله Scaling Instruction-Finetuned Language Models

در این بخش ، بخش های مختلف مقاله به طور خلاصه بیان شده است.

### 1-1) مقدمه

مقدمه مقاله بر توسعه مدل‌های هوش مصنوعی متمرکز است که قادر به تعمیم به وظایف غیرقابل مشاهده هستند، به ویژه در قلمرو پردازش زبان طبیعی (NLP). این پیشرفت حاصل از مدل های زبانی از پیش آموزش دیده را نشان می دهد که می توانند وظایف را بر اساس توصیفات زبان طبیعی انجام دهند. یک پیشرفت قابل توجه، فاین تیون این مدل ها بر روی مجموعه ای از وظایف است که به عنوان دستورالعمل بیان می شوند، که پاسخگویی مدل ها به دستورالعمل ها را افزایش می دهد و نیاز به نمونه های چندتایی را کاهش می دهد.

این مقاله با بررسی مقیاس‌پذیری آن با تعداد تسک ها و اندازه مدل و تأثیر آن بر قابلیت‌های استدلال مدل‌ها، فاین تیونinstruction را بیشتر پیش می‌برد. نویسندگان دریافته‌اند که فاین تیون instruction با تعداد کارها و اندازه مدل به طور موثر مقیاس می‌شود. به طور خاص، ادغام تعداد کمی از مجموعه داده

های زنجیره ای از فکر (CoT) در ترکیب فاین تیون، عملکرد را در ارزیابی های مختلف به طور قابل توجهی بهبود می بخشد.

این مقاله Flan-PaLM را معرفی می‌کند، مدلی با پارامتر 540B که به‌خوبی با وظایف 1.8K، از جمله داده‌های CoT تنظیم شده است. Flan-PaLM عملکرد برتر را در مقایسه با مدل قبلی خود، PalM نشان می دهد و معیارهای جدیدی را در زمینه های مختلف ایجاد می کند. به عنوان مثال، از CoT و خودسازگاری برای دستیابی به امتیاز 75.2٪ در معیار سنجش درک زبان چند وظیفه‌ای عظیم (MMLU) استفاده می‌کند. همچنین پیشرفت های قابل توجهی را در قابلیت های چند زبانه و ارزیابی های ارزیابی کننده انسانی نشان می دهد که نشان دهنده بهبود قابلیت استفاده است. علاوه بر این، فاین تیون دستورالعمل عملکرد را در چندین معیار ارزیابی هوش مصنوعی افزایش می دهد.

این مقاله همچنین فاین تیون مدل‌های Flan-T5 را مورد بحث قرار می‌دهد که از پارامترهای M80 تا B11 متغیر است. این مدل‌ها توانایی‌های zero shot، few shot و CoT را نشان می‌دهند که از مدل‌های قبلی مانند T5 و حتی PalM در کارهای خاص بهتر عمل می‌کنند. نتایج بر اثربخشی فاین تیون دستورالعمل در افزایش عملکرد، قابلیت استفاده و تطبیق پذیری مدل های زبان از پیش آموزش دیده در مدل های مختلف، تنظیمات اولیه و وظایف ارزیابی تاکید می کند.

### 2-1) Flan FineTuning

بخش 2 مقاله، با عنوان "Flan Finetuning"، فرآیند فاین تیون مدل‌های زبان را با وظایف مبتنی بر دستورالعمل برای بهبود عملکرد و تعمیم آن‌ها به وظایف نامرئی مورد بحث قرار می‌دهد. این بخش پیشتر به سه بخش تقسیم می شود: 2.1 داده های فاین تیون، 2.2 رویه فاین تیون و 2.3 پروتکل ارزیابی.

فاین تیون داده ها:

این بخش داده های مورد استفاده برای فاین تیون مدل ها را توضیح می دهد. نویسندگان با ترکیب چهار ترکیب کار از مطالعات قبلی، تا 1836 کار فاین تیون را مقیاس می‌دهند: Muffin، T0-SF، NIV2، و CoT (زنجیره فکر). این مخلوط ها شامل وظایفی مانند داده های گفتگو، ترکیب برنامه، استدلال حسابی و استنتاج زبان طبیعی است. این مقاله بر اهمیت طیف متنوعی از وظایف برای بهبود توانایی مدل برای تعمیم به کارهای جدید و دیده نشده تأکید می کند. این مقاله همچنین قالب‌بندی داده‌های فاین تیون را مورد بحث قرار می‌دهد و ذکر می‌کند که آنها از الگوهای آموزشی برای هر کار استفاده می‌کنند و به صورت دستی حدود ده الگوی دستورالعمل را برای هر یک از 9 مجموعه داده در مخلوط CoT می‌نویسند.

رویه فاین تیون:

این بخش روش فاین تیون اعمال شده در طیف وسیعی از خانواده‌های مدل، از جمله T5، PaLM، و U-PaLM را توصیف می‌کند که اندازه آنها به طور قابل توجهی متفاوت است. این روش شامل استفاده از یک برنامه نرخ یادگیری ثابت و فاین تیون با استفاده از بهینه ساز Adafactor است. این مقاله روش packaging را برای ترکیب چندین مثال آموزشی در یک توالی توضیح می دهد و به استفاده از ماسک برای جلوگیری از attention توکن ها به دیگران در سراسر مرز نمونه package شده اشاره می کند. همچنین کارایی محاسباتی فرآیند فاین تیون را برجسته می‌کند و اشاره می‌کند که تنها از بخش کوچکی از محاسبه نسبت به محاسبات آموزشی استفاده می‌کند.

پروتکل ارزیابی:

نویسندگان بر عملکرد مدل در کارهایی که بخشی از داده‌های فاین تیون نیستند، تمرکز می‌کنند. ارزیابی بر روی طیف وسیعی از معیارها، از جمله MMLU، BBH، TyDiQA، و MGSM انجام می‌شود که طیف وسیعی از موضوعات و زبان‌ها را پوشش می‌دهند. این مقاله، بسته به ماهیت کار و معیار، استفاده از تحریک مستقیم و تحریک زنجیره‌ای از فکر (CoT) را برای ارزیابی توصیف می‌کند. همچنین یک معیار «میانگین نرمال شده» را به عنوان معیاری مجموع برای شش نمره نرمال شده از معیارهای مختلف معرفی می کند.

در پایان، این بخش از مقاله یک مرور کلی از فرآیند فاین تیون، تنوع و قالب بندی داده های فاین تیون، روش فاین تیون، و پروتکل ارزیابی دقیق برای ارزیابی عملکرد فاین تیون ارائه می دهد. مدل هایی در طیف گسترده ای از وظایف و زبان ها.

### 3-1) Scaling to 540B and 1.8k task

بخش «مقیاس‌سازی به پارامترهای B540 و وظایف K1.8» در این مقاله تأثیر مقیاس‌گذاری اندازه مدل و تعداد وظایف فاین تیون بر عملکرد مدل‌های زبان را مورد بحث قرار می‌دهد. این مطالعه اندازه‌های مدل را در پارامترهای B8، B62، و B540 مقیاس‌بندی می‌کند و با افزودن متوالی ترکیب کار (CoT، Muffin، T0-SF، NIV2) تعداد وظایف فاین تیون را افزایش می‌دهد. نتایج نشان می‌دهد که فاین تیون دستورالعمل‌های چند کاره به طور قابل‌توجهی عملکرد را در تمام اندازه‌های مدل افزایش می‌دهد، با افزایشی از ۹.۴٪ تا ۱۵.۵٪ در مقایسه با مدل‌های بدون فاین تیون. بهبود عملکرد به ویژه در هنگام افزایش تعداد وظایف فاین تیون تا 282 قابل توجه است، که نشان می دهد اکثر دستاوردها از مدل استفاده از

دانش از قبل موجود به جای کسب دانش جدید از وظایف اضافی ناشی می شود. این مطالعه همچنین بهبود عملکرد قابل توجهی را هنگامی که مقیاس مدل با مرتبه‌ای افزایش می‌یابد مشاهده می‌کند، اگرچه مزایای نسبی در کاهش نرخ خطا برای مدل‌های بزرگ‌تر قابل توجه‌تر است.

تجزیه و تحلیل بیشتر نشان می دهد که در حالی که افزایش تعداد وظایف فاین تیون عملکرد را افزایش می دهد، سود نهایی پس از 282 کار کاهش می یابد. این پدیده به عدم تنوع در وظایف اضافی یا ظرفیت مدل برای استفاده مؤثر از دانش از پیش موجود نسبت داده می شود. علاوه بر این، این مطالعه بررسی می‌کند که چگونه مقیاس‌بندی اندازه مدل و تعداد وظایف به طور بالقوه می‌تواند منجر به افزایش عملکرد شود، و پیشنهاد می‌کند که تحقیقات آینده باید به بررسی دقیق‌سازی دستورالعمل‌ها، به‌ویژه در مقیاس‌سازی اندازه‌های مدل، ادامه دهند، حتی اگر مزایای افزایش تعداد وظایف فاین تیون باشد. ممکن است افزایشی باشد.

### 4-1) Finetuning with COT annotation

بخش 4 مقاله، با عنوان "فاین تیون با COT"، تاثیر ادغام annotation های زنجیره ای از فکر (CoT) را در فرآیند فاین تیون مدل های زبان بررسی می کند. این بخش به تفصیل و به سه بخش فرعی تقسیم شده است:

فاین تیون در زنجیره فکر، استدلال را در کارهای انجام شده بهبود می بخشد:

این بخش فرعی تجزیه و تحلیل عمیقی از مزایای استفاده از CoT annotation در فرآیند فاین تیون ارائه می دهد. نکات کلیدی عبارتند از:

- افزایش عملکرد: گنجاندن داده های CoT در مخلوط فاین تیون منجر به بهبود قابل توجهی در قابلیت های استدلال مدل می شود. قابل ذکر است، مدل Flan-PaLM که با استفاده از مجموعه داده‌های بزرگ شامل داده‌های CoT تنظیم شده است، عملکرد استثنایی را به نمایش می‌گذارد، بهتر از PalM عمل می‌کند و معیارهای جدیدی را در وظایف استدلالی مختلف تعیین می‌کند.
- دستاوردهای معیار: مدل Flan-PaLM پیشرفت های قابل توجهی را در چندین معیار نشان می دهد. به عنوان مثال، در معیار MMLU، Flan-PaLM به 75.2 درصد رسیده است، به طور قابل توجهی بهتر از مدل های قبلی. همچنین عملکرد قابل توجهی را در معیارهای چند زبانه نشان می دهد و توانایی های استدلال برتر و قابلیت های چند زبانه آن را برجسته می کند.

- استفاده از خودسازگاری با CoT: ترکیب درخواست CoT با خود سازگاری (SC) عملکرد مدل را بیشتر افزایش می دهد. این رویکرد منجر به نتایج پیشرفته‌ای در معیارهای مختلف می‌شود، که نشان می‌دهد درخواست CoT، زمانی که در کنار SC استفاده می‌شود، می‌تواند ابزاری قوی برای تقویت عملکرد مدل در وظایف استدلال پیچیده باشد.

برخی از داده های زنجیره ای برای حفظ توانایی استدلال مورد نیاز است:

این بخش به نقش حیاتی داده‌های CoT در حفظ قابلیت‌های استدلال مدل می‌پردازد. بینش های کلیدی عبارتند از:

- اهمیت داده های CoT: تجزیه و تحلیل نشان می دهد که فاین تیون ترکیبی از داده های CoT و غیر CoT برای حفظ عملکرد در وظایف استدلال بسیار مهم است. فاین تیون انحصاری داده‌های غیر CoT می‌تواند منجر به کاهش عملکرد مدل در ارزیابی‌های CoT شود.
- رویکرد فاین تیون متوازن: این مطالعه یک رویکرد متعادل را برای فاین تیون پیشنهاد می کند، که در آن ترکیبی از داده های CoT و غیر CoT استفاده می شود. این روش تضمین می‌کند که مدل در معیارهای CoT و غیر CoT به خوبی عمل می‌کند و اهمیت مجموعه داده‌های فاین تیون متنوع را برجسته می‌کند.

Unlock استدلال صفر شات( zero shot ) :

بخش فرعی پایانی به توانایی مدل‌های تنظیم‌شده برای اجرای استدلال صفر شات می‌پردازد، که جنبه‌ای مهم از توانایی مدل‌های هوش مصنوعی برای تعمیم و استدلال بدون مثال‌های صریح است. مشاهدات کلیدی عبارتند از:

- استدلال CoT صفر شات: مدل Flan-PaLM که با داده های CoT به خوبی تنظیم شده است، توانایی اجرای استدلال CoT را در یک محیط صفر شات نشان می دهد. این نشان می‌دهد که مدل می‌تواند مراحل استدلالی ایجاد کند و بدون نیاز به نمونه‌های کمی برای CoT به نتیجه برسد.
- عملکرد برتر در وظایف چالش برانگیز: قابلیت های استدلال CoT صفر-شات مدل به ویژه در کارهای چالش برانگیز مانند موارد موجود در معیار BIG-Bench مشهود است. مدل‌های Flan-PaLM به طور قابل‌توجهی از مدل‌های پایه PaLM بهتر عمل می‌کنند و اثربخشی فاین تیون CoT را نشان می‌دهند.

- نقش عبارت «بیایید گام به گام فکر کنیم» : عبارت «بیایید گام به گام فکر کنیم» در فعال کردن قابلیت‌های استدلال CoT مدل مؤثر است. به نظر می رسد این عبارت مدل را وادار می کند تا در یک فرآیند استدلال گام به گام شرکت کند، که منجر به افزایش عملکرد در وظایف استدلال می شود.

در پایان، بخش 4 مقاله کاوشی جامع از مزایا و روش‌های ترکیب annotation زنجیره‌ای از فکر در فرآیند فاین تیون مدل‌های زبانی را ارائه می‌کند. این بخش شواهد قابل‌توجهی از اثربخشی این رویکرد ارائه می‌کند، که بهبود عملکرد قابل توجهی را در معیارهای مختلف و توانایی باز کردن قابلیت‌های استدلال صفر شات در مدل‌های هوش مصنوعی نشان می‌دهد.

### 5-1) Conclusion

بخش 7 مقاله اهمیت و اثربخشی فاین تیون آموزش در زمینه مدل‌های زبانی را مورد بحث قرار می‌دهد. در اینجا به تفصیل بحث در چهار پاراگراف آمده است:

اثربخشی در فاین تیون زنجیره‌ای از فکر (CoT):

این مقاله نقش حیاتی فاین تیون CoT را در بهبود توانایی‌های استدلال مدل‌های زبانی برجسته می‌کند. در حالی که مطالعات قبلی نشان داده‌اند که فاین تیون در وظایف غیر CoT می‌تواند عملکرد را در کارهای غیرقابل مشاهده مشابه افزایش دهد، تمایل به بدتر شدن عملکرد در وظایف CoT دارد. برای مقابله با این، این مقاله یک رویکرد فاین تیون مشترک در داده‌های غیر CoT و CoT را پیشنهاد می‌کند که به طور قابل توجهی عملکرد در وظایف CoT را افزایش می‌دهد و در عین حال دستاوردهای وظایف غیر CoT را حفظ می‌کند. این رویکرد جامع به یک مدل واحد اجازه می‌دهد تا در مجموعه‌ای از ارزیابی‌های متنوع برتری یابد و استحکام فاین تیون CoT را در افزایش قابلیت‌های مدل تأیید کند.

تعمیم در بین مدل ها:

این مطالعه کاربرد فاین تیون instruction را در طیف متنوعی از مدل ها گسترش می دهد و سازگاری و اثربخشی آن را نشان می دهد. این تحقیق شامل مدل‌هایی با معماری‌ها، اندازه‌ها و اهداف پیش‌آموزشی مختلف است که کاربرد جهانی فاین تیون دستورالعمل‌ها را تأیید می‌کند. نتایج نشان‌دهنده افزایش مداوم عملکرد مدل، همسو با مطالعات قبلی است. علاوه بر این، این مقاله Flan-U-PaLM را

معرفی می‌کند، مدلی که فاین تیون دستورالعمل را با سایر تکنیک‌های انطباق مدل مانند UL2R ادغام می‌کند و آن را به عنوان قوی‌ترین مدل توسعه‌یافته در این مطالعه نشان می‌دهد.

قابلیت استفاده و کاهش مضرات:

فاین تیون دستورالعمل نه تنها برای افزایش قابلیت استفاده از مدل‌های زبان، بلکه برای کاهش آسیب‌های احتمالی نیز شناخته شده است. این مقاله چالش‌های مرتبط با استفاده مستقیم از یک ایست بازرسی از پیش آموزش دیده را مورد بحث قرار می‌دهد، مانند ناتوانی مدل در دانستن زمان توقف تولید متن، که منجر به خروجی‌های نامطلوب می‌شود. با این حال، فاین تیون دستورالعمل به طور قابل توجهی عملکرد مدل را بهبود می بخشد، به ویژه در وظایف استدلال، برنامه ریزی، و توضیح پیچیده، همانطور که توسط رتبه بندی های انسانی بهبود یافته مشهود است. این مقاله بر همسویی مدل‌های تنظیم‌شده با ترجیحات انسانی تأکید می‌کند، ویژگی‌ای که برای پذیرش گسترده‌تر و استفاده مسئولانه از مدل‌های زبانی حیاتی است.

کارایی محاسباتی:

در حالی که مقیاس بندی اندازه مدل های زبان به طور کلی عملکرد را افزایش می دهد، نیاز به منابع محاسباتی قابل توجهی دارد. فاین تیون دستورالعمل راه حلی را با افزایش عملکرد مدل با حداقل سربار محاسباتی ارائه می دهد. به عنوان مثال، فاین تیون مدل PaLM 540B تنها به کسری از محاسبات پیش از آموزش نیاز دارد، اما منجر به بهبود قابل توجهی در معیارهای ارزیابی می شود. این مقاله همچنین خاطرنشان می‌کند که مدل‌های کوچک‌تر که از فاین تیون دستورالعمل استفاده می‌کنند، می‌توانند از مدل‌های بزرگ‌تر و غیرتنظیم‌تر بهتر عمل کنند و بر کارایی و اثربخشی این رویکرد تأکید می‌شود. در پرتو این یافته‌ها، این مقاله از اتخاذ فاین تیون دستورالعمل در تمامی مدل‌های زبانی از پیش آموزش‌دیده با در نظر گرفتن مزایای گسترده و حداقل نیازهای محاسباتی آن حمایت می‌کند.

## 2) ارزیابی مدل FLAN-T5

در این بخش هدف ارزیابی مدل Flan T5 large با 3 روش مختلف پرامپت میباشد.

### 2-1) لود دیتاست و بررسی آن

به کمک کد زیر دیتاست آموزش را لود میکنیم.

```
# Load dataset train and val

dataset = load_dataset("tasksource/bigbench", "sports_understanding")
```

به کمک کد زیر نیز دیتاست تست را لود میکنیم.

```
# Load dataset test
dataset_test = load_dataset("lukaemon/bbh", "sports_understanding")
```

نگاهی به دیتاست ترین بیندازیم :

```
DatasetDict({
    train: Dataset({
        features: ['inputs', 'targets', 'multiple_choice_targets',
'multiple_choice_scores', 'idx'],
        num_rows: 789
    })
    validation: Dataset({
        features: ['inputs', 'targets', 'multiple_choice_targets',
'multiple_choice_scores', 'idx'],
        num_rows: 197
    })
})
```

همچنین یک نمونه از داده های آموزش به صورت زیر میباشد:

```
dataset["train"][0]{
'}inputs': 'Determine whether the following statement or statements
are plausible or implausible:\nStatement: Trevor Bauer swung to
protect the runner in the World Series\nPlausible/implausible,'?
' targets': ['plausible'],
' multiple_choice_targets': ['plausible', 'implausible'],
' multiple_choice_scores,[0 ,1] :'
 'idx': 0}
```

همانطور که مشاهده میشود target ها در اینجا plausible و implausible میباشد.

حال نگاهی به دیتاست تست بیاندازیم.

```
DatasetDict({
    test: Dataset({
        features: ['input', 'target'],
        num_rows: 250
    })
})

dataset_test["test"][0] :
{'input': 'Is the following sentence plausible? "Elias Lindholm beat the
buzzer."',
 'target': 'no'}
```

همانطور که مشاهده میکنید خروجی در اینجا yes و no میباشد.

### 2-2) لود کردن مدل

برای لود کردن مدل از دستورات زیر استفاده میشود.

```
# Load Model and Tokenizer

tokenizer = T5Tokenizer.from_pretrained("google/flan-t5-large")
model = T5ForConditionalGeneration.from_pretrained("google/flan-t5-large")
```

### 2-3) روش Answer-only

در این روش مستقیم سوال را از مدل پرسیده و از آن میخواهیم تا با yes و no پاسخ دهد و دقت را با توجه به لیبل آن محاسبه میکنیم.

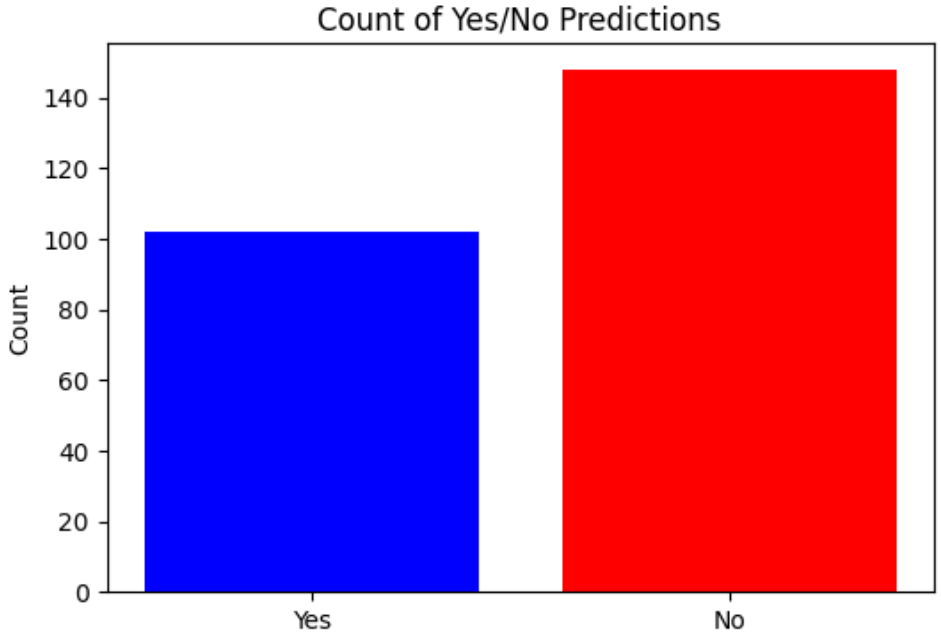
با توجه به درخواست bar plot و سیو کردن خروجی ها در csv تمامی بخش ها در کد زیر آورده شده است.

پرامپ در کد زیر مشخص میباشد.

```
def predict_yes_no(dataset_test):
    predictions = []

    for i in tqdm(range(len(dataset_test["test"]))):
        # Extract the input question
        question = dataset_test["test"][i]['input']
        # Structuring the prompt more explicitly
        formatted_input = f"Given the statement: {question} Is the statement true or false? Provide 'yes' for true and 'no' for false."

        # Tokenize and generate prediction
        inputs = tokenizer(formatted_input, return_tensors="pt", padding=True, truncation=True, max_length=512)
        outputs = model.generate(inputs['input_ids'], max_length=128)
        raw_prediction = tokenizer.decode(outputs[0], skip_special_tokens=True)

        # Post-process to get a yes/no answer
        prediction = "yes" if "yes" in raw_prediction.lower() else "no"
        predictions.append(prediction)

    # Actual labels
    actual_labels = [dataset_test["test"][i]['target'] for i in range(len(dataset_test["test"]))]

    # Save predictions and actual targets to a CSV file
```

```
    results_df = pd.DataFrame({'Input': [dataset_test["test"][i]['input'] for i
in range(len(dataset_test["test"]))],
                               'Actual Target': actual_labels,
                               'Predicted': predictions})
    results_df.to_csv('model_predictions_yes_no.csv', index=False)

    # Calculate accuracy
    accuracy = accuracy_score(actual_labels, predictions)
    print(f"Accuracy: {accuracy}")

    # Count the occurrences of 'yes' and 'no'
    yes_count = predictions.count('yes')
    no_count = predictions.count('no')

    # Create a bar plot for the counts of 'yes' and 'no' predictions
    plt.figure(figsize=(6, 4))
    plt.bar(['Yes', 'No'], [yes_count, no_count], color=['blue', 'red'])
    plt.ylabel('Count')
    plt.title('Count of Yes/No Predictions')
    plt.show()

    return results_df, accuracy

# Call the function with your dataset
results_df, accuracy = predict_yes_no(dataset_test)
```

دقت خروجی برابر 58.8 میشود و bar plot آن به صورت زیر میشود.




شکل 3: **Bar Plot** برای بخش **answer only**

### 3-Shot (2-4

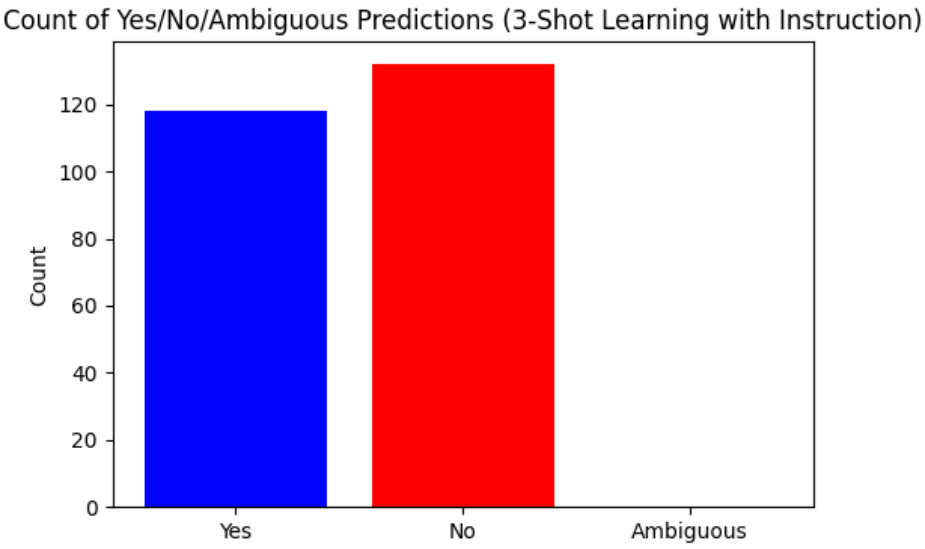
تفاوت این بخش با بخش قبلی ان است که ابتدا 3 سوال از مجموعه داده آموزشی به مدل به صورت رندوم میدهیم و میگوییم حال به سوال پاسخ بده با توجه به هینکه خروجی ها در آموزش plausible و implausible میباشد یک mapping هم صورت میگیرد تا به yes/no تبدیل شود. ( با توجه به اینکه در کد های اولیه یک سری جواب که yes و no نداشت هم وجود داشت ستون ambitious اضافه شده بود اما در نسخه نهایی این ستون اصلا خالی میباشد )

```
def predict_yes_no_3_shot_mapped_with_instruction(dataset_train, dataset_test):
    predictions = []
    raw_predictions = []  # Store raw predictions for inspection

    # Select three random examples from the training set
    example_indices = random.sample(range(len(dataset_train["train"])), 3)
    examples = [dataset_train["train"][i] for i in example_indices]

    for i in tqdm(range(len(dataset_test["test"]))):
        # Construct the 3-shot prompt with examples
        prompt = ""
        for example in examples:
            statement = example['inputs'].replace('\n', ' ')
            answer = "yes" if
example['multiple_choice_targets'][example['multiple_choice_scores'].index(1)] ==
'plausible' else "no"
```

```
            prompt += f"Statement: {statement} Answer with 'yes' or 'no':
{answer}\n"

        # Add the current test question to the prompt with instruction to answer
with 'yes' or 'no'
        question = dataset_test["test"][i]['input']
        prompt += f"Statement: {question} Answer with 'yes' or 'no':"

        # Tokenize and generate prediction
        inputs = tokenizer(prompt, return_tensors="pt", padding=True,
truncation=True, max_length=512)
        outputs = model.generate(inputs['input_ids'], max_length=128)
        raw_prediction = tokenizer.decode(outputs[0], skip_special_tokens=True)
        raw_predictions.append(raw_prediction)  # Store the raw prediction

        # Post-process to get a yes/no answer
        if "yes" in raw_prediction.lower() and "no" not in
raw_prediction.lower():
            prediction = "yes"
        elif "no" in raw_prediction.lower() and "yes" not in
raw_prediction.lower():
            prediction = "no"
        else:
            # If both or neither are present, or if it's ambiguous, you might
need a more sophisticated method
            prediction = "ambiguous"
        predictions.append(prediction)

    # Check some raw predictions
    for i in range(3):
        print(f"Raw prediction example {i+1}: {raw_predictions[i]}")

    # Actual labels
    actual_labels = [dataset_test["test"][i]['target'] for i in
range(len(dataset_test["test"]))]

    # Save predictions and actual targets to a CSV file
    results_df = pd.DataFrame({'Input': [dataset_test["test"][i]['input'] for i
in range(len(dataset_test["test"]))],
                               'Actual Target': actual_labels,
                               'Predicted': predictions})
    results_df.to_csv('model_predictions_yes_no_3_shot_mapped_with_instruction.cs
v', index=False)

    # Calculate accuracy (excluding ambiguous cases)
    accurate_predictions = [pred for pred, actual in zip(predictions,
actual_labels) if pred != "ambiguous"]
```

```
    accurate_labels = [actual for pred, actual in zip(predictions, actual_labels)
if pred != "ambiguous"]
    accuracy = accuracy_score(accurate_labels, accurate_predictions) if
accurate_predictions else 0
    print(f"Accuracy (excluding ambiguous cases): {accuracy}")

    # Count the occurrences of 'yes', 'no', and 'ambiguous'
    yes_count = predictions.count('yes')
    no_count = predictions.count('no')
    ambiguous_count = predictions.count('ambiguous')

    # Create a bar plot for the counts of 'yes', 'no', and 'ambiguous'
predictions
    plt.figure(figsize=(6, 4))
    plt.bar(['Yes', 'No', 'Ambiguous'], [yes_count, no_count, ambiguous_count],
color=['blue', 'red', 'grey'])
    plt.ylabel('Count')
    plt.title('Count of Yes/No/Ambiguous Predictions (3-Shot Learning with
Instruction)')
    plt.show()

    return results_df, accuracy

# Call the function with your dataset
results_df_3_shot_mapped_with_instruction,
accuracy_3_shot_mapped_with_instruction =
predict_yes_no_3_shot_mapped_with_instruction(dataset, dataset_test)
```

دقت در این حالت 54 درصد که ممکن است به خاطر رندوم بودن انتخال از مجموعه داده ترین یا جهت دار کردن مدل به سمت دیگر میشود و همچنین مواردی که از معایب و مقایسه چند شات در بخش 1 مطرح شد میتواند از دلایل این مورد باشد.




شکل 4 : **Bar Plot** برای بخش **3-Shot**

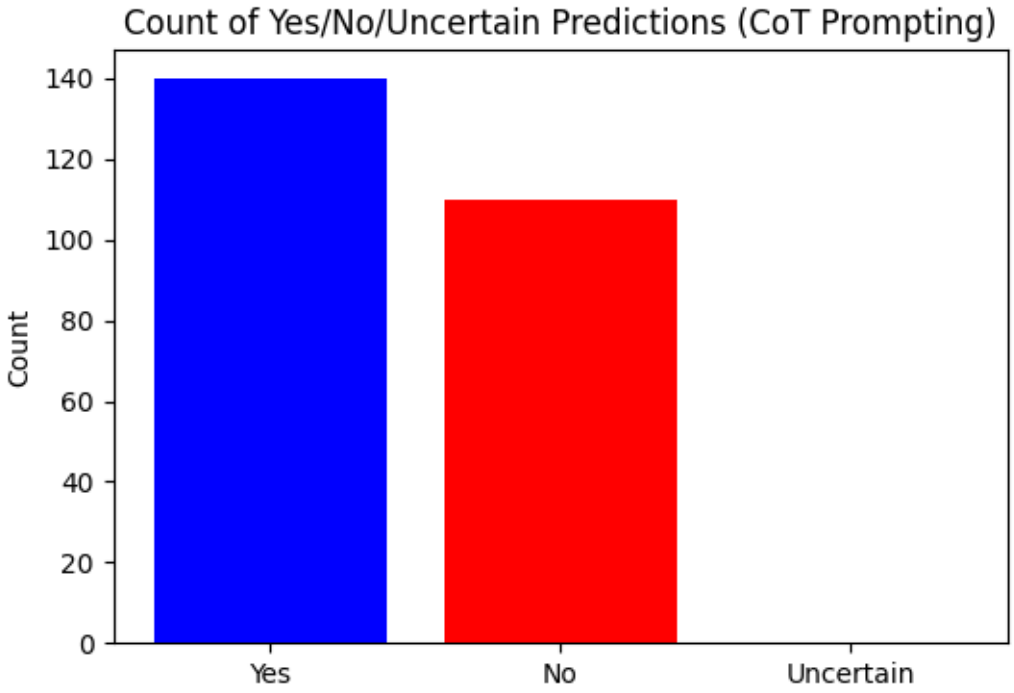
### 2-4) روش COT

این روش میگوید که مرحله به مرحله فکر کن بعد پاسخ را بگو که در نهایت به دقت 51 درضد میرسید که نشان میدهد این مدل پرامپت دادن برای این مدل مناسب نمیباشد در کل تمامی مواردی که میتواند بله یا نه تلقی شود گذاشته شده است.

```
def extract_yes_no(answer):
    # Define patterns that typically indicate a yes or no answer
    yes_pattern =
re.compile(r"\b(yes|yeah|sure|correct|indeed|right|true)\b",
re.IGNORECASE)
    no_pattern =
re.compile(r"\b(no|nah|nope|false|incorrect|wrong|not)\b", re.IGNORECASE)

    # Check for 'yes' or 'no' in the answer
    if yes_pattern.search(answer):
        return "yes"
    elif no_pattern.search(answer):
        return "no"
    else:
        return "uncertain"  # or any default value you prefer

def predict_with_cot(dataset_test):
    predictions = []
    max_length_output = 512  # Adjust as needed, but keep within the
model's limits

    for question in tqdm(dataset_test["test"]):
```

```
        cot_input = f"Answer the following yes/no question by reasoning
step-by-step: {question['input']}"
        inputs = tokenizer(cot_input, return_tensors="pt", padding=True,
truncation=True, max_length=512)

        # Generate the output with a specified maximum length
        outputs = model.generate(inputs['input_ids'],
max_length=max_length_output)

        raw_prediction = tokenizer.decode(outputs[0],
skip_special_tokens=True)

        # Extract yes/no from the prediction
        yes_no_prediction = extract_yes_no(raw_prediction)
        predictions.append(yes_no_prediction)

    # Actual labels
    actual_labels = [item['target'] for item in dataset_test["test"]]

    # Calculate accuracy
    accuracy = accuracy_score(actual_labels, predictions)

    # Save predictions and actual targets to a CSV file
    results_df = pd.DataFrame({'Input': [item['input'] for item in
dataset_test["test"]],
                               'Actual Target': actual_labels,
                               'Predicted': predictions})
    results_df.to_csv('model_predictions_cot.csv', index=False)

    # Count the occurrences of 'yes', 'no', and 'uncertain'
    yes_count = predictions.count('yes')
    no_count = predictions.count('no')
    uncertain_count = predictions.count('uncertain')

    # Create a bar plot for the counts of 'yes', 'no', and 'uncertain'
predictions
    plt.figure(figsize=(6, 4))
    plt.bar(['Yes', 'No', 'Uncertain'], [yes_count, no_count,
uncertain_count], color=['blue', 'red', 'grey'])
    plt.ylabel('Count')
    plt.title('Count of Yes/No/Uncertain Predictions (CoT Prompting)')
    plt.show()

    return results_df, accuracy

# Call the function with your dataset
results_df_cot, accuracy_cot = predict_with_cot(dataset_test)
print(f"Accuracy: {accuracy_cot}")
```

دقت در این حالت 51 و bar plot آن به صورت زیر میباشد.




شکل 5 : **Bar Plot** در بخش **COT**

تمامی فایل های csv پاسخ ها در فایل ارسالی موجود میباشد و در پوشه Q2_part2_csv میباشد.

## 3) بررسی مقاله متد و متریک انتخاب few shot

مقاله یک روش نوآورانه به نام "Active-Prompt" را معرفی می‌کند که برای بهبود عملکرد مدل‌های زبان بزرگ (LLM) در کارهای مختلف از طریق استفاده از اعلان‌های مثال خاص طراحی شده است. این اعلان‌ها به‌طور منحصربه‌فردی با استدلال زنجیره‌ای از فکر (CoT) مشروح می‌شوند. تمرکز اصلی این روش، انتخاب تأثیرگذارترین و سودمندترین سؤالات برای annotation از مجموعه ای از پرس و جوهای ویژه کار است، که یک چالش مهم است زیرا اثربخشی روش های CoT به طور سنتی به مجموعه ثابتی از نمونه های مشروح شده توسط انسان بستگی دارد، که ممکن است در همه وظایف به طور جهانی مؤثر نباشد.

Active prompt از چهار مرحله مجزا تشکیل شده است:

1.تخمین عدم قطعیت:

- مدل عدم قطعیت را در بین پیش بینی های خود برای هر سوال مشخص می کند.

- چندین معیار برای این تخمین پیشنهاد شده است، از جمله عدم توافق( disagreement ) ، آنتروپی، واریانس و self confidence.

- اختلاف نظر و آنتروپی در درجه اول به دلیل سادگی در اجرا و اثربخشی مورد استفاده قرار می گیرند.

2.انتخاب( selection ) :

- سؤالاتی با بالاترین عدم قطعیت تخمین زده شده برای annotation انتخاب می شوند.

- این مرحله بسیار مهم است زیرا مستقیماً بر کیفیت و ارتباط نمونه‌های مورد استفاده برای ارائه LLM تأثیر می‌گذارد.

3. Annotation:

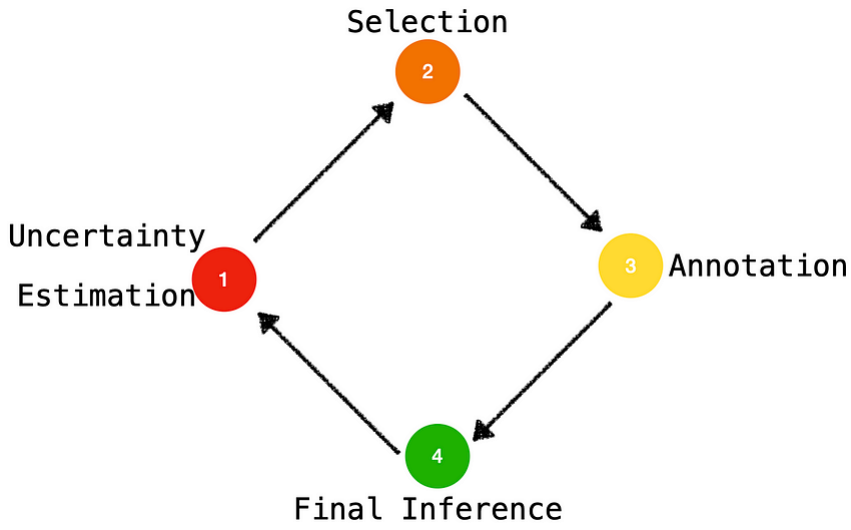
- سوالات انتخاب شده توسط annotator های انسانی با زنجیره های منطقی و پاسخ annotate می شوند.

- این فرآیند برای ایجاد نمونه های جدید متناسب با نیازهای استدلالی خاص کار حیاتی است.

4.استنتاج:

- نمونه های جدید ساخته شده در مرحله استنتاج برای تحریک LLM استفاده می شوند.

- این مرحله شامل استفاده از نمونه های جدید برای تحریک LLM با هدف استخراج پاسخ های با کیفیت بالا است.



بخش «توضیح متریک» مقاله به معیارهای مورد استفاده برای اندازه‌گیری عدم قطعیت پیش‌بینی‌های مدل می‌پردازد، بخش کلیدی روش Active-Prompt. این معیارها در شناسایی آموزنده ترین سوالات برای annotation بسیار مهم هستند. آنها عبارتند از:

1. اختلاف ( disagreement ) :

- سطح واریانس بین پیش بینی های مختلف را اندازه گیری می کند.

- سطح بالاتر اختلاف، عدم اطمینان بیشتر را نشان می دهد و مناطقی را که CoT می تواند سودمند باشد، مشخص می کند.

2. آنتروپی:

- اندازه گیری آماری غیرقابل پیش بینی بودن یا تصادفی بودن.

- مقادیر آنتروپی بالاتر نشان می دهد که پیش بینی های مدل پراکنده تر هستند، که نشان دهنده عدم قطعیت بالاتر است.

3. واریانس:

- گسترش پیش بینی های مدل را کمی می کند.

- واریانس بالاتر نشان می دهد که پیش بینی ها پراکنده تر هستند و مدل در مورد پاسخ صحیح اطمینان کمتری دارد.

4. اعتماد ( self confidence ) :

- تخمین خود مدل از اعتمادش به پاسخ هایش را نشان می دهد.

- اعتماد به نفس پایین به عدم اطمینان یا تردید در مورد درستی پاسخ اشاره می کند.

با ادغام این معیارها، روش Active-Prompt یک استراتژی ظریف و مؤثر برای انتخاب و annotation سودمندترین سؤالات ارائه می دهد. این رویکرد به طور قابل توجهی عملکرد LLM را، به ویژه در وظایف استدلالی پیچیده، با تمرکز بر مناطقی که استدلال CoT مشروح شده توسط انسان می‌تواند اساسی‌ترین پیشرفت را ارائه دهد، افزایش می‌دهد. تاکید این روش بر حاشیه نویسی مبتنی بر عدم قطعیت و استفاده استراتژیک از معیارهایی مانند اختلاف نظر و آنتروپی، آن را به یک پیشرفت قابل توجه در زمینه LLM تبدیل کرده است.

## 4) روش بهبود رویکرد COT

رویکرد زنجیره فکر (CoT) در هوش مصنوعی و یادگیری ماشینی شامل ایجاد مراحل میانی یا مسیرهای استدلالی هنگام حل مسائل پیچیده است. این روش به ویژه در افزایش شفافیت و تفسیرپذیری فرآیند تصمیم گیری مدل سودمند است.

خود سازگاری (SC) تکنیکی است که می تواند با CoT ترکیب شود. این شامل ایجاد مسیرهای استدلال یا پاسخ های متعدد برای یک مسئله معین و سپس بررسی متقاطع این پاسخ ها برای سازگاری است. منطق این است که اگر چندین مسیر استدلال مستقل به یک نتیجه منتهی شود، احتمال اینکه نتیجه درست باشد بیشتر است. این می تواند به ویژه در سناریوهایی که خروجی مدل نامشخص است یا دقت بالا حیاتی است مفید باشد.

ادغام خودسازگاری با زنجیره فکر (CoT-SC) می تواند چندین مزیت را نسبت به استفاده از CoT به تنهایی ارائه دهد:

- افزایش قابلیت اطمینان: با اعتبارسنجی متقابل مسیرهای استدلالی، خروجی های مدل می توانند قابل اعتمادتر باشند. این امر به ویژه در سناریوهای تصمیم گیری پرمخاطره مهم است.
- کاهش خطا: CoT-SC می تواند به شناسایی و کاهش خطاهایی که ممکن است در زنجیره های استدلال فردی ایجاد شود کمک کند.
- بینش عمیق تر: این رویکرد می تواند بینش عمیق تری را در مورد فرآیند حل مسئله ارائه دهد، زیرا مسیرهای استدلال مختلف ممکن است جنبه های مختلف مشکل را برجسته کند.
- استحکام در برابر ابهام: در مواردی که مشکل یا داده ها مبهم هستند، داشتن زنجیره های فکری متعدد و اطمینان از سازگاری آنها می تواند به نتیجه گیری های قوی تری منجر شود.
- در حالی که CoT-SC امیدوارکننده به نظر می رسد، در نظر گرفتن سربار محاسباتی و پیچیدگی آن نیز ضروری است. ایجاد و بررسی متقاطع مسیرهای استدلال چندگانه به منابع بیشتری نیاز دارد و می تواند برای پیاده سازی و تفسیر پیچیده تر باشد. با این وجود، برای کاربردهای خاصی که دقت و قابلیت اطمینان در آنها اهمیت دارد، این مبادلات ممکن است ارزشش را داشته باشند.

## 5) منابع استفاده شده

منابع اسفاده شده برای سوال 2 :

مقالات مطرح شده در صورت سوال و سایت های زیر :

https://medium.com/@johannes.koeppern/self-consistency-with-chain-of-thought-cot-sc-2f7a1ea9f941

https://sh-tsang.medium.com/brief-review-self-consistency-improves-chain-of-thought-reasoning-in-language-models-6471ea9bc00a

https://cobusgreyling.medium.com/active-prompting-with-chain-of-thought-for-large-language-models-b9708b2ddc22

# سوال ۳ – Speech Synthesis

## 1) مقدمه

در این بخش هدف فاین تیون مدل SpeechT5 که برای مایکروسافت میباشد بر روی زبان فارسی برای تسک tts میباشد.

در پیاده سازی این سوال از سخت افزار NVIDIA 1080 12G استفاده شده است.

همچنین برای این بخش از کتابخانه های زیر استفاده میشود.

```
import os
import matplotlib.pyplot as plt
from collections import defaultdict
from typing import Any, Dict, List, Union
from dataclasses import dataclass
from IPython.display import Audio
import torch

from datasets import load_dataset, DatasetDict, Audio, concatenate_datasets
from transformers import Seq2SeqTrainingArguments, Seq2SeqTrainer
from transformers import SpeechT5Processor, SpeechT5ForTextToSpeech,
SpeechT5HifiGan
from speechbrain.pretrained import EncoderClassifier
```

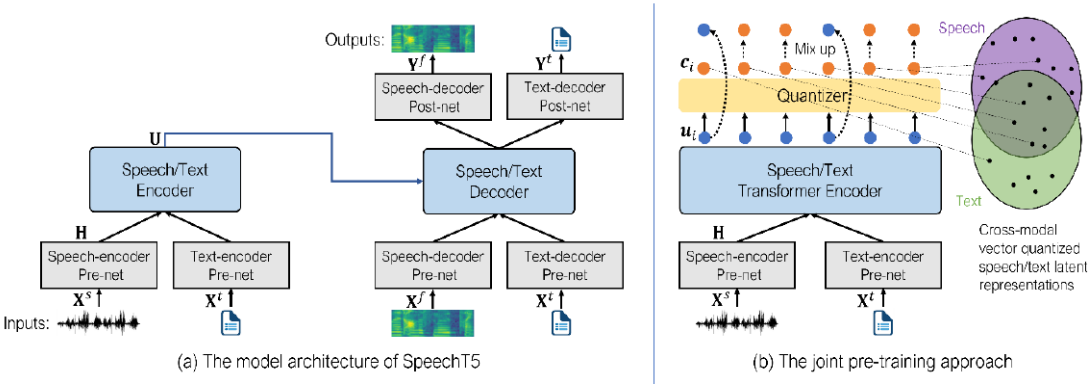
## 2) لود کردن مدل

جهت لود کردن مدل به سادگی از دستور زیر استفاده میشود.

```
processor = SpeechT5Processor.from_pretrained("microsoft/speecht5_tts")
```

```
model = SpeechT5ForTextToSpeech.from_pretrained("microsoft/speecht5_tts")
```

شکل زیر مدل کلی speecht5_tts را نشان میدهد.




شکل 6 : معماری **speech t5**

## 3) لود کردن و بررسی دیتاست

در صورت سوال خواسته شده تا ورژن 13 دیتاست common voice بررسی شود که در این بخش به جای دانلود مستقیم آن را از hugging face لود میکنیم.

```
# Load the Common Voice dataset for Persian language, version 13.0
common_voice = DatasetDict()

common voice["train"] = load dataset("mozilla-foundation/common voice 13 0", "fa",
split="train+validation", use_auth_token=True)
common_voice["test"] = load_dataset("mozilla-foundation/common_voice_13_0", "fa",
split="test", use_auth_token=True)
```

در ادامه داده های دیتاست را تماما سپمل rate آن ها را به 16000 تبدیل میکنیم. ( این بخش برای ادامه کار ضروری میباشد)

```
# Cast the 'audio' column to the Audio format with a specific sampling rate
common_voice = common_voice.cast_column("audio", Audio(sampling_rate=16000))
```

حال نگاهی به دیتاست بیاندازیم :

```
DatasetDict({
  train: Dataset({
    features: ['client_id', 'path', 'audio', 'sentence', 'up_votes', 'down_votes', 'age', 'gender',
'accent', 'locale', 'segment', 'variant'],
    num_rows: 38464
  })
  test: Dataset({
    features: ['client_id', 'path', 'audio', 'sentence', 'up_votes', 'down_votes', 'age', 'gender',
'accent', 'locale', 'segment', 'variant'],
    num_rows: 10440
  })
})
```

دیتاست شامل ترین و تست بوده و ستون های اطلاعات هر بخش مشخص میباشد.

اقدامی بعدی ترکیب داده های آموزش و تست و پردازش تمام آن ها با هم میباشد ( میتوانستیم این کار را نکنیم و در نهایت از داده های آموزش 10 درصد به عنوان داده ی ارزیابی جدا کنیم ولی با توجه به اینکه معیار ارزیابی انسانی میباشد تصمیم بر این شد که برای اضافه تر کردن داده ها و آموزش بهتر داده ها را با هم ترکیب کنیم. )

به کمک کد زیر داده ها با هم ترکیب میشوند.

```
# Merge the 'train' and 'test' splits
common_voice_merged = concatenate_datasets([common_voice["train"],
common_voice["test"]])
```

بدین ترتیب فرم نهایی دیتاست به صورت زیر میباشد.

```
Dataset({
  features: ['client_id', 'path', 'audio', 'sentence', 'up_votes', 'down_votes', 'age', 'gender', 'accent',
'locale', 'segment', 'variant'],
  num_rows: 48904
})
```

که شامل 48904 فایل صوتی تقریبا 4 ثانیه ای میباشد.

## 4) آپدیت توکنایزر و تمیز کردن داده ها

در این بخش ابتدا از processor استفاده میشود البته کد بدون استفاده از پروسسور و trainer در انتها آمده است.

مجموعه داده ممکن است شامل کاراکترهایی باشد که در واژگان tokenizer در SpeechT5 نیستند. بنابراین می‌توانیم آن را بدون تغییر معنای آن با مناسب آن جایگزین کنیم. به علاوه یک سری کاراکتر در tokenizer ممکن است تعریف نشده باشد که باید توکنایزر را آپدیت کرد.

ابتدا باید بفهمیم توکن های پشتیبانی نشده چیست. «SpeechT5Tokenizer» با کاراکترها به عنوان نشانه کار می کند، بنابراین بیایید همه کاراکترهای متمایز مجموعه داده را استخراج کنیم. ما یک تابع نگاشت «extract_all_chars» می‌نویسیم که رونویسی‌ها را از تمام مثال‌ها به یک رونویسی طولانی الحاق می‌کند و سپس رشته را به مجموعه‌ای از کاراکترها تبدیل می‌کند. و در نهایت mapping را انجام میدهیم.

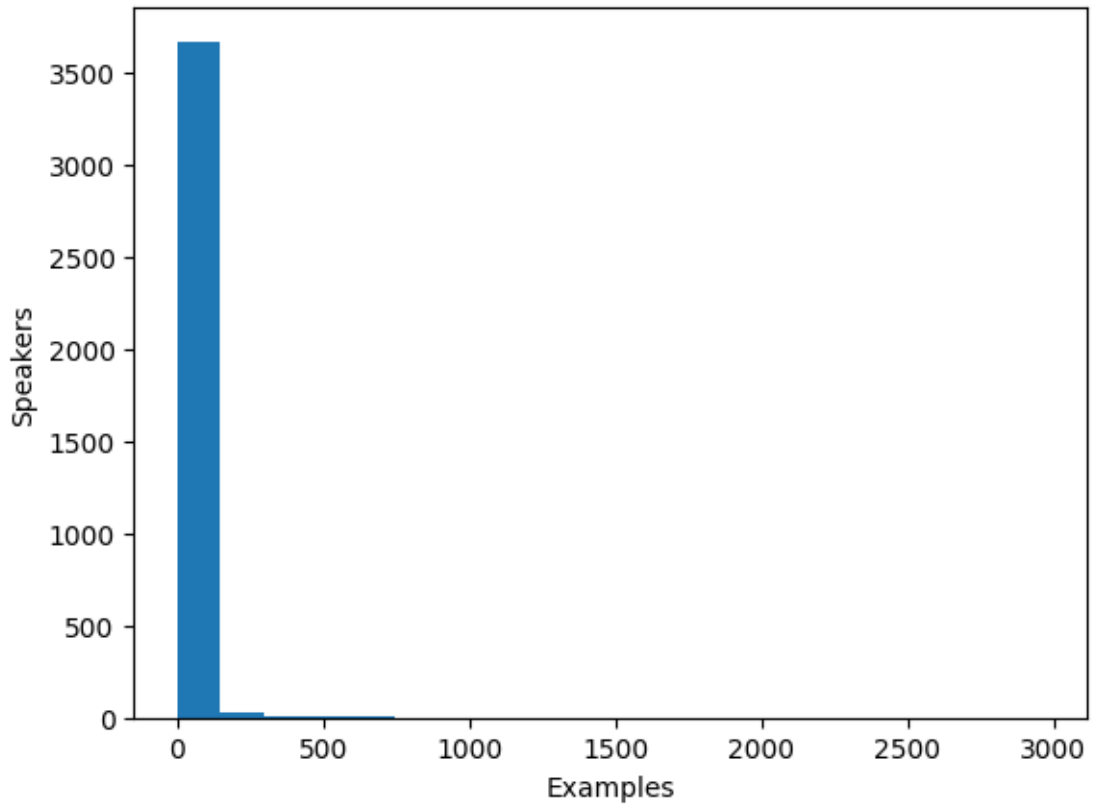
تمامی فانکشن ها و عملیات دارای کامنت بوده و واضح میباشد.

```
# Define a function to extract all unique characters from a batch of
sentences
def extract_all_chars(batch):
    all_text = " ".join(batch["sentence"])  # Combine all sentences into one
string
    vocab = list(set(all_text))  # Create a list of unique characters
    return {"vocab": [vocab], "all_text": [all_text]}

# Use the 'map' function to apply the 'extract_all_chars' function to the 'train'
split of the 'common_voice' dataset
vocabs = common_voice["train"].map(
    extract_all_chars,
    batched=True,
    batch_size=-1,
    keep_in_memory=True,
    remove_columns=common_voice_merged.column_names,  # Remove unnecessary
columns
)

# Get the unique characters from the extracted vocab
dataset_vocab = set(vocabs["vocab"][0])

# Get the unique characters from the tokenizer's vocabulary
tokenizer_vocab = {k for k, _ in tokenizer.get_vocab().items()}

persian_char = dataset_vocab - tokenizer_vocab

replacements = [
    ('š', 'ش'),
    ('ā', 'آ'),
]

def cleanup_text(inputs):
    for src, dst in replacements:
        inputs["sentence"] = inputs["sentence"].replace(src, dst)
    return inputs

common_voice_merged = common_voice_merged.map(cleanup_text)


# Extend the tokenizer's vocabulary
new_tokens = processor.tokenizer.add_tokens(list(persian_char))

# Verify that the new characters are added to the vocabulary
updated_vocab = processor.tokenizer.get_vocab()

tokenizer_vocab = {k for k, _ in tokenizer.get_vocab().items()}
dataset_vocab - tokenizer_vocab
```

بدین ترتیب فوق داده ها تمیز و کاراکترها به توکنایزر اضافه میشود.

### 5) speaker embedding

ما میدانیم که تعداد گوینده ها متفاوت میباشد و اینکه هر کدام چند نمونه از داده ها را گوینده میباشند نیز در خروجی کار اهمیت دارد.




شکل 7 : توزیع گوینده ها نمونه های دیتاست تجمیع شده **common voice**

در ادامه نیاز هست که speaker ها را محدود کنیم اما با توجه به بررسی و آموزش های انجام شده و بررسی دقت ها از 10 تا 1000 را برمیداریم.

```
def select_speaker(speaker_id):
    return 10 <= speaker_counts[speaker_id] <= 1000

common_voice_merged = common_voice_merged.filter(select_speaker,
input_columns=["client_id"])
```

که این شامل 546 گوینده میشود.

در ادامه برایبرای اینکه مدل TTS بتواند بین چند گوینده تمایز قائل شود، باید برای هر نمونه یک speaker embedding ایجاد کنیم و به سادگی یک ورودی اضافی به مدل است که ویژگی‌های صدای یک بلندگوی خاص را ثبت می‌کند.

برای این کار از spkrec-xvect-voxceleb از speech brain استفاده شده است. که یک شکل موج صوتی ورودی را می گیرد و یک بردار 512 عنصری حاوی جاسازی بلندگوی مربوطه را خروجی می دهد.

```
# Determine the device
device = "cuda" if torch.cuda.is_available() else "cpu"

# Load a pretrained speaker recognition model from SpeechBrain
speaker_model = EncoderClassifier.from_hparams(
    source="speechbrain/spkrec-xvect-voxceleb",
    run_opts={"device": device},
    savedir=os.path.join("/tmp", "speechbrain/spkrec-xvect-voxceleb")
)

# Define a function to create speaker embeddings from a given waveform
def create_speaker_embedding(waveform):
    with torch.no_grad():
        # Encode the input waveform to obtain speaker embeddings
        speaker_embeddings = speaker_model.encode_batch(torch.tensor(waveform))

        # Normalize the speaker embeddings along the embedding dimension (L2
normalization)
        speaker_embeddings = torch.nn.functional.normalize(speaker_embeddings,
dim=2)

        # Squeeze the tensor to remove any unnecessary dimensions and move it to
the CPU
        speaker_embeddings = speaker_embeddings.squeeze().cpu().numpy()

    return speaker_embeddings
```

## 6) نگاشت نهایی و بررسی نهایی پیش پردازش دیتاست

در نهایت نیز نگاشت انجام میشود که در اینجا از processor استفاده میشود و نسخه بدون استفاده از processor در انتها آمده است.

```
def prepare_dataset(example):
    # load the audio data
    audio = example["audio"]
```

```
    # feature extraction and tokenization
    example = processor(
        text=example["sentence"],
        audio_target=audio["array"],
        sampling_rate=audio["sampling_rate"],
        return_attention_mask=False,
    )

    # strip off the batch dimension
    example["labels"] = example["labels"][0]

    # use SpeechBrain to obtain x-vector
    example["speaker_embeddings"] = create_speaker_embedding(audio["array"])

    return example
```

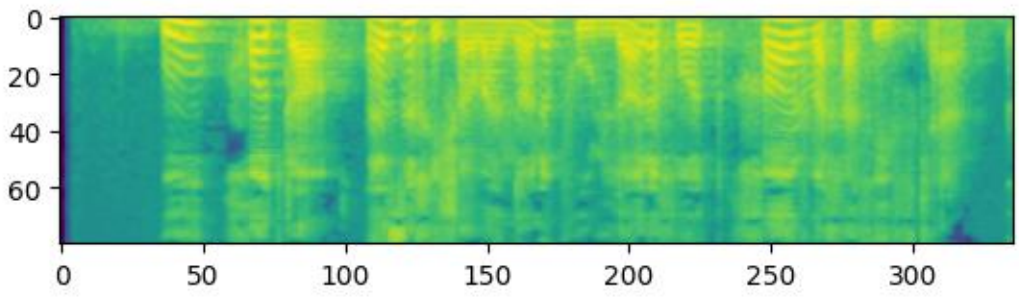
```
### Map Entire Dataset

common_voice_merged = common_voice_merged.map(
    prepare_dataset, remove_columns=common_voice_merged.column_names,
)
```

برای آنکه مطمین شویم تبدیلات و نگاشت ها و ... درست میباشد مراحل را یک دور از به هم تبدیل میکنیم تا از صحت کار پردازش انجام شده مطمئن شویم :

یک داده را process کرده و decode میکنیم و spectrum آن را میگیریم و سپس از مدل HfiGAN برای تبدیل لیبل به صوت استفاده میکنیم.

```
processed_example = prepare_dataset(common_voice_merged[0])

plt.figure()
plt.imshow(processed_example["labels"].T)
plt.show()

vocoder = SpeechT5HifiGan.from_pretrained("microsoft/speecht5_hifigan")

spectrogram = torch.tensor(processed_example["labels"])
with torch.no_grad():
    speech = vocoder(spectrogram)

from IPython.display import Audio
Audio(speech.cpu().numpy(), rate=16000)
```

Spectrum فوق به صورت زیر میباشد و فایل صوتی اجرایی نیز مناسب و درست است و بدین ترتیب کار پیش پردازش داده ها تمام است.




شکل 8 : **spectrum** فایل صوتی اولین داده مجموعه **merge dataset**

در ادامه با توجه به محدودتت توکن در T5 که برابر 600 میباشد باید مواردی که تعداد توکن بیشتر از 600 دارند را حذف کنیم.

```
# 600 token is for T5 filter length:

def is_not_too_long(input_ids):
    input_length = len(input_ids)
    return input_length < 600

common_voice_merged = common_voice_merged.filter(is_not_too_long,
input_columns=["input_ids"])
```

پس از فیلتر فوق در کل 29153 تا از داده ها باقی میماند.

## 7) تقسیم داده ها به داده های آموزش و ارزیابی و data collection

ابتدا به نسبت 90 10 داده ها را تقسیم میکنیم.

```
common_voice_merged = common_voice_merged.train_test_split(test_size=0.1)
```

فرم نهایی دیتاست به صورت زیر میشود.

```
DatasetDict({
    train: Dataset({
        features: ['input_ids', 'labels', 'speaker_embeddings'],
        num_rows: 26237
    })
    test: Dataset({
```

```
        features: ['input_ids', 'labels', 'speaker_embeddings'],
        num_rows: 2916
    })
})
```

که شامل 26237 داده برای ترین و 26237 داده برای ارزیابی میباشد. ( هر داده 4 ثانیه میباشد بنابراین مدت زمان دیتای تحت آزمایش 1750 دقیقه میباشد)

در ادامه یک data collector تعریف کنیم تا چندین نمونه را در یک دسته ترکیب کنیم. برای برچسب‌های spectrum ، قسمت‌های padd با مقدار -100 جایگزین می‌شوند. این مقدار ویژه به مدل می‌گوید هنگام محاسبه spectrum loss، آن قسمت از spectrum را نادیده بگیرد.

```
@dataclass
class TTSDataCollatorWithPadding:
    processor: Any

    def __call__(self, features: List[Dict[str, Union[List[int], torch.Tensor]]])
-> Dict[str, torch.Tensor]:

        input_ids = [{"input_ids": feature["input_ids"]} for feature in features]
        label_features = [{"input_values": feature["labels"]} for feature in
features]
        speaker_features = [feature["speaker_embeddings"] for feature in
features]

        # collate the inputs and targets into a batch
        batch = processor.pad(
            input_ids=input_ids,
            labels=label_features,
            return_tensors="pt",
        )

        # replace padding with -100 to ignore loss correctly
        batch["labels"] = batch["labels"].masked_fill(
            batch.decoder_attention_mask.unsqueeze(-1).ne(1), -100
        )

        # not used during fine-tuning
        del batch["decoder_attention_mask"]

        # round down target lengths to multiple of reduction factor
        if model.config.reduction_factor > 1:
            target_lengths = torch.tensor([
                len(feature["input_values"]) for feature in label_features
            ])
            target_lengths = target_lengths.new([
```

```
                length - length % model.config.reduction_factor for length in
target_lengths
            ])
            max_length = max(target_lengths)
            batch["labels"] = batch["labels"][:, :max_length]

        # also add in the speaker embeddings
        batch["speaker_embeddings"] = torch.tensor(speaker_features)

        return batch


data_collator = TTSDataCollatorWithPadding(processor=processor)
```

## 8) توضیح اجمالی در مورد processor و نقش آن

در این قطعه کد تا اینجا که برای آماده‌سازی داده‌های ورودی برای مدل SpeechT5، مدلی که می‌تواند برای کارهای تبدیل متن به گفتار استفاده شود، استفاده می‌شود. processor مسئول چندین مرحله پیش پردازش مهم است:

1. استخراج ویژگی: داده های صوتی خام و داده های متنی را به فرمتی مناسب برای مدل تبدیل می کند. برای داده‌های صوتی، ممکن است شامل عملیات‌هایی مانند نمونه‌برداری مجدد، استخراج ویژگی‌های Mel-spectrogram یا اعمال تغییرات دیگری باشد که مدل انتظار دارد. برای داده های متنی، متن را نشانه گذاری می کند و آن را به دنباله ای از نشانه ها یا شناسه هایی تبدیل می کند که مدل بتواند آن را بفهمد.

2. پردازش دسته ای : processor همچنین قادر به مدیریت دسته ای از داده ها است. این به این معنی است که می توانید چندین نمونه را به طور همزمان به processor وارد کنید و آنها را به صورت موازی پردازش می کند که برای آموزش مدل ها در مجموعه داده های بزرگ کارآمد و ضروری است.

3. Attention Mask :Attention Mask یک مفهوم مهم در مدل هایی است که از مکانیسم های توجه مانند ترانسفورمر ها استفاده می کنند. ماسک توجه یک ماسک باینری است (تانسور 0 و 1) که نشان می‌دهد به کدام نشانه‌ها در دنباله ورودی باید توجه شود و کدام‌ها باید نادیده گرفته شوند. این امر به ویژه برای مدیریت توالی های با طول متغیر در پردازش دسته ای مهم است. در زمینه مدل‌های تبدیل متن به گفتار، ماسک‌های توجه می‌توانند اطمینان حاصل کنند که مدل به نشانه‌های padding یا سایر بخش‌های نامربوط ورودی توجه نمی‌کند.

در کد موجود ، «return_attention_mask=False» در فراخوانی پردازنده تنظیم شده است، که نشان می‌دهد ماسک‌های توجه برای داده‌های پردازش شده برگردانده نمی‌شوند. با این حال، بعداً در کد، از ماسک های توجه در هنگام ایجاد داده های دسته ای استفاده می شود.

- ماسک توجه همراه با برچسب ها (مقادیر هدف برای پیش بینی های مدل) استفاده می شود. به طور خاص، ماسک برای نادیده گرفتن نشانه‌های padding در محاسبهloss استفاده می‌شود. این کار با تنظیم مقادیر برچسب روی "-100" انجام می شود، جایی که ماسک توجه مقدار "0" دارد (نشان دهنده عدم توجه یا padding). به این ترتیب، loss مدل تنها بر اساس بخش‌های مربوطه از داده‌های ورودی، بدون توجه به قطعات بالشتک‌شده محاسبه می‌شود.

batch["labels"] = batch["labels"].masked_fill(batch.decoder_attention_mask.unsqueeze(-' 1).ne(1), -100)" به طور خاص برچسب ها را برای پر کردن نشانه ها با `-100 جایگزین می کند. «، بنابراین مدل می داند که این نشانه ها را در طول آموزش نادیده بگیرد.

- ماسک توجه همچنین در مدیریت طول‌های متغیر دنباله‌های ورودی نقش دارد و اطمینان حاصل می‌کند که مکانیسم توجه مدل فقط بر روی بخش‌های معنی‌دار ورودی تمرکز می‌کند، نه قسمت‌های padd شده.

## 8) پیاده سازی بدون استفاده از processor

اگر بخواهیم از proceesor استفاده نکنیم و بیاییم از feature extractor مستقیم استفاده کنیم مراحل به صورت زیر میباشد.

ابتدا باید feature extractor را لود کنیم.

```
from transformers import SpeechT5FeatureExtractor, SpeechT5Tokenizer

# Initialize the feature extractor and tokenizer
feature_extractor =
SpeechT5FeatureExtractor.from_pretrained("microsoft/speecht5_tts")
tokenizer = SpeechT5Tokenizer.from_pretrained("microsoft/speecht5_tts")
```

در ادامه باقی بخش های پیش پردازشی ثابت میماند ولی در هنگام مپ کردن پروسه به صورت زیر میشود.

```
def prepare_dataset(example):
    # load the audio data
    audio = example["audio"]
```

```
    # Extract features for audio
    input_values = feature_extractor(audio["array"],
sampling_rate=audio["sampling_rate"]).input_values

    # Tokenize the text
    input_ids = tokenizer(example["sentence"]).input_ids

    # Use SpeechBrain to obtain x-vector
    speaker_embeddings = create_speaker_embedding(audio["array"])

    return {
        "input_ids": input_ids,
        "input_values": input_values,
        "speaker_embeddings": speaker_embeddings
    }
```

سایر توابع هیچ تغییری نمیکند و عملا این کار به راحتی انجام میشود و میتوان بدون استفاده از processor تمامی بخش های ما قبل آموزش مدل را انجام داد. فرمت خروجی دیتاست در این حالت به صورت زیر میباشد ( قبل از تقسیم به 2 بخش آموزش و تست )

```
DatasetDict({
    train: Dataset({
        features: ['input_ids', 'labels', 'speaker_embeddings'],
        num_rows: 26237
    })
    test: Dataset({
        features: ['input_ids', 'labels', 'speaker_embeddings'],
        num_rows: 2916
    })
})
```

که نشان میدهد پیاده سازی بدون استفاده از processor هم به خوبی انجام شده است و همه چیز درست میباشد.

## 9) آموزش مدل با trainer

در این بخش ابتدا فرایند آموزش مدل با ترینر بیان میشود و در بخش قبل بدون استفاده از trainer مدل آموزش داده میشود.

قبل از هر چیز با توجه به اینکه tokenizer را با کاراکتر های فارسی update کردیم باید طول توکن embedding مدل را resize کنیم این کار به کمک کد زیر انجام میشود.

```
# Resize Model
model.resize_token_embeddings(len(tokenizer))
```

در ادامه به راحتی آرگمان های مورد نیاز برای ترین را به مدل میدهیم و مدل را آموزش میدهیم.

```
training_args = Seq2SeqTrainingArguments(
    output_dir="./result",
    per_device_train_batch_size=16,
    gradient_accumulation_steps=2,
    learning_rate=1e-4,
    warmup_steps=500,
    max_steps=15000,
    gradient_checkpointing=True,
    fp16=True,
    evaluation_strategy="steps",
    per_device_eval_batch_size=8,
    save_steps=1000,
    eval_steps=500,
    logging_steps=25,
    report_to=["tensorboard"],
    load_best_model_at_end=True,
    greater_is_better=False,
    label_names=["labels"],
    push_to_hub=False,
)
```

مقدار lr به صورت تجربی و پس از چندین آموزش بر روی 1e-4 قرار داده شد ( البته 5e-4 هم نتیجه نسبتا مطلوبی تولید میکرد)

تعداد step ابتدا برابر 7000 و سپس برابر 15000 قرار داده شده است که در همان 7000 هم خروجی به وویس مطلوبی میرسید ولی مدت زمان ترین را برای خروجی واضح تر و بهتر بیشتر کردیم اولی 1.5 ساعت و دومی 3 ساعت بر روی gpu 1080 با رم 12 گیگ طول کشید.

در نهایت به کمک آرگمان های فوق trainer را تشکیل داده و train میکنیم.

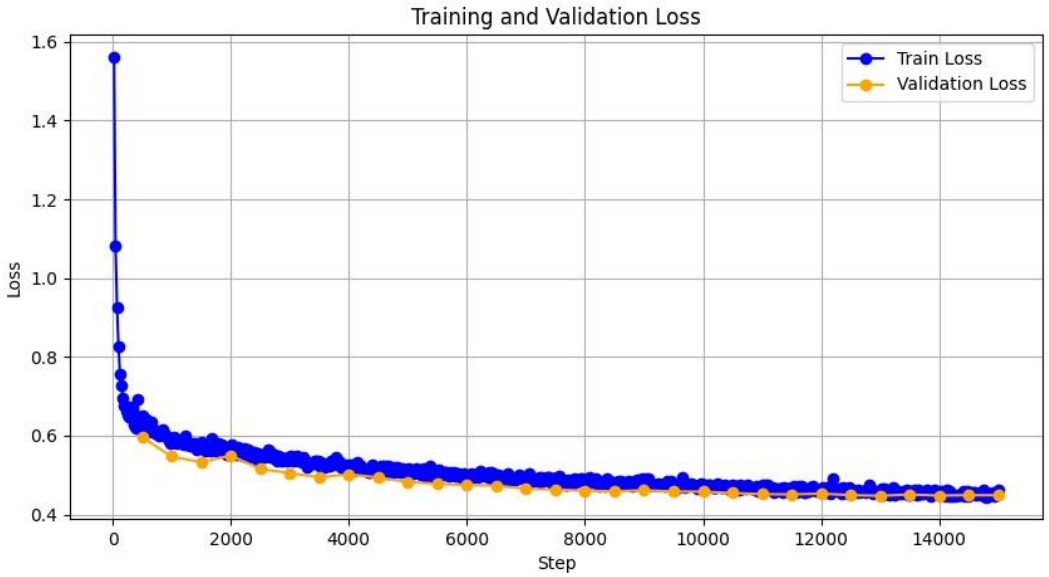
```
trainer = Seq2SeqTrainer(
    args=training_args,
    model=model,
    train_dataset=common_voice_merged["train"],
    eval_dataset=common_voice_merged["test"],
    data_collator=data_collator,
```

```
    tokenizer=processor.tokenizer,
)

trainer.train()
```

در ادامه پس از ذخیره سازی های log های مدل loss های آموزش و ارزیابی را به کمک tensor board رسم میکنیم. دیتا های log ذخیره شده در پیوست ارسال شده است.

به کمک کد زیر منحنی های loss را رسم میکنیم.

```
import pandas as pd
import matplotlib.pyplot as plt

# Load the training and validation CSV files
train_data = pd.read_csv('train_Q3_DGM.csv')
val_data = pd.read_csv('val_Q3_DGM.csv')

# Extract the relevant columns
train_steps = train_data['Step']
train_loss = train_data['Value']
val_steps = val_data['Step']
val_loss = val_data['Value']

# Create a plot
plt.figure(figsize=(10, 5))
plt.plot(train_steps, train_loss, label='Train Loss', marker='o', linestyle='-',
color='blue')
plt.plot(val_steps, val_loss, label='Validation Loss', marker='o', linestyle='-',
color='orange')
plt.xlabel('Step')
plt.ylabel('Loss')
plt.title('Training and Validation Loss')
plt.legend()
plt.grid(True)

# Save the plot as an image
plt.savefig('loss_plot.png')

# Show the plot (optional)
plt.show()
```




شکل **9** : منحنی **loss** مدل **speech t5** بر روی **coommon voice** فارسی

## 10) آموزش مدل بدون استفاده از trainer

در بخش قبل جهت آموزش از trainer استفاده شد حال در این بخش میخواهیم همان فرایند را بدون استفاده از trainer پیاده سازی کنیم.

ابتدا نیاز است که یک دیتالودر تشکیل دهیم.

```
from torch.utils.data import DataLoader

train_loader = DataLoader(common_voice_merged["train"], batch_size=16,
shuffle=True, collate_fn=data_collator)
eval_loader = DataLoader(common_voice_merged["test"], batch_size=8,
shuffle=False, collate_fn=data_collator)
```

مقدار 16 و 8 در training arg هم همین بود.

در ادامه سایر فرایند ترین انجام میشود و optimizer ها تعریف میشود.

```
import torch
from transformers import AdamW, get_linear_schedule_with_warmup

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model.to(device)

optimizer = AdamW(model.parameters(), lr=1e-4)
```

```
num_training_steps = 15000
scheduler = get_linear_schedule_with_warmup(optimizer,
num_warmup_steps=500, num_training_steps=num_training_steps)
```

پس از آن لوپ ترین نوشته میشود همانطور که میدانید لاس در خود output میباشد.

```
model.train()
num_epoch = 3
for epoch in range(num_epochs):  # Define num_epochs based on your
preference or calculations
    for batch in train_loader:
        batch = {k: v.to(device) for k, v in batch.items()}
        outputs = model(**batch)
        loss = outputs.loss
        loss.backward()

        optimizer.step()
        scheduler.step()
        optimizer.zero_grad()

        # Log, print, or save the loss value
        print(f"Loss: {loss.item()}")
```

به همین ترتیب برای لوپ ارزیابی داریم :

```
model.eval()
total_eval_loss = 0
for batch in eval_loader:
    batch = {k: v.to(device) for k, v in batch.items()}

    with torch.no_grad():
        outputs = model(**batch)

    loss = outputs.loss
    total_eval_loss += loss.item()

# Calculate the average loss over all of the batches.
avg_eval_loss = total_eval_loss / len(eval_loader)
print(f"Validation Loss: {avg_eval_loss}")
```

پس از آموزش مدل با این شرایط مقدار نهایی loss برای train به مقدار 0.48 و برای داده های ارزیابی به مقداری 0.51 رسید که مانند آموزش قبل نشد کیفیت و مقادیر لاس و احتمالا نیاز به یک سری

scheduler ها و یک سری optimization ها و یا به عنوان مثال Gradient Clipping میباشد که trainer به طور اتومات در نظر میگیرد ولی در اینجا در نظر گرفته نشده است.

## 11) ارزیابی و گرفتن خروجی از مدل

برای گرفتن خروجی از مدل ابتدا نیاز است که یک speaker embedding برداریم ( باید توجه داشت که بخشی از دقت و کیفیت خروجی به این که کدام انتخاب شود هم بستگی دارد. )

```
# obtain a speaker embedding. We can simply grab one from the test set.
example = common_voice_merged["test"][304]
speaker_embeddings =
torch.tensor(example["speaker_embeddings"]).unsqueeze(0)
```

در ادامه به کمک کد زیر یک text را نوشته آن را به tokenizer میدهیم و spectrum آن را از مدل گرفته و به کمک hifigan همانطور که قبلا توضیح داده شد آن را به ووپس تبدیل میکنیم و در نهایت خروجی را ذخیره میکنیم.

```
text = "این یک تست میباشد"
tokenizer.decode(tokenizer(text)["input_ids"])
inputs = processor(text=text, return_tensors="pt")
spectrogram = model.generate_speech(inputs["input_ids"],
speaker_embeddings)
with torch.no_grad():
    speech = vocoder(spectrogram)



from IPython.display import Audio
Audio(speech.cpu().numpy(), rate=16000)

import soundfile as sf
sf.write("output3.wav", speech.numpy(), samplerate=16000)
```

فایل های خروجی در فایل ارسالی در پوشه Q3_result_audio موجود میشود که 3 فایل output 1 تا 3 خروجی تا 7000 میباشد که کیفیت نسبتا مناسبی تنها با 1.5 ساعت آموزش مدل بر روی GPU 1080 TI 12G دارد.

## 12) منابع استفاده شده

منابع استفاده شده در سوال 3 :

https://huggingface.co/learn/audio-course/chapter6/fine-tuning

https://colab.research.google.com/drive/1i7I5pzBcU3WDFarDnzweIj4-sVVoIUFJ

با تشکر از لطف و زحمات شما