Fine-tune A GPT-3 Model To A Specific Domain

Steve Wen

August 2022

Abstract omitted

1 Introduction

Large generative language models have emerged in recent years, e.g. BERT[3], MT-NLG[8], PaLM[2], etc. Many of them have reached state-of-art results in many domains within and out of traditional NLP tasks. Their successes should largely give credit to massive amount of parameters and the attention mechanism behind, the first of which could greatly help to learn more complicated hidden functions, while the second of which could perfectly capture long-term dependencies, which is also a very essential part of the NLP tasks.

Many of these language models are firstly trained to attain generalization, deploying self-supervised learning during pre-training phase and fine-tuned afterwards for specific tasks.

By deeply exploiting these two key aspects mentioned above, GPT-3 has achieved competitive results on language tasks even without fine-tuning its 175B parameters. [1] In this work, we will look into how to fine-tune a GPT-3 model to different domains. All the works are still theoretical and no experiment has been carried out yet.

2 Background

We now present the Fine-Tune process in detail. We will first formalize the concept of Fine-Tune and then introduce the tasks we performed. Later in section 3 we will presented the engineering part of this work.

2.1 Fine-Tune

Fine-Tune is a transfer learning process performed after the pre-training phase, by providing training samples of a specific domain intended to learn a local distribution. This is often done by utilizing the gradient methods from supervised learning to do the weight update. Naturally, this process could also be done using Reinforcement Learning by maximizing the reward signal.[7] In this work, we only focus on the supervised learning update.

2.2 NLP related tasks

NLP tasks range from classification, sentiment analysis, summarization, translation, Q&A and many more. Those tasks are not specific to to the traditional ones, but could expand to other domains as well. Here we list the domains we have performed on.

- *Traditional Language Tasks*: The NLP task we perform on the traditional domain is email categorization, which is basically classify the given email in one of the predefined category. The data we used is from sklearns.
- *BioMedical NLP Tasks*: It has been pointed out that GPT-3 may not be very competitive in BioMedical domains.[6]. The NLP task we perform is the Biomedical Research Question Answering. Every round, the model is given a question and some context and the model needs to find the correct long answer to it. Due to we do not know the inner accuracy calculation mechanism of the provided model, we just followed the outcome we got from API. The data we used is from PubMedQA[4]
- *Physics NLP tasks*: The physics NLP tasks mainly focus on the model's understanding of the physical world. That often involves some commonsense questions and basic physics questions. The task we chose this time is the physics Q&A, where according to the given question, the model needs to infer the question without any given context. The data we used is from Rainbow[5]

3 Detailed Steps

Open AI has listed the detailed steps on fine-tuning the GPT-3 model. It is well built using CLI (Command Line Interface). To summarize, it could be split into 3 steps as follows.

Data Preparation

The data file format GPT-3 could accept is JSONL, in which each line is a prompt-completion pair in the format of a dictionary.

```
{"prompt": "<prompt_text>", "completion": "<ideal_generated_text>"}
...
```

Luckily, a data preparation tool is included in the package with command

```
openai tools fine_tunes.prepare_data -f <LOCAL_FILE>
```

The only requirement for the data file is containing a prompt and a completion column/key. The format could be one of CSV, TSV, XLSX, JSON or JSONL, and it will save the output into a JSONL file ready for fine-tuning, with suggested changes.

Fine-Tune the model

There are four base models could be chosen from, with the powerful davinci included.

```
openai api fine_tunes.create -t <TRAIN_FILE_ID_OR_PATH> -m <BASE_MODEL>
```

Use the model

To use the model, we could either call the CLI with

```
openai api completions.create -m <FINE_TUNED_MODEL> -p <YOUR_PROMPT>
```

or use the python API

```
openai.Completion.create(model= modelId, prompt= prt + ' \\n\\n###\\n\\n',)
```

4 Results

Here we present all the results we get from fine-tuning the GPT-3 base model ada.
Disclaimer: As the budget is limited, every training only has around 1k to 2k examples.

4.1 Traditional NLP Task Result

The model did not improve a lot over the fine-tune, but with a linear boost in the middle. This shape of the graph could be interpreted as the accuracy is round to three digits only. The accuracy might be choppy before the linear boost and afterwards, but it cannot be reflected from the given digits.

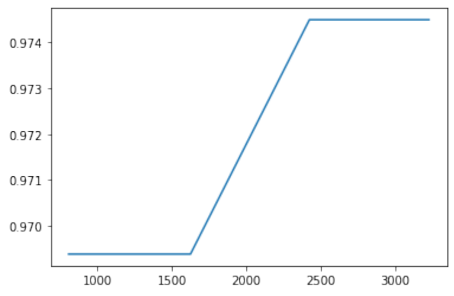

Figure 1: Classification Task on 4 categories of emails

4.2 BioMedical NLP Task Result

The accuracy of the biomedical data improved much over the training steps. The red line is the drawn regression over the steps. The the accuracy landed over 70% after fine-tuning, which indicates the GPT-3 could have great potential to do better in the BioMedical data field with a longer fine-tuning process. More research needs to be carried out to see whether this result contradicts with the conjecture [6] they have made.

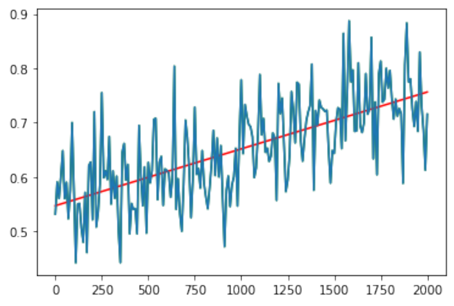

Figure 2: Biomedical Data Question Answering

4.3 Physics NLP Task Result

Although starting at lower point than the Bio Task, which would indicate that the model could hardly understand the world, it shows much stronger improvement than the previous one. It learns to adapt to the physics domain rather quickly with a slope higher than that of the biomedical one.

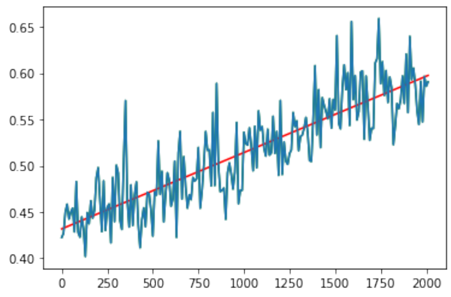

Figure 3: Physics Question Answering

5 An idea

When we were reading the GPT-3 paper, an idea hit us that among all those language tasks GPT- 3 could perform, (translation, filling the blank, etc) there is consent between us that a universal logic of language does exist, but disguised in the black box.

Languages are encrypted tokens. No matter what kind of language we are speaking, whether we use the subject as *I, Je* we all want to express similar ideas, following similar logics. In recent years, some conlangs (composition for constructed languages) appeared in fictions, like High Valyrian (GOT), Klingon (Star Trek). Those languages do not have any prototypes in real world scenario, but they do follow certain alphabets and language logic. Thus these are valid languages safe to use.

After a thorough research through the internet, I did not find many works on the field of conlang generation using these large language models. I do not know whether this is not exciting enough or it is very difficult to do the experiments, as it is hard to quantitate the metric of the loss function.

References

[1] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[2] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[4] Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William W Cohen, and Xinghua Lu. Pubmedqa: A dataset for biomedical research question answering. *arXiv preprint arXiv:1909.06146*, 2019.

[5] Nicholas Lourie, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Unicorn on rainbow: A universal commonsense reasoning model on a new multitask benchmark. *AAAI*, 2021.

[6] Milad Moradi, Kathrin Blagec, Florian Haberl, and Matthias Samwald. Gpt-3 models are poor few-shot learners in the biomedical domain. *arXiv preprint arXiv:2109.02555*, 2021.

[7] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke E. Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Francis Christiano, Jan Leike, and Ryan J. Lowe. Training language models to follow instructions with human feedback. *ArXiv*, abs/2203.02155, 2022.

[8] Shaden Smith, Mostofa Patwary, Brandon Norick, Patrick LeGresley, Samyam Rajbhandari, Jared Casper, Zhun Liu, Shrimai Prabhumoye, George Zerveas, Vijay Korthikanti, et al. Using deepspeed and megatron to train megatron-turing nlg 530b, a large-scale generative language model. *arXiv preprint arXiv:2201.11990*, 2022.

A GPT-3 Fine-Tuning Example

The following is a python demo of a fully automation Fine-Tuning a GPT-3 model through both CLI and API commands. To see more code, please visit: my github

```
import openai
from sklearn.datasets import fetch_20newsgroups
import pandas as pd
import os



# Sample classification on the Toy dataset
def basic_classification(
        cat,
        file_name,
        mod,
        prt,
        token= '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',
                        ):
    sci_dataset = fetch_20newsgroups(
        subset='train',
        shuffle= True,
        random_state= 42,
        categories= cat)

    texts = [text.strip() for text in sci_dataset['data']]
    labels = [sci_dataset.target_names[x].split('.')[-1]
              for x in sci_dataset['target']]

    df = pd.DataFrame(zip(texts, labels), columns=['prompt', 'completion'])

    f= file_name.split('.')
    assert(f[-1]== 'jsonl') # Ensure the file format is jsonl

    df.iloc[:1000].to_json(file_name, orient='records', lines=True)


    # Prepare and upload the data for fine-tuning
    os.system('openai tools fine_tunes.prepare_data -f %s' % file_name)
    os.environ['OPENAI_API_KEY'] = token


    # Fine-Tune the model
    result= os.popen('openai api fine_tunes.create -t %s -v %s '
                     '--compute_classification_metrics '
                     '--classification_n_classes %d -m %s' %
                     (f[0]+ '_prepared_train.jsonl',
                      f[0]+ '_prepared_valid.jsonl',
                      len(cat), mod)).readlines()
```

```
# Get the model ID
modelId= result[-1].split()[-3]

# Return the result of the classification
openai.api_key= token
response= openai.Completion.create(model= modelId,
                                   prompt= prt + ' \\n\\n###\\n\\n',
                                   max_tokens= 1,
                                   temperature= 0,
                                   logprobs= 4)
return response['choices'][0]['text']
```