By Microsoft Designer

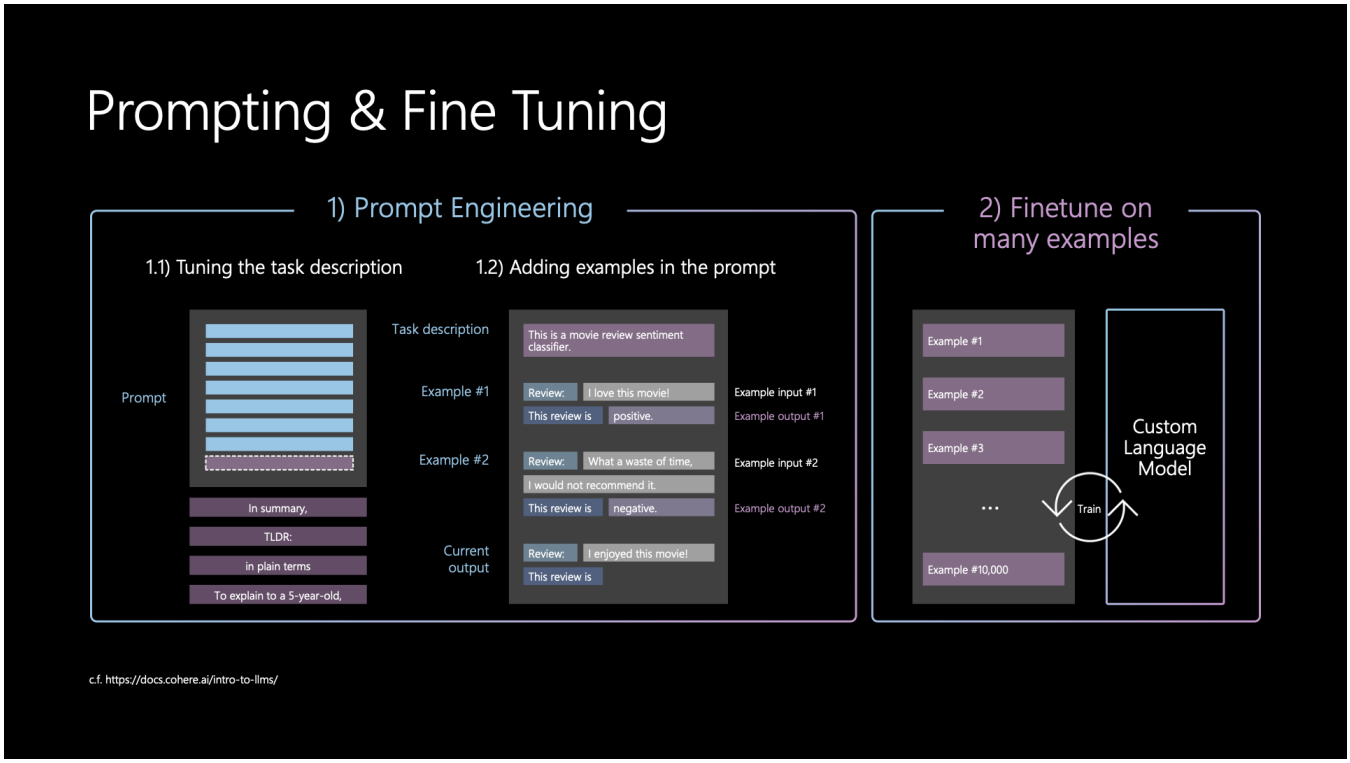
Mastering Generative AI Interactions: A Guide to In-Context Learning and Fine-Tuning

Pradeep Menon

Creating impact through Technology | Data & AI | Cloud Computing | Design

OpenAI's GPT (Generative Pre-trained Transformer) models have revolutionized the field by performing incredible tasks such as language translation, text summarization, and creative writing. If you haven't already, check out my previous two blog posts discussing transformer architecture and how GPT is trained. You can find them here and here!

As the below diagram depicts, one of the most impressive features of GPT models is their ability to learn through both in-context learning (using prompts) and fine-tuning.



In this blog post, we'll dive into both approaches to better understand how they work. Let's get started!

Teaching Through Prompting

Prompting is a technique in which a GPT model adjusts its response based on the context provided in the input text.

Remember that GPT models, like any other LLMs, are pre-trained with a large corpus

the model can efficiently understand and answer what you want.

There are several ways to use in-context learning, and the process can be applied depending on the information given and the desired output. Here are some common ways to use in-context learning:

1. **Prompt engineering**: Carefully crafting prompts to provide clear instructions and context to the model. This can be done using specific keywords, phrases, or formatting to guide the model's response.

2. **Adding context sentences**: Providing additional context in the form of sentences or paragraphs containing relevant background information or explanations. This can help the model understand the problem better and generate more appropriate responses.

3. **Providing examples**: Including examples within the input text can help the model grasp the intended output format or desired answer. This is particularly useful for tasks that require a specific structure or format, such as generating code or writing a specific type of text.

4. **Using conversational context**: Presenting the input text as a conversation between the user and the model with multiple turns can help provide a more natural interaction and ensure the model maintains context throughout the conversation.

5. **Combining approaches**: In many cases, combining multiple approaches can improve the model's performance. For example, providing examples and a clear prompt might help the model generate more accurate and relevant responses.

Key Takeaway: Writing effective prompts for LLMs is crucial for obtaining desired outputs. Key strategies include using task descriptions to guide the model towards specific responses, providing examples for context, fine-tuning with structured prompt-response pairs, and employing best practices such as clarity, iteration, and experimentation. By mastering the art of prompt engineering, you can optimize LLM performance and enhance AI-generated results.

Mastering In-context Learning with the 3C Framework

Let us discuss the art of creating effective prompts for Large Language Models (LLMs). A well-crafted prompt can significantly improve the accuracy of AI assistants' responses to our requests. To assist you in writing the best possible prompts, the 3C framework works well. The 3Cs are:

Clarity, Context, and Constraints

Let us drill into the 3Cs.

1. **Clarity:** Having clarity in prompt engineering is crucial when it comes to guiding large language models (LLMs) such as GPT. The models need to know exactly what is needed of them, and providing specific details with clear intentions will allow them to output the desired results. By keeping prompts clear and concise, the risk of generating irrelevant or off-topic responses can be minimized. This is especially important when dealing with questions with multiple possible interpretations or when generating content that requires precise language. So, it's always best to keep prompts as clear and straightforward as possible to help reduce ambiguity.

2. **Context:** Providing clear context in prompt engineering is crucial for guiding large language models. Context helps the model understand the problem, question, or task more accurately, which can lead to more relevant and focused responses. In addition, context helps maintain coherence across the conversation or series of inputs, ensuring that the model's responses are consistent and connected to the overall theme or topic. You can help disambiguate similar terms, phrases, or concepts by providing context, allowing the model to generate more precise and appropriate responses. Moreover, context can help the model understand the desired format or structure of the response, such as a summary, a detailed explanation, or a comparison between different concepts. By coupling the context with examples, you can guide the model to perform tasks it has not encountered during its pre-training, enabling zero-shot or few-shot learning.

3. **Constraints:** Finally, constraints in prompt engineering can actually help improve the performance of large language models (LLMs) such as GPT by guiding their responses more effectively. By setting clear constraints, you can encourage the model to generate responses that are more focused on the desired topic or question, thus filtering out irrelevant or off-topic information. Constraints can guide the model to generate outputs in a specific format or structure, such as a list, a step-by-step explanation, or a summary. This can be particularly helpful when dealing with tasks that require a particular response type or presentation. By setting constraints on the length or complexity of the response, you can control the verbosity of the model's output, ensuring that it is concise and easy to understand. By providing constraints that require the model to base its response on specific sources, facts, or evidence, you can limit the subjectivity of the generated output and ensure a more objective and reliable response. Additionally, imposing constraints can lead to more creative outputs. Challenging the model to work within specific boundaries or limitations may generate more novel or unexpected responses. All in all, constraints can be a powerful tool in improving the effectiveness and reliability of large language models and can even lead to more interesting and creative outputs.

Key Takeaway: Clarity, context, and constraints are essential in prompt engineering for guiding LLMs like GPT. Clarity minimizes irrelevant responses by reducing

ambiguity and providing clear intentions. Context enhances the understanding, maintains coherence, and enables zero-shot learning by providing relevant background information. Constraints improve model performance by focusing responses, guiding output format, ensuring objectivity, and fostering creative outputs.

Teaching Through Fine-Tuning

In-context learning comes in handy when dealing with a smaller dataset or having limited access to training examples. With in-context learning, you can use prompts and examples within the input to help guide the LLM's output. It's particularly useful for tasks that require adaptation on the fly or when you need the model to generalize from a few examples. So, remember, for smaller datasets or quick adaptability, in-context learning could be your go-to approach!

However, you might want to fine-tune your base LLM model to make it more suitable for your specific use case. Fine-tuning has some amazing advantages, like improved performance on domain-specific tasks, a better understanding of the context, and even reduced risk of generating unwanted content. It's recommended to fine-tune when you're working with a niche domain, dealing with unique datasets, or targeting specific user requirements.

When we fine-tune a large language model (LLM) such as GPT-3, we update its parameters to perform better in a specific task, domain, or dataset. This helps the model generate more accurate and relevant responses when given inputs within the target context. Fine-tuning builds upon the knowledge and understanding that the base model has acquired during its initial pre-training phase.

Take a look at the diagram below shows what happens under the hood during the fine-tuning process:

Let us discuss these steps in a little more detail.

1. **Prepare custom dataset**: To fine-tune the LLM, you must have a dataset that matches the exact task or area you want your model to rock at. This dataset usually has examples of inputs

and outputs or labels the model should produce.

2. **Choose a loss function**: A loss function measures the difference between the model's predictions and the actual target outputs. Fine-tuning aims to minimize this loss by adjusting the model's parameters. Common loss functions used in LLMs include cross-entropy loss for classification tasks and mean squared error for regression tasks.

3. **Set hyperparameters**: Before you start fine-tuning, you should set certain hyperparameters, like learning rate, batch size, and the number of training epochs. These parameters can significantly impact the model's performance and convergence during fine-tuning.

4. **Train the model**: Now, the model needs to be retrained with the new data. While fine-tuning, the model's parameters (weights and biases) get updated using a back-propagation process. This involves calculating the gradients of the loss function concerning the model's parameters and then updating the parameters using an optimization algorithm (e.g., stochastic gradient descent or Adam). The learning rate determines the size of the update steps. One can choose to update all the base model's parameters (although it will be more compute-intensive) or freeze specific layers and only update a few.

5. **Validate model's performance**: As the model gets fine-tuned, it's critical to watch its performance on a validation dataset, which should be different from the training dataset. This helps you track the model's progress and prevent overfitting, where the model gets too focused on the training data and won't work well with new, unseen data.

6. **Stop early or choose the best model**: To avoid overfitting, you can use tricks like early stopping, where the fine-tuning process stops when the model's performance on the validation set starts to decline. Alternatively, you can save the model's parameters at different stages during fine-tuning and pick the version with the best performance on the validation set.

When preparing data for fine-tuning with OpenAI models, you generally need to format your text as a series of input-output pairs. These pairs are structured as prompts (inputs) and corresponding responses (outputs) that the model should learn to generate.

The format should be consistent across all examples in your dataset. One common approach is to use a JSON format, where each example is an object with "prompt" and "completion" fields. For instance:

```
[
```

{ "prompt": "Translate the following English text to French: 'Hello, how are you?'", "completion": "Bonjour, comment ça va?" },

{ "prompt": "What is the capital of France?", "completion": "Paris" } and so on...

]

Key Takeaway: The key takeaway from these paragraphs is that there are two main approaches to tailoring a large language model (LLM) for specific tasks or domains: in-context learning and fine-tuning. In-context learning is suitable for smaller datasets or quick adaptability while fine-tuning is recommended for niche domains, unique datasets, or specific user requirements. Fine-tuning involves several steps, such as picking a dataset, choosing a loss function, setting hyper-parameters, tweaking the model's parameters, monitoring performance, and stopping early or selecting the best model to prevent overfitting.

When to Use What?

Fine-tuning may not always be the optimal choice. In-context learning is useful for dealing with smaller datasets or limited access to training examples. By providing prompts and examples within the input, in-context learning can guide the LLM's. This approach is particularly helpful for tasks that require adaptability or when the model needs to generalize from a few examples. Therefore, remember that in-context learning might be the go-to approach for smaller datasets or quick adaptability.

However, there may be times when fine-tuning the base LLM model is preferable to make it more suitable for your specific use case. Fine-tuning offers several benefits, such as improved performance on domain-specific tasks, a better understanding of context, and reduced risk of generating unwanted content. It is recommended to fine-tune the model when working with a niche domain, unique datasets, or targeting specific user requirements.

Let me explain with an example here:

As we know, GPT-3 is a state-of-the-art large language model that was fine-tuned to create Codex, a specialized language model designed for generating code. To accomplish this, the base GPT-3 model was **fine-tuned** on a diverse and extensive dataset that included programming-related texts and code examples from sources such as GitHub repositories, Stack Overflow, and various technical documentation. As a result, Codex has an improved ability to understand and generate code in multiple programming languages and to solve programming-related tasks more effectively.

Now, the base model of Codex was again fine-tuned to create specialized models like NL2Flow, NL2 Power Fx, and upcoming fine-tuned models like NL2 Action that are part of Microsoft's power platform offerings. The process involved selecting task-specific datasets containing examples of natural language inputs and their corresponding code outputs or actions. By fine-tuning Codex on these specific tasks and datasets, the resulting models can effectively translate natural language instructions into corresponding code, actions, or workflows, catering to the unique requirements of different applications and user groups.

So you might be thinking, why wouldn't prompting work here?

Well, let's be real; prompting could still work for generating code or for specific tasks, but it might not be as effective as fine-tuning. This is because the base GPT-3 model, while super flexible, might not have the super in-depth knowledge or understanding of specific tasks, languages, or domains that the fine-tuned models have.

When you're working with niche domains or tasks, using prompts alone might result in less accurate or relevant outputs. The base model might struggle to really get what's going on or generate code that's smooth and fast for the task at hand. This is where fine-tuning comes in, as it takes the model's knowledge and understanding of the target domain to another level, resulting in better performance and more accurate outputs.

While prompting could still be clutch for simpler tasks or when you're okay with a higher level of generalization, relying only on prompting might not be enough for tasks that need a deeper understanding of a particular domain or programming language. Fine-tuning lets the model go deep, generating more accurate and contextually appropriate responses.

Key Takeaway: In-context learning is a valuable option for smaller datasets or situations requiring quick adaptability. It utilizes prompts and examples within the input to guide the LLM's output, making it ideal for tasks that need on-the-fly adjustments or generalization from a few instances. In contrast, fine-tuning offers enhanced performance on domain-specific tasks, improved contextual understanding, and reduced risk of unwanted content generation. It is best suited for niche domains, unique datasets, or targeting specific user requirements.

Conclusion

In conclusion, LLMs have shown impressive abilities in various tasks such as language translation, text summarization, and creative writing. To make the most of these models, it's important to understand and apply the right approaches, like in-context learning through prompt creation and fine-tuning the model for specific domains or tasks. By thoughtfully engineering prompts and following the 3C framework, you can create clearer and more contextually accurate queries for LLMs. Additionally, fine-tuning allows for even more specialized and improved performance in niche domains, unique datasets, and specific user requirements.

It's essential to understand when to use in-context learning and when to fine-tune it to achieve the best possible results. While in-context learning is great for smaller datasets and quick adaptability, fine-tuning is recommended for more targeted and domain-specific applications. As AI continues to advance and LLMs become even

more powerful, mastering these techniques will be crucial for tapping into their full potential and revolutionizing how we interact with and leverage AI in our daily lives and transforming industries.

References

1. Fine-tuning - OpenAI API
2. Fine-Tuning OpenAI Models with Python: A Step-by-Step Guide — Articulate Python: Python Tutorials
3. How to customize a model with Azure OpenAI Service - Azure OpenAI | Microsoft Learn
4. OpenAI API | Weights & Biases Documentation (wandb.ai)
5. Behind the Scenes – What it Takes to Teach GPT-3 How to Build Low-Code Apps | Microsoft Power Apps
6. Evaluating Large Language Models Trained on Code

Vinod Kumar Rajagopal 3mo

Nice article. And as you keep de-mystifying things .. can't help to compare with : how we bring up kids. Teaching alphabets , what makes a word , how to frame sentence and then how to ask the right questions (and kids can give really good answer with the pure mind) ... I was always fascinated to see how kids learn .. how similar things in a machine 😀

Like · Reply | 1 Reaction

Vinod Seetharaman MD 5mo

Digital Health Evangelist, Clinician-Technologist, Health & Life-sciences Digital Transformation Leader Asia-Pacific | ...

Great read Pradeep! Thank you.

Like · Reply | 1 Reaction

Debananda Ghosh 5mo

Global Black Belt,Senior Specialist- Microsoft Cloud Analytics | Author

Good one Pradeep Menon

Like · Reply | 1 Reaction

Anshul Sharma 5mo

Product @ Microsoft

Nice read, thanks for sharing Pradeep!

Like · Reply | 1 Reaction

Suma Manohar 5mo
Senior Global Black Belt - Data & AI, Microsoft

The article is very insightful and in very easy to understand language. Kudos Pradeep.
Next up, In the series, I would be interested to know your thoughts on democratization of LLMs!

Like · Reply | 1 Reaction

See more comments

To view or add a comment, **sign in**

More articles by this author

The Seven Step Framework: Accelerating...
Sep 1, 2023

Pivoting Towards an AI-Centric Organization
Aug 29, 2023

A Deep-Dive into Fine-Tuning of Large Langua
Aug 11, 2023

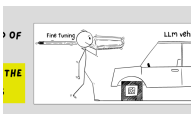
See all

Others also viewed

Generative AI and Workflow Automation: The Dynamic Duo Reshaping Industries
CA Mohammad Hassan · 1mo

Unraveling the Power of Cognitive Search and Generative AI
Faizan Haq · 2mo

Contextual Artificial Intelligence
Sonia Ramnani · 5mo



A Beginner's Guide to Fine-Tuning Large Language Models
Archana Vaidheeswaran · 3mo

Unlocking the AI Magic: Demystifying Prompt Engineering, RAG, and Fine-Tuning
Vineet Kumar · 1mo

Analyzing Tabular Data with Large Language Models
Abdul (Owais) Hazari · 2mo

Show more

Insights from the community

Artificial Intelligence

How do you improve your deep learning models with user input?

Artificial Intelligence

How do you increase the speed of reinforcement learning?

Deep Learning

How do you use GAN and VAE for data augmentation and semi-supervised learning?

Artificial Intelligence

How do you interpret and visualize the attention weights and outputs of your model?

Digital Asset Management

What are the best tools and methods for automating metadata generation and tagging?

Machine Learning

What is the best way to use transfer learning for few-shot deep learning?

Show more

Explore topics

Sales

Marketing

Public Administration

Business Administration

HR Management

Engineering

Soft Skills

See All