

NLP 2

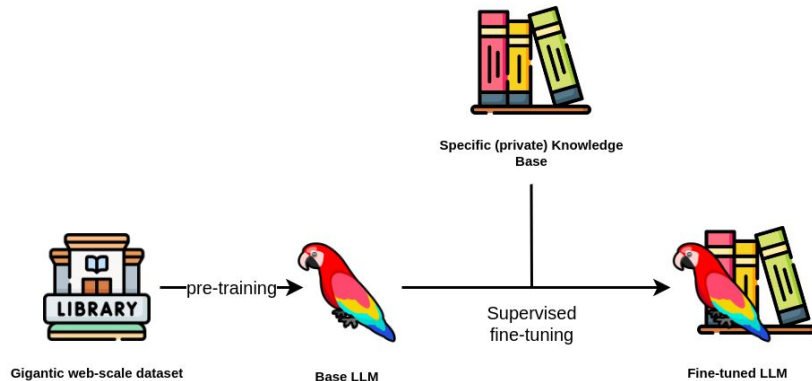
Inteligencia artificial avanzada para la ciencia
de datos II Modulo 5 NLP 2

Fine tuning / Transference learning

LLM fine tuning

Fine-tuning is a technique that helps train accurate prediction models even with small datasets.

It is often used when a deep learning-based solution is required, but the available data is not sufficient to train a CNN from scratch.



Example

Davinci GTP-3 is a base model that transforms an input into one or more questions (meanings) to accomplish a task.

Text.davinci-003 is a tuned model that reuses davinci core model and then generates the answers to those questions.

Tasks are not the same, models changes but training just improve.

Base Model

(davinci (GPT-3))

Prompt: Tell me how to fine-tune a model

Completion:

How can I control the complexity of a model?
How do I know when my model is done?
How do I test a model?
How do I define a task for my model?
How do I scale a model?
How do I make a probabilistic model?
How do I start using a model once it's complete?
How do I cope with missing data?
How do I make my modeling efforts sustainable?

Fine-tuned Model

(text-davinci-003)

Prompt: Tell me how to fine-tune a model

Completion:

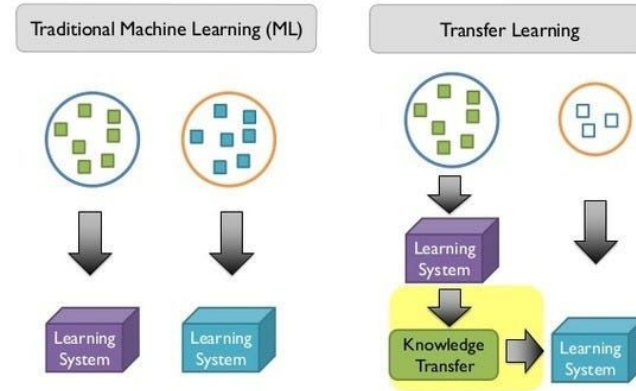
Fine-tuning a model involves adjusting the parameters of a pre-trained model in order to make it better suited for a given task. There are generally three steps to fine-tuning a model:

1. Select a base model - Select a pre-trained deep learning model that has been trained on a large dataset.
2. Adjust parameters - Adjust parameters of the pre-trained model to better suit the desired task. This may include changing the number of layers, adjusting learning rate, adding regularization, or tweaking the optimizer.
3. Train the model - Train the new model on the desired dataset. The amount of data and the amount of training required will depend on the task and the model.

Transfer learning

In transfer learning, a model trained on one task is reused to solve another task, but freezing the parameters of the existing model. The process is as follows:

- The trained model is loaded and the pre-trained layers are frozen to prevent information loss.
- New trainable layers are added on top of the frozen layers, which are trained on another dataset.



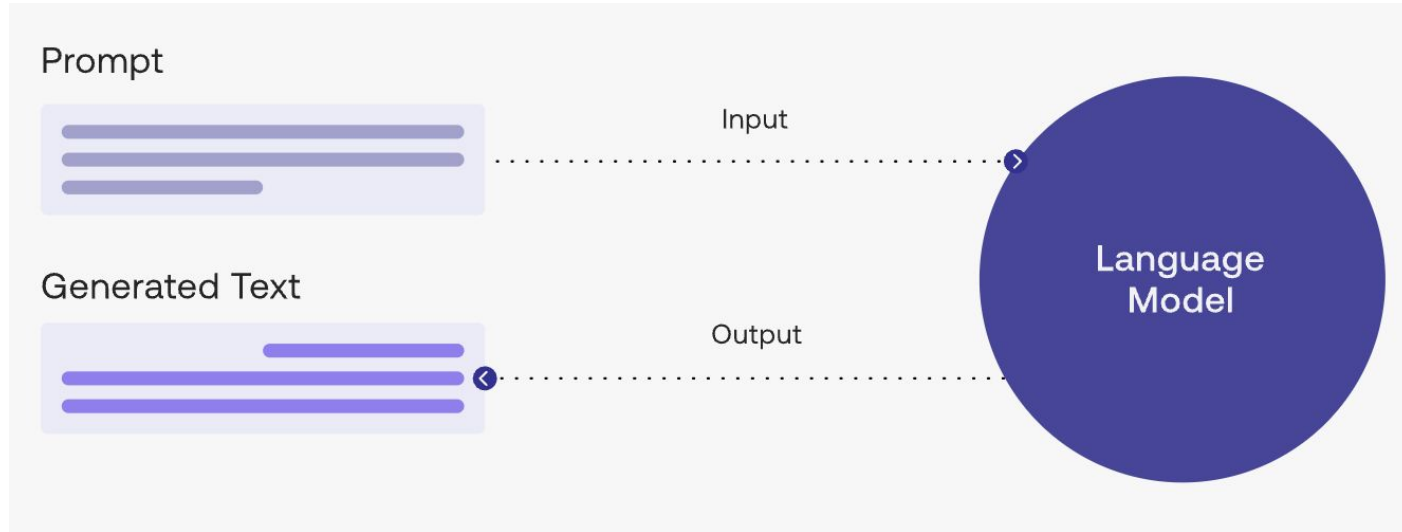
Benefits

Aspect	Transfer Learning	Fine-Tuning
Pre-trained Knowledge	Leveraging pre-trained models like BERT, GPT, etc., provides access to a vast amount of general language knowledge.	Utilizes pre-trained models as a starting point, which can accelerate the learning process.
Reduced Data Requirement	Requires less labeled data for downstream tasks since the model has already learned from extensive data.	Requires less data than training from scratch, but more data than transfer learning.
Generalization	Can perform well on a wide range of NLP tasks due to its general language understanding.	Adapts the pre-trained model's knowledge to a specific NLP task, potentially achieving better task-specific performance.
Time-Efficiency	Faster development of NLP applications as most of the model is already trained, saving time and resources.	Faster than training from scratch but may still require significant fine-tuning time.

Limitations

Aspect	Transfer Learning	Fine-Tuning
Domain Specificity	May not perform optimally for domain-specific tasks if the pre-trained model lacks domain-specific knowledge.	Relies on the quality and relevance of the pre-trained model to the target task; may not always adapt well to niche domains.
Overfitting Risk	The model may overfit the training data if not fine-tuned properly, leading to reduced performance on the target task.	Fine-tuning may lead to overfitting if the labeled data is insufficient or the hyperparameters are not tuned correctly.
Interpretability	Pre-trained models are often large and complex, making it challenging to interpret their decision-making processes.	Fine-tuning can make it harder to interpret model outputs since it incorporates task-specific nuances.
Transferability	Faster development of NLP applications as most of the model is already trained, saving time and resources.	Faster than training from scratch but may still require significant fine-tuning time.

Prompt Engineer



The future of AI, prompt engineer deals with the way we instruct LLMs, there are many ways to ask for something

Chain of thoughts

A "chain of thoughts" refers to the model's ability to generate a coherent and contextually relevant sequence of text based on the input it receives. LLMs can generate responses or text that appear as if they follow a logical and structured chain of thoughts, even though they are not conscious beings and do not actually think.

<https://arxiv.org/abs/2201.11903>

Standard Prompting

Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The answer is 27. ❌

Chain-of-Thought Prompting

Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. $5 + 6 = 11$. The answer is 11.

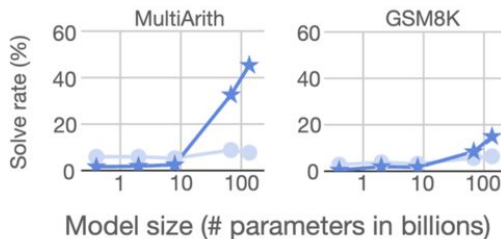
Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The cafeteria had 23 apples originally. They used 20 to make lunch. So they had $23 - 20 = 3$. They bought 6 more apples, so they have $3 + 6 = 9$. The answer is 9. ✅

LaMDA:

- standard prompting
- ★ chain of thought prompting



Chain of thoughts

A "chain of thoughts" refers to the model's ability to generate a coherent and contextually relevant sequence of text based on the input it receives. LLMs can generate responses or text that appear as if they follow a logical and structured chain of thoughts, even though they are not conscious beings and do not actually think.

<https://arxiv.org/abs/2201.11903>

Standard Prompting

Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The answer is 27. ❌

Chain-of-Thought Prompting

Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. $5 + 6 = 11$. The answer is 11.

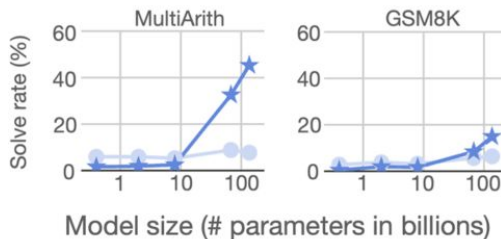
Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The cafeteria had 23 apples originally. They used 20 to make lunch. So they had $23 - 20 = 3$. They bought 6 more apples, so they have $3 + 6 = 9$. The answer is 9. ✅

LaMDA:

- standard prompting
- ★ chain of thought prompting

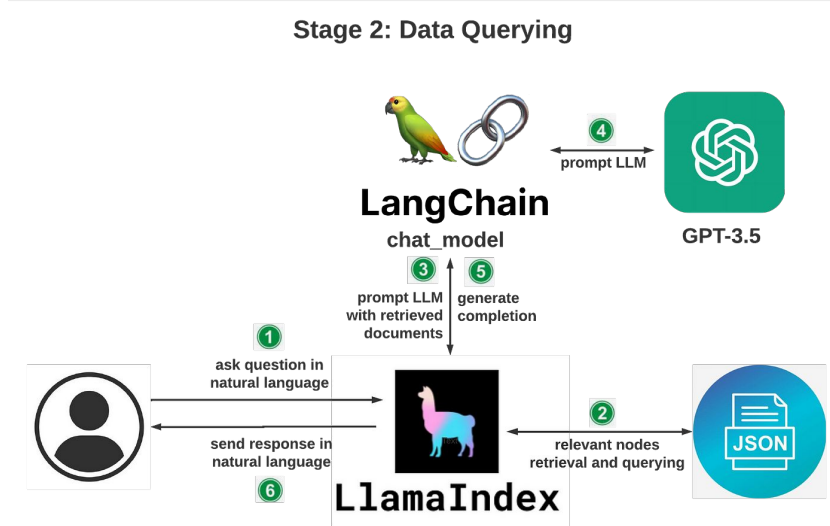


Langchain

LangChain is a framework for developing applications powered by language models. It enables applications that:

Are context-aware: connect a language model to sources of context (prompt instructions, few shot examples, content to ground its response in, etc.)

Reason: rely on a language model to reason (about how to answer based on provided context, what actions to take, etc.)



Thanks

Do you have any questions?

emmanuel.paez@tec.mx
Slack #module-5-nlp-1