**Theory - Large language models (LLMs)**

For the theory about NLP & Transformers, embeddings, take a look at the previous document

Large Scale Language Models (LLMs) are natural language processing (NLP) models that have been trained on extensive datasets to enhance their language processing abilities. These models have the capacity to analyze and comprehend textual data to accomplish multiple tasks like language translation, sentiment analysis and content generation.

Some of the well-known examples of large language models include OpenAI's ChatGPT, Microsoft's Turing NLG, Deepmind's Gopher and

Chinchilla, Google's T5 and MT5, and Baidu's Ernie 3.0, all of which are trained on billions of tokens.

For NLP Basics, check the document : 0\_NLP.docx

## Parameters of LLMs (temperature, top-k, top-p…):

LLM Parameters are settings that you can adjust to control how the LLM generates texts. They can affect the quality, diversity, and creativity of the generated texts. Some of the common LLM parameters are temperature, number of tokens, top-p, presence penalty, and frequency penalty.

**Temperature**: selects the next word to output using probabilities for all the different words that could follow and then. A Temperature of 0 makes the model deterministic. It limits the model to use the word with the highest probability. Adjusting the temperature value can affect the style of the output. A lower temperature will result in more realistic and dependable text, while a higher temperature will produce more creative and humorous output. However, it's important to avoid extremes in temperature, as this can lead to nonsensical text.

**Top-k & Top-p :** Aside from Temperature, Top-k and Top-p are the two other ways to pick the output token.

Top-k tells the model to pick the next token from the top ‘k’ tokens in its list, sorted by probability. If you set k as 3, you’re telling the model to only pick from the top 3 options.

Une image contenant texte, capture d’écran, Police

Description générée automatiquementTop-p is similar but picks from the top tokens based on the sum of their probabilitiesTop-p is a parameter that controls how many words are considered candidates for the next word in the text generation process. A higher “Top-p” means more diversity and creativity, while a lower “Top-p” means more accuracy and reliability.

**The number of tokens:** The number of tokens is a parameter that controls how long the generated text is. A higher number of tokens means more detail and information, while a lower number of tokens means more conciseness and simplicity.

**Stop Sequences**

A stop sequence is a string that tells the model to stop generating more content. It is another way to control how long your output is. It stops the generation when crossing a word even if the number of tokens limit is much higher.

**Presence Penalty:**

Presence penalty is a parameter that controls how much the generated text reflects the presence of certain words or phrases in the output text so far. A higher “presence penalty” encourages the model to explore new topics and makes it less repetitive, while a lower “presence penalty” means more repetition and less exploration.

**Frequency Penalty:**

**Sources:**

* [The Secrets of Large Language Models Parameters: How They Affect the Quality, Diversity, and… | by Michael Ehab | Medium](https://michaelehab.medium.com/the-secrets-of-large-language-models-parameters-how-they-affect-the-quality-diversity-and-32eb8643e631#:~:text=LLM%20Parameters%20are%20settings%20that,presence%20penalty%2C%20and%20frequency%20penalty.)
* [LLM Parameters Demystified: Getting The Best Outputs from Language AI (cohere.com)](https://txt.cohere.com/llm-parameters-best-outputs-language-ai/)
* [🟢 Paramètres LLM | Learn Prompting: Your Guide to Communicating with AI](https://learnprompting.org/fr/docs/basics/configuration_hyperparameters)

## RLHF :

Reinforcement learning from Human Feedback (also referenced as RL from human preferences) is a challenging concept because it involves a multiple-model training process and different stages of deployment. The training process is composed of three core steps:

1. Pretraining a language model (LM),
2. gathering data and training a reward model, and
3. fine-tuning the LM with reinforcement learning.

How :

* **Pretraining language models:** With a language model, one needs to generate data to train a reward model, which is how human preferences are integrated into the system
* Une image contenant texte, diagramme

  Description générée automatiquement**Reward model training** : Generating a reward model (RM, also referred to as a preference model) calibrated with human preferences is where the relatively new research in RLHF begins. The underlying goal is to get a model or system that takes in a sequence of text, and returns a scalar reward which should numerically represent the human preference.

The training dataset for the RM is generated by sampling prompts and passing them through an LM to generate new text. Human annotators rank the generated text outputs from the LM, and rankings are used to create a regularized dataset.

Une image contenant texte, diagramme, capture d’écran, Police

Description générée automatiquement

* **Fine-tuning with RL:** Training a language model with reinforcement learning (RL) involves fine-tuning some or all of the parameters of the initial model using the Proximal Policy Optimization (PPO) algorithm.

Sources :

* [Illustrating Reinforcement Learning from Human Feedback (RLHF) (huggingface.co)](https://huggingface.co/blog/rlhf)
* [Reinforcement Learning from Human Feedback: From Zero to chatGPT - YouTube](https://www.youtube.com/watch?v=2MBJOuVq380)

## Finetuning

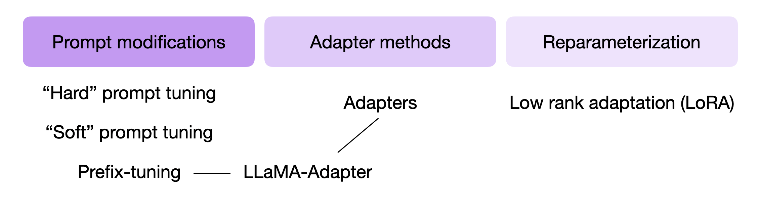
**Why Finetuning?**

Pretrained large language models are often referred to as foundation models for a good reason: they perform well on various tasks, and we can use them as a foundation for finetuning on a target task.

**parameter-efficient finetuning methods for LLMs**

How can we adapt a model to a target task? There are three conventional approaches. These methods above are compatible with generative (decoder-style) models such as GPT and embedding-focused (encoder-style) models such as BERT. In contrast to these three approaches, in-context learning only applies to generative models. It’s also worth highlighting that when we finetune generative models, we work with and build on the embeddings they create instead of the generated output texts.

1. Feature-based approach
2. Finetuning I — Updating The Output Layers
3. Finetuning II — Updating All Layers

**parameter-efficient finetuning techniques (PEFT)**

Most widely used techniques:

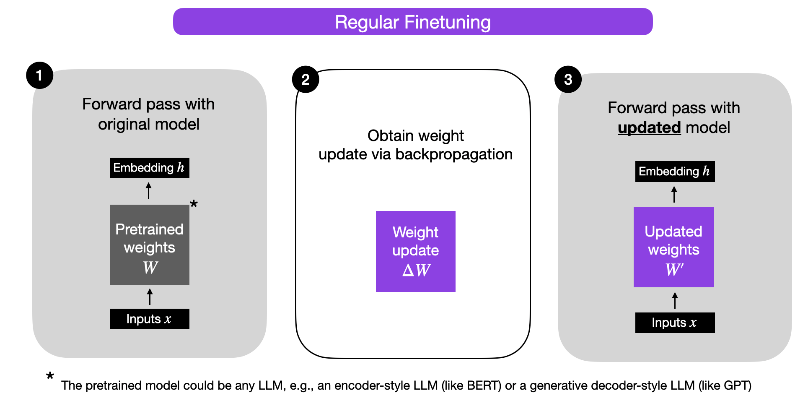
* Hard & soft prompt tuning
* Adapters
* LLaMA-Adapter
* LoRA

Sources:

* [Understanding Parameter-Efficient Finetuning of Large Language Models: From Prefix Tuning to LLaMA-Adapters (lightning.ai)](https://lightning.ai/pages/community/article/understanding-llama-adapters/)

**Lora:**

Une image contenant texte, capture d’écran, Police, conception

Description générée automatiquement

Alternatively, we can keep the weight update matrix separate and compute the outputs as follows: h = W x + ΔW x

Une image contenant texte, capture d’écran, Police, conception

Description générée automatiquementWhen we train fully connected (i.e., “dense”) layers in a neural network, as shown above, the weight matrices usually have full rank, which is a technical term meaning that a matrix does not have any linearly dependent (i.e., “redundant”) rows or columns. In contrast, to full rank, low rank means that the matrix has redundant rows or columns.

low “intrinsic dimension” -> low internal complexity in dimensions. A low intrinsic dimension means the data can be effectively represented or approximated by a lower-dimensional space while retaining most of its essential information or structure.

In other words, this means we can decompose the new weight matrix for the adapted task into lower-dimensional (smaller) matrices without losing too much important information.

ΔW = WA.WB, where WA is an an A × r-dimensional matrix, and WB is an an r × B-dimensional matrix. With a smaller r, the capacity of the low-rank matrix to capture task-specific information decreases.

Details in the source below.

Advantages :

* Parameter efficiency: The new Wa Wb matrices have low dimensions compared to ΔW
* Reducing inference overhead: Keep our pretrained model as a base model for various customers, and we want to create a finetuned LLM for each customer starting from the base model. In this case, we don’t need to store the full weight matrices W’ for each customer, where storing all the weights W’.
* To illustrate this point with concrete numbers, a full 7 B LLaMA checkpoint requires 23 GB of storage capacity, while the LoRA weights can be as small as 8 MB if we choose a rank of r=8.

Sources:

* [Parameter-Efficient LLM Finetuning With Low-Rank Adaptation (LoRA) - Lightning AI](https://lightning.ai/pages/community/article/lora-llm/)

## Language generation strategies

[How to generate text: using different decoding methods for language generation with Transformers (huggingface.co)](https://huggingface.co/blog/how-to-generate)