How is an LLM trained?

Abdelwahid Benslimane
wahid.benslimane@gmail.com

Introduction

The explanations in the present document are intentionally conceptual and aimed at a wide audience, with a deliberate omission of technical and mathematical details that are not crucial for a general understanding of the training framework.

For an overview of how an LLM works, you can read the following document (suitable for non-technical people): https://github.com/abenslimaneakawahid/generative-AI/blob/main/GenAI_simplified.pdf.

A large language model (LLM), like any deep learning model, is characterized by a set of parameters, also known as weights, which are numerical values that contribute to the mathematical representation of the task the model is supposed to perform. These weights are randomly initialized at the start, and to make them converge towards the values that will ensure that the model is effective, we need to train the model.

Training is a crucial step as the ultimate quality of the model relies heavily on the quality of both the training process and the material required to handle this phase: the training dataset. Therefore, it is essential to approach it with the utmost care.

The training process can be broken down into several parts: initial training, fine-tuning and reinforcement learning with human feedback (RLHF). They are all primarily based on gradient descent and gradient backpropagation, provided with some enhancements or adaptations.

For advanced explanations about gradient descent and gradient backpropagation, although not necessary to understand what follows, you can read these documents (a good mathematical background is required):

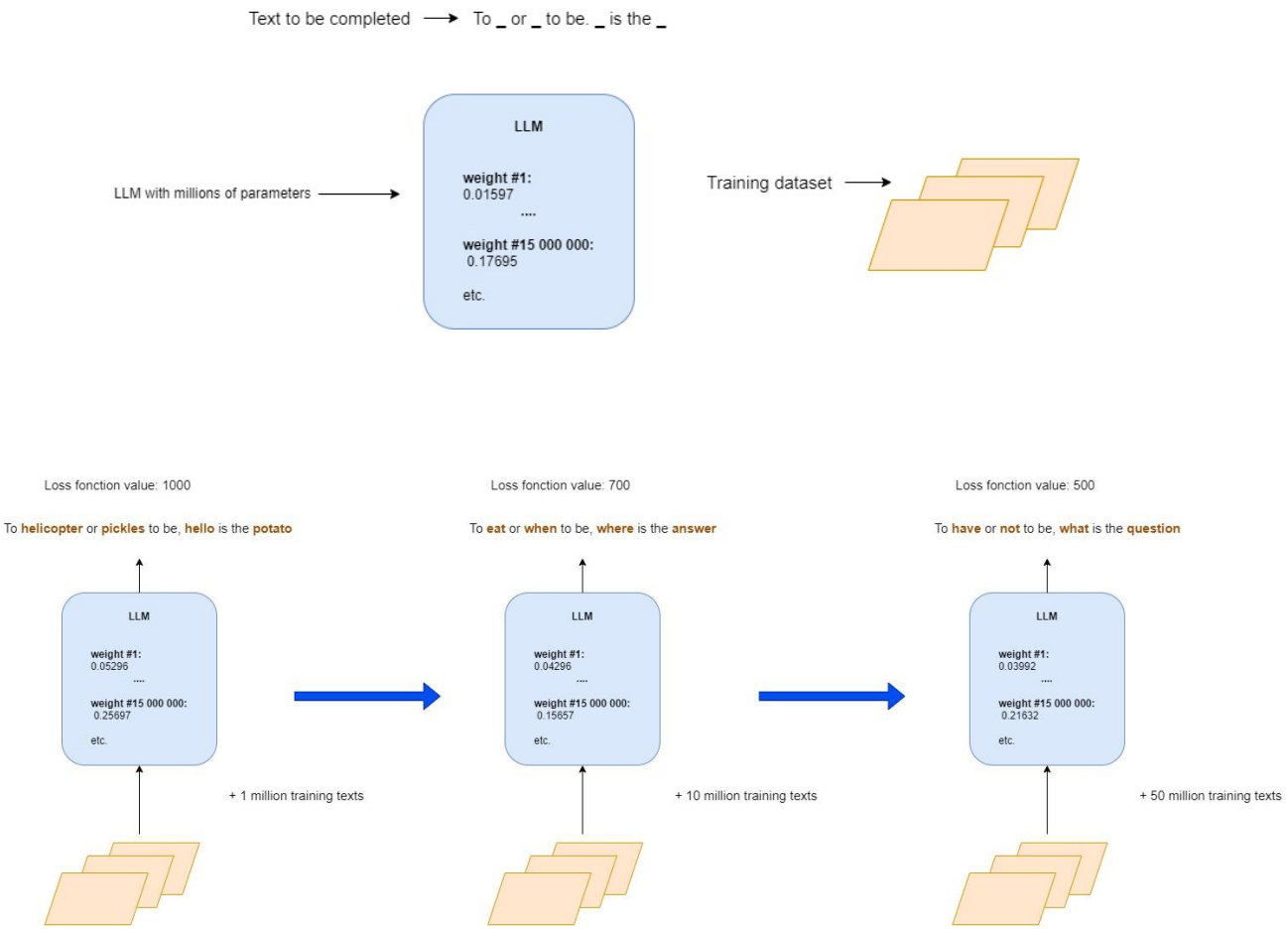
- gradient descent: https://github.com/abenslimaneakawahid/iterative-methods/blob/main/Gradient_desc_opt_var_step_size.pdf,
- gradient backpropagation : https://github.com/abenslimaneakawahid/ML-and-maths-theory/blob/main/gradient_backpropagation.pdf.

Initial training

This first training stage mainly consists of learning the structure of a language, or at least the pattern of relationships between words with respect to given sequence of words and in a given language on the basis of staggering amount of public data mostly retrieved from the Internet.

In fact, we train the model to guess words in blank texts and we associate the model's performance (its ability to guess words) with a loss function that we try to minimise (using gradient descent).

The more the model is subjected to training data, the more the value of the loss function will decrease and the better the model will be able to guess the words it is supposed to guess. This part is called unsupervised learning because the input data are in no way associated with any output. The model is not guided and is left to find its bearings on its own.



After this first stage, the LLM is capable of generating word sequences that are already very coherent and realistic but can hardly be used for a very specific case if it has not been confronted with data representative of it.

If we want the model to adopt a specific behaviour or to be able to answer questions relating to a precise domain or to accomplish a specific task, at least a second stage is necessary: fine-tuning.

Fine-tuning

Fine-tuning still consists of word prediction, but this time the training data submitted to the model are input-output pairs.

Let's assume that we want to train the model to sound like a member of the British aristocracy. we will submit a set of instructions to the model, each associated with a response in the form of the one we want to obtain, as in the example below.

Input: “give me an example of an introductory sentence in an e-mail to a colleague”

Output: “Greetings, esteemed colleague, and I trust this missive finds you in the finest of spirits.”

Fine-tuning may involve the addition of task-specific layers to the LLM, if we want to train it on a particular task.

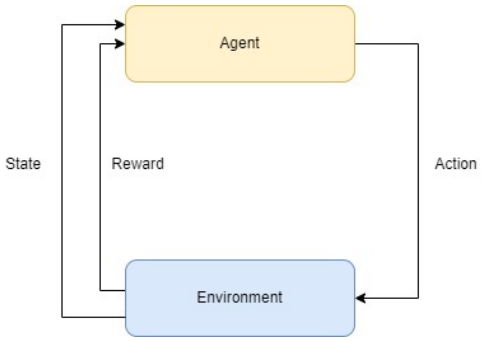
This part of the learning process is called supervised learning because we influence the behaviour of the model, and the input training data is associated with calibrated responses written or selected by members of the team in charge of building the model. However, the amount of data required for this part is much smaller than for the initial training.

In order to obtain even better-quality results, or at least results that meet the expectations of human users, we can resort to RLHF.

RLHF

Before explaining what RHLF is, we first need to define reinforcement learning. Reinforcement learning consists in training an agent (an LLM, a robot, an autonomous car, etc.) to behave in a certain way or to accomplish a task by means of “rewards”, which it then seeks to optimise by adapting its strategy as it goes along. For example, one might want to train a robot to throw a paper ball into a trash bin. If the paper ball lands in the bin, the robot will receive the maximum reward, and the closer it gets to its goal, the more points it will receive.

In technical terms, it can be said that the agent (robot) will perform actions (shooting) within a specific environment (a site containing a paper ball and a trash bin), which will update the state of the environment (place where the paper ball lands) as well as the agent's internal representation of the environment and result in rewards, allowing the agent to adapt its future actions, all in a cyclical fashion.



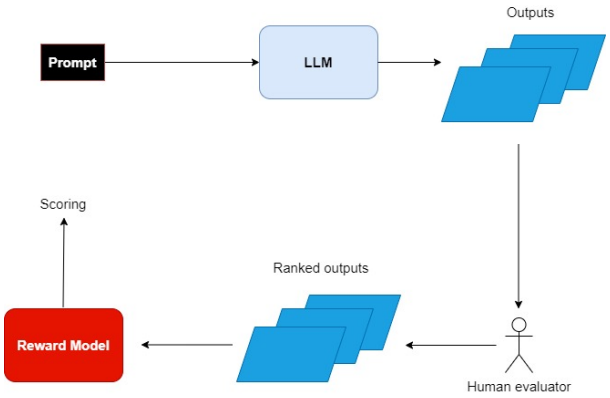
Going back to the LLM or language in general, for every prompt/question, there can be multiple possible outputs/answers, varying not only in form but also in substance, and the esteemed quality

of each depends on the judgment of users or interlocutors. When training an LLM, though, one may seek to guide it towards a specific type of response.

To implement RLHF (supervised learning), it is necessary to have a reward model (RM), which is essentially a model that takes into account the text generated by a pre-trained and fine-tuned LLM that we aim to improve and produces a quality score. This RM is typically another LLM that has been modified and trained to generate a scalar value instead of a sequence of words. The RM thus produces a mathematical modelling of human preferences.

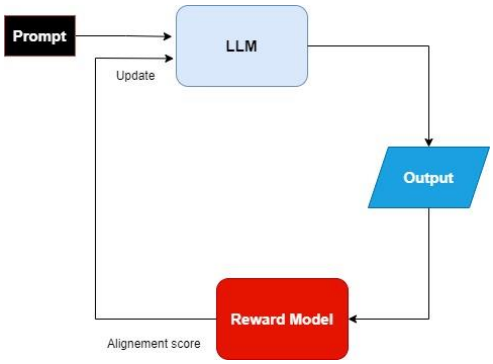
The primary LLM (the one we want to improve) becomes the agent in the context of RLHF.

To create a training dataset for the RM, prompts are submitted to the primary LLM, which generates multiple outputs per prompt. Human evaluators are then asked to rank them based on their estimated quality and label them accordingly (this is why we speak of human feedback).



Ultimately, once the RM is trained, we can initiate the reinforcement learning process.

During each iteration of this process, prompts are presented to the LLM for text generation, and the resulting texts are then passed to the RM which will calculate a score based on their alignment with human preferences. The LLM is then updated to generate texts that will achieve even better scores in the next iteration.



Considering the high cost associated with updating the primary LLM, parts of the set of weights may be frozen to mitigate training expenses.