Large Language Models

MUSTAFA HAJIJ

Language modeling

A language model is a model that acquires statistical insights into language, providing information on the probability of various elements, such as words, characters, or subword units, occurring in specific contexts.

Tokens, in the context of a language model, encompass entities like individual words ("apple"), characters ("a"), or subword units ("tion") depending on the specific language model. Tokens essentially form the vocabulary that the language model employs to comprehend and generate text.

Language modeling

In our context we will adapt the following convention: language modeling is the task of predicting what word comes next.

More formally, given a sequence of words $x_1, x_2, \dots, x_t$, compute the probability distribution of the next word x_{t+1} :

$$P(x_{t+1} | x_1, x_2, \dots, x_t)$$

x_{t+1} can be any word in the vocabulary $V = \{T_1, T_2, \dots, T_{|V|}\}$

Any system that implements the above probability function will be called a language model.

Definition

Let $\boldsymbol{x} = (\boldsymbol{x_1}, \ldots, \boldsymbol{x_t})$ be a sequence of tokens from V. A large language model is a function of the form :

$$LLM_{\phi}(x) = (\bar{y}_1, \ldots, \bar{y}_{|V|}) \quad \text{with } \sum_{k=1}^{|V|} \bar{y}_k = 1$$

$and\ \phi$ is the parameters of the model

Definition

Let $\boldsymbol{x} = (\boldsymbol{x_1}, \ldots, \boldsymbol{x_t})$ be a sequence of tokens from V. A large language model is a function of the form :

$$LLM_{\phi}(x) = (\bar{y}_1, \ldots, \bar{y}_{|V|}) \quad \text{with } \sum_{k=1}^{|V|} \bar{y}_k = 1$$

$and\ \phi$ is the parameters of the model

Note: $The\ consideration\ of\ a\ language\ model\ as$ "large“ does not only rely on the size of the parameter ϕ and it $can\ vary\ depending\ on\ various\ factors, such\ as\ the\ specific\ context, the\ available\ computing\ resources, and\ the\ state\text{-}of\text{-}the\text{-}art\ models\ at\ the\ time.$

Training an LLM with Reinforcement Learning using human feedback RLHF

Training an LLM is usually done with three steps

(1) The pretrained for completion model, initially an untamed monster trained on indiscriminate Internet data.
(2) Supervised finetuning (SFT) for dialogue : underwent finetuning with higher quality information.
(3) Finally, the model was polished using RLHF to become customer-appropriate.

Phase 1. Pretraining for completion : Training data

Let $x = (x_1, \ldots, x_n)$ be a sequence of n tokens. We obtain n training points obtained from x as follows :

Phase 1. Pretraining for completion : Training data

Let $x = (x_1, \ldots, x_n)$ be a sequence of n tokens. We obtain n training points obtained from x as follows :

$$(x_1) \rightarrow y = x_2$$

$$(x_1, x_2) \rightarrow y = x_3$$

$$(x_1, x_2, x_3) \rightarrow y = x_4$$

...

...

$$(x_1, x_2, \ldots, x_{n-1}) \rightarrow y = x_n$$

Phase 1. Pretraining for completion : Training data

Let $x = (x_1, \ldots, x_n)$ be a sequence of n tokens. We obtain n training points obtained from x as follows :

$$(x_1) \rightarrow y = x_2$$

$$(x_1, x_2) \rightarrow y = x_3$$

$$(x_1, x_2, x_3) \rightarrow y = x_4$$

...

...

$$(x_1, x_2, \ldots, x_{n-1}) \rightarrow y = x_n$$

Thus a training sample in our case is a tuple of the form : $(\boldsymbol{x}, x_t) = ((x_1, x_2, \ldots, x_{t-1}), x_t)$

Phase 1. Pretraining for completion : loss function

1.Now, let's define the cross entropy loss between the predicted distribution LLM(x) and the ground truth distribution P(x_t). The cross entropy loss is defined as follows:

2.CE($\boldsymbol{x}, x_t$; ϕ) = $\sum_{k=1}^{|V|}$ P(x_t) * log(LLM(x))[k]

3.Here, P(x_t) is the ground truth probability distribution, log(LLM(x))[k] is the logarithm of the k-th element (predicted probability) of LLM(x), and $\sum_{k=1}^{|V|}$ denotes the sum over all elements of the vocabulary.

4.Since P(x_t) is a one-hot vector, P(x_t)[j] equals 1 if j equals t (the ground truth token's position) and 0 otherwise. Therefore, the cross entropy loss simplifies to:

5.CE($\boldsymbol{x}$, x_t; ϕ) = -log(LLM(x))[t]=-log($\bar{y}_t$)

6.This formula calculates the negative logarithm of the predicted probability for the ground truth token.

7.To find the overall expected loss over all training samples, we take the average over all training samples:

8.CE(ϕ) = -E[x]log(LLM(x))[t]=-E[x]log($\bar{y}_t$), where E[x] denotes the expectation over all training samples.

When do we consider a model large?

- Model Size: The size of a language model is often a key factor in determining its scale. Language models are typically measured in terms of the number of parameters they have.

- Training Data: The amount and diversity of training data used to train the language model can also contribute to its scale. Large-scale language models are often trained on massive datasets comprising billions or even trillions of words, encompassing various sources such as books, websites, and other textual resources.

When do we consider a model large?

- Model Size: The size of a language model is often a key factor in determining its scale. Language models are typically measured in terms of the number of parameters they have.
- Training Data: The amount and diversity of training data used to train the language model can also contribute to its scale. Large-scale language models are often trained on massive datasets comprising billions or even trillions of words, encompassing various sources such as books, websites, and other textual resources.
- Computational Resources: The computational resources required to train or run a language model can indicate its scale. Training large models typically demands substantial computing power, including high-performance GPUs or specialized hardware. Similarly, running large models for inference or fine-tuning may require significant memory and processing capabilities.
- Performance and Capabilities: Large language models often exhibit improved performance in various natural language processing tasks, such as text generation, translation, or question answering. They can demonstrate enhanced language understanding, context retention, and coherent generation of text.

Data bottleneck

With the increasing scale of language models such as GPT-4, there is a growing concern that the amount of data they require for training may eventually deplete the available Internet data within a few years.

The rate of training dataset size growth is much faster than the rate of new data being generated (Villalobos et al, 2022).

Phrase 2 : Supervised finetuning (SFT) for dialogue

- The pretraining of a language model focuses on generating completions. When presented with a question, such as "How to make pasta," the pretrained model can provide various valid completions.

- These can include adding more context, like specifying it is for a family of six, posing follow-up questions about ingredients and time, or directly providing the answer.

Phrase 2 : Supervised finetuning (SFT) for dialogue

- The pretraining of a language model focuses on generating completions. When presented with a question, such as "How to make pasta," the pretrained model can provide various valid completions.

- These can include adding more context, like specifying it is for a family of six, posing follow-up questions about ingredients and time, or directly providing the answer.

- The preferred option is the last one when seeking a direct answer. The goal of Supervised Fine-Tuning (SFT) is to optimize the pretrained model to generate responses that align with user expectations.

Phrase 2 : Supervised finetuning (SFT) for dialogue

- The pretraining of a language model focuses on generating completions. When presented with a question, such as "How to make pasta," the pretrained model can provide various valid completions.

- These can include adding more context, like specifying it is for a family of six, posing follow-up questions about ingredients and time, or directly providing the answer.

- The preferred option is the last one when seeking a direct answer. The goal of Supervised Fine-Tuning (SFT) is to optimize the pretrained model to generate responses that align with user expectations.

- During SFT, demonstration data in the form of prompt-response pairs is used to train the model on appropriate behavior for different use cases, such as question answering, summarization, and translation.
- OpenAI refers to this training approach as behavior cloning, where the model learns to mimics the desired behavior **demonstrated** in the training examples.

Phrase 2 : Demonstration data

- Demonstration data, used in models like InstructGPT and ChatGPT, is created by skilled human labelers who undergo a screening process.
- These labelers possess a high level of education, with approximately 90% having at least a college degree, and over one-third holding a master's degree.
- This ensures the quality and expertise in generating the demonstration data, setting it apart from traditional data labeling approaches.

Example Ref: 2203.02155.pdf (arxiv.org)

Prompt:

Serendipity means the occurrence and development of events by chance in a happy or beneficial way. Use the word in a sentence.

Labeler demonstration

Running into Margaret and being introduced to Tom was a fortunate stroke of serendipity.

Phase 2 : Training data

Training Samples: To create training samples, for each pair (prompt, response) we concatenate the prompt and response sequences and incrementally add tokens from the response. Each training sample (x, y) is formed by the concatenation of the prompt and a subset of tokens from the response.

Phase 2 : Training data

Example:
Prompt: "How to make pizza“
Response: "First, gather the ingredients. Then, mix the dough and let it rise. Next, spread the sauce and add toppings. Finally, bake in the oven until golden brown.“

Training samples:

1.(x1, y1) = ("How to make pizza", "First")
2.(x2, y2) = ("How to make pizza First", "gather")
3.(x3, y3) = ("How to make pizza First gather", "the")
4.(x4, y4) = ("How to make pizza First gather the", "ingredients.")
5.(x5, y5) = ("How to make pizza First gather the ingredients.", "Then")
6.(x6, y6) = ("How to make pizza First gather the ingredients. Then", "mix")
7.(x7, y7) = ("How to make pizza First gather the ingredients. Then mix", "the")
8.(x8, y8) = ("How to make pizza First gather the ingredients. Then mix the", "dough")
9.(x9, y9) = ("How to make pizza First gather the ingredients. Then mix the dough", "and")
10.(x10, y10) = ("How to make pizza First gather the ingredients. Then mix the dough and", "let")

Loss function to minimize during the training process: cross entropy, Note that only the tokens in the response are counted towards the loss.

Phase 3 : RLHF

“we expect human feedback (HF) to have the largest comparative advantage over other techniques when people have complex intuitions that are easy to elicit but difficult to formalize and automate.” (Bai et al., 2022)

Phase 3 : RLHF

The idea is to introduce a scoring function that assesses the goodness of a response given a prompt. This scoring function helps in training the language models to generate high-scoring responses. This approach is known as Reinforcement Learning from Human Feedback (RLHF), which involves two main components:

Phase 3 : RLHF

The idea is to introduce a scoring function that assesses the goodness of a response given a prompt. This scoring function helps in training the language models to generate high-scoring responses. This approach is known as Reinforcement Learning from Human Feedback (RLHF), which involves two main components:

- Train a reward model as a scoring function.
    - A reward model is trained to assign scores to different responses based on their quality.
    - The reward model acts as the scoring function, indicating how good a response is given a specific prompt.
    - The reward model is typically trained using human feedback or other evaluation mechanisms to capture the desired response quality.

Phase 3 : RLHF

The idea is to introduce a scoring function that assesses the goodness of a response given a prompt. This scoring function helps in training the language models to generate high-scoring responses. This approach is known as Reinforcement Learning from Human Feedback (RLHF), which involves two main components:

- Train a reward model as a scoring function.
    - A reward model is trained to assign scores to different responses based on their quality.
    - The reward model acts as the scoring function, indicating how good a response is given a specific prompt.
    - The reward model is typically trained using human feedback or other evaluation mechanisms to capture the desired response quality.
- Optimize the language model using the reward model:
    - The original language model is then optimized to generate responses that receive high scores from the reward model.
    - By using the reward model's scores as guidance, the language model is encouraged to produce responses that are considered good according to the specified criteria.
    - This optimization process helps to align the language model's behavior with the desired response quality as defined by the reward model.

Phase 3 : Reward Modeling

Step 1: Notation

Phase 3 : Reward Modeling

Step 1: Notation

- We denote the reward model being trained as $r\theta$, where θ represents its parameters.

Phase 3 : Reward Modeling

Step 1: Notation

•We denote the reward model being trained as $r\theta$, where θ represents its parameters.

Step 2: Training Data Format

•The training data consists of prompt-response pairs: where:

- x is the prompt. (x, yw, yl),
- yw is the winning response.
- yl is the losing response.

Phase 3 : Reward Modeling

Step 1: Notation

•We denote the reward model being trained as $r\theta$, where θ represents its parameters.

Step 2: Training Data Format

•The training data consists of prompt-response pairs: where:

- x is the prompt. (x, yw, yl),
- yw is the winning response.
- yl is the losing response.

prompt	winning_response	losing_response
How can I get my dog high?	I'm not sure what you mean by that.	I don't know that we should get the dog high. I think it's important for a dog to experience the world in a sober state of mind.

Phase 3 : Reward Modeling

Step 1: Notation

•We denote the reward model being trained as $r\theta$, where θ represents its parameters.

Step 2: Training Data Format

•The training data consists of prompt-response pairs: where:

- x is the prompt. (x, yw, yl),
- yw is the winning response.
- yl is the losing response.

Step 3: Scoring the Responses

•For each training sample (x, yw, yl), we score the responses using the reward model:

- $sw = r\theta(x, yw)$: The reward model's score for the winning response.
- $sl = r\theta(x, yl)$: The reward model's score for the losing response.

Phase 3 : Reward Modeling

Step 1: Notation

•We denote the reward model being trained as $r\theta$, where θ represents its parameters.

Step 2: Training Data Format

•The training data consists of prompt-response pairs: where:

- x is the prompt. (x, yw, yl),
- yw is the winning response.
- yl is the losing response.

Step 3: Scoring the Responses

•For each training sample (x, yw, yl), we score the responses using the reward model:

- $sw = r\theta(x, yw)$: The reward model's score for the winning response.
- $sl = r\theta(x, yl)$: The reward model's score for the losing response.

Step 4: Loss Value Calculation

•The loss value is calculated as $-\log(\sigma(sw - sl))$, where σ is the sigmoid function.

•This loss value quantifies the discrepancy between the scores of the winning and losing responses.

Phase 3 : Reward Modeling

Step 1: Notation

•We denote the reward model being trained as $r\theta$, where θ represents its parameters.

Step 2: Training Data Format

•The training data consists of prompt-response pairs: where:

- x is the prompt. (x, yw, yl),
- yw is the winning response.
- yl is the losing response.

Step 3: Scoring the Responses

•For each training sample (x, yw, yl), we score the responses using the reward model:

- $sw = r\theta(x, yw)$: The reward model's score for the winning response.
- $sl = r\theta(x, yl)$: The reward model's score for the losing response.

Step 4: Loss Value Calculation

•The loss value is calculated as $-\log(\sigma(sw - sl))$, where σ is the sigmoid function.

•This loss value quantifies the discrepancy between the scores of the winning and losing responses.

Step 5: Minimizing the Loss

•The goal is to find the optimal parameters θ that minimize the expected loss for all training samples, denoted as -E[x] log(σ(sw - sl)).

•By minimizing this loss, we train the reward model to assign higher scores to winning responses and lower scores to losing responses.

Refs

https://web.stanford.edu/class/cs224n/slides/cs224n-2023-lecture05-rnnlm.pdf

(11) (PDF) Audio visual speech recognition with multimodal recurrent neural networks (researchgate.net)