Compression for AGI

Yuxuan Lu

Nov. 3 2023

Northeastern Human-Centered AI Lab

Takeaways

- Generative models are *Lossless* compressors
    - "ChatGPT Is a Blurry JPEG of the Web"? No!
- LLMs are *State-of-the-Art* text compressors
    - comparing to deflate(gzip), Zstd, etc.
- Re-think about the training objective of foundation models

Table of Contents

Revisit the Training Process of LLMs

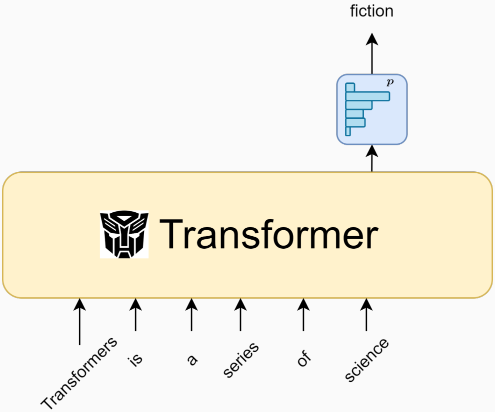

Figure 1: Transformer: A Black Box – step 1

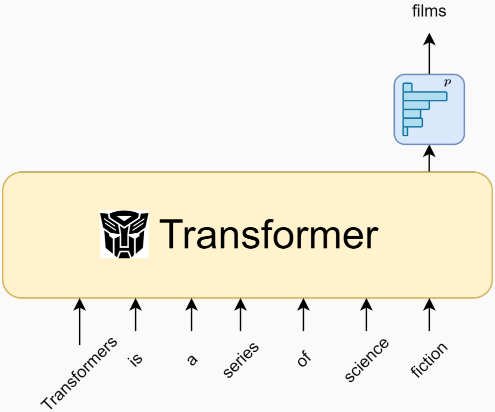

Figure 2: Transformer: A Black Box – step 2

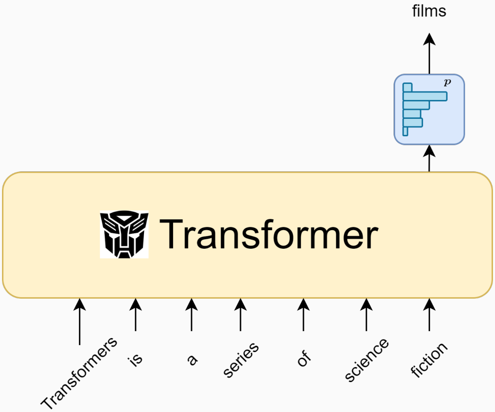

Figure 3: Transformer: A Black Box – step 2

Transformer: A Black Box

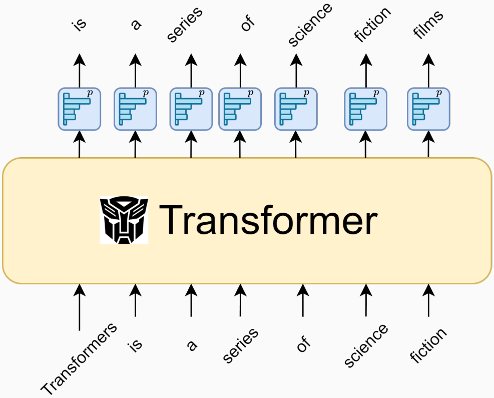

Figure 4: Transformer: A Black Box – training

Transformer: A Black Box

- We have a sequence: $X = [x_1, x_2, \ldots, x_n]$
- We put this sequence into LLM and get a sequence of probabilities:
  - $p_i = P(x_i|x_{<i})$
- We want p_i approaches 1, so we minimize the following loss during pre-training:
  - $L = \sum_{i=1}^{n} -\log P(x_i|x_{<i})$

Why are LLMs *Lossless* compressors

- Naturally, you would assume an LLM is a *lossy* compressor
  - That turns trainig corpus to model parameters
  - For LLaMa, the training dataset is 5.6TB
  - And the 65 billion parameters takes about 130GB of storage
    - So 43x compression rate?
    - Lossy!

Northeastern University

- Naturally, you would assume an LLM is a *lossy* compressor
    - That turns trainig corpus to model parameters
    - For LLaMa, the training dataset is 5.6TB
    - And the 65 billion parameters takes about 130GB of storage
        - So 43x compression rate?
        - Lossy!
- Actually, LLaMa can compress the entire 5.6TB of training corpus to 397.3 GB losslessly
    - 14x compression
    - Best text compressor: 8.7x compression

- To compress the *entire training corpus* $\mathcal{C}$ *losslessly*, we only need *the CODE to train the models* and $\sum_{i \in \mathcal{C}} -\log P(x_i | x_{<i})$ bits of imformation.
  - The result size (after compression) *ISN'T* related the number of parameters!

- To compress the *entire training corpus $\mathcal{C}$ losslessly*, we only need *the CODE to train the models* and $\sum_{i\in\mathcal{C}} -\log P(x_i|x_{<i})$ bits of imformation.
    - The result size (after compression) *ISN'T* related the number of parameters!
- But, HOW?

- Imagine Alice is trying to send some text to Bob through a telephone wire
- Alice needs to encode the data to "something"
- Bob needs to decode the "something" back to data
- So we need an "encoding" algorithm and a "decoding" algorithm

- Let's say we have a vocabulary of $\mathcal{V}$ (which should be defined in the training code), and we want to encode the first token x_1.

- Let's say we have a vocabulary of $\mathcal{V}$ (which should be defined in the training code), and we want to encode the first token x_1.
- First, we initialize the model and fed it the training data
    - and we can get the probability distribution of the first token p and an index $x_1 \in \mathcal{V}$

- Let's say we have a vocabulary of $\mathcal{V}$ (which should be defined in the training code), and we want to encode the first token x_1.
- First, we initialize the model and fed it the training data
    - and we can get the probability distribution of the first token p and an index $x_1 \in \mathcal{V}$
- With arithmetic encoding, we can encode the index x_1 with $-\log_2 p_{x_1}$ bits
    - Rather than $-\log_2 \frac{1}{|\mathcal{V}|}$ bits

- Let's say we have a vocabulary of $\mathcal{V}$ (which should be defined in the training code), and we want to encode the first token x_1.
- First, we initialize the model and fed it the training data
  - and we can get the probability distribution of the first token p and an index $x_1 \in \mathcal{V}$
- With arithmetic encoding, we can encode the index x_1 with $-\log_2 p_{x_1}$ bits
  - Rather than $-\log_2 \frac{1}{|\mathcal{V}|}$ bits
- We send those $-\log_2 p_{x_1}$ bits to Bob
  - Let's call it z_1

- Let's say we have a vocabulary of $\mathcal{V}$ (which should be defined in the training code), and we want to encode the first token x_1.
- First, we initialize the model and fed it the training data
  - and we can get the probability distribution of the first token p and an index $x_1 \in \mathcal{V}$
- With arithmetic encoding, we can encode the index x_1 with $-\log_2 p_{x_1}$ bits
  - Rather than $-\log_2 \frac{1}{|\mathcal{V}|}$ bits
- We send those $-\log_2 p_{x_1}$ bits to Bob
  - Let's call it z_1
- ... and update the model to minimize the loss, and repeat these steps for every data

- Let's say we have a vocabulary of $\mathcal{V}$ (which should be defined in the training code), and we want to encode the first token x_1.
- First, we initialize the model and fed it the training data
    - and we can get the probability distribution of the first token p and an index $x_1 \in \mathcal{V}$
- With arithmetic encoding, we can encode the index x_1 with $-\log_2 p_{x_1}$ bits
    - Rather than $-\log_2 \frac{1}{|\mathcal{V}|}$ bits
- We send those $-\log_2 p_{x_1}$ bits to Bob
    - Let's call it z_1
- ... and update the model to minimize the loss, and repeat these steps for every data

- Bob retrives the code for training the model

- Bob retrives the code for training the model
- And he can initialize the exact same model (with random seed defined in the code)

Northeastern University

- Bob retrives the code for training the model
- And he can initialize the exact same model (with random seed defined in the code)
- So he will have the *exact same distribution* p for the first token
    - He then can decode the token with p and z_1 with arithmetic decoding

Northeastern University

- Bob retrives the code for training the model
- And he can initialize the exact same model (with random seed defined in the code)
- So he will have the *exact same distribution* p for the first token
    - He then can decode the token with p and z_1 with arithmetic decoding
- And he will run the training loop to update the model with x_1
- Note that the model parameters isn't transmitted through the phone wire

- In the above process, the model parameter is always synced between bob and alice
- With arithmetic encoding, we can use fewer bits to encode something have a higher probability in a distribution.
  - If the distribution is uniform, $-\log_2 p_{x_i} = -\log_2 \frac{1}{|\mathcal{V}|}$, which is same as naive storage
  - And as the model is continually training, it can predict the next token with higher and higher probability.
- Bob can re-construct the entire training corpus $\mathcal{C}$ with $\sum_{i \in \mathcal{C}} -\log P(x_i | x_{<i})$ bits of information
  - Which is exactly the same with the sum of training loss on all tokens!

Rethink the goal for foundation models

Classic Chinese Room thought experiment

- If a computer program translates English and Chinese using an oracle comprising all possible combinations of Chinese and English combinations.
    - Does it have understanding of translation?

- If a computer program translates English and Chinese using an oracle comprising all possible combinations of Chinese and English combinations.
    - Does it have understanding of translation?
    - It has the *least* understanding of translation
- What if it do it by a set of rules?
    - it have *some* understanding of translation.

Classic Chinese Room thought experiment

- If a computer program translates English and Chinese using an oracle comprising all possible combinations of Chinese and English combinations.
    - Does it have understanding of translation?
    - It has the *least* understanding of translation
- What if it do it by a set of rules?
    - it have *some* understanding of translation.
- If we can make the rule set smaller, it will generalise better.

Lossy v.s. Lossless

- “Lossy” compression means the model “remembers” everything in the training dataset
  - BAD generalize ability
- “Lossless” compression means the model can better predict unseen
  - Better compression means GOOD generalize ability
    - Because *EVERY* example is unseen

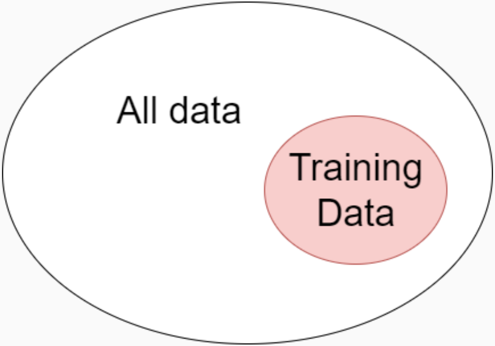

Figure 5: All data V.S. Training Data

- For foundation models, what we want is good generalization ability
    - i.e. ability to generate or write *UNSEEN* samples.

A recipe for perception

- Collet all useful perceptual information
- Learn to compress it as best as possible with a powerful fundation model
    - Better architecture
    - Scale
    - Tool use
    - ...