Generative Models for Text

Yuchen Liang

REU Summer 2025

The Ohio State University
COLLEGE OF ENGINEERING

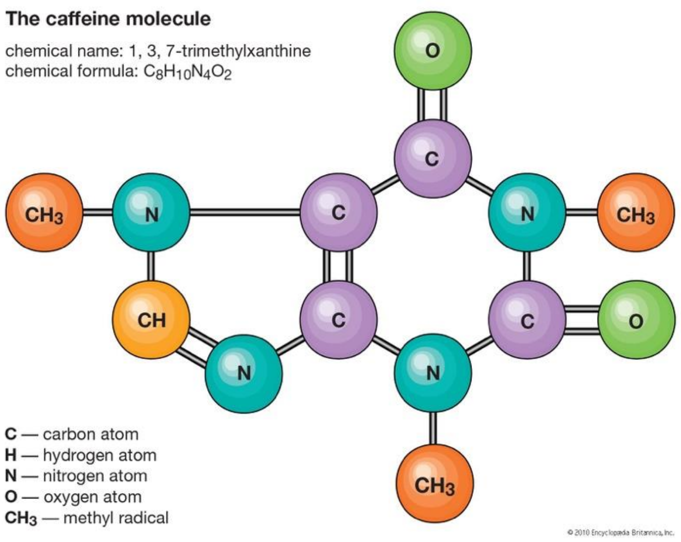
Agenda

1. **Text and Language Model**
2. Large Language Models
3. Model Adaptations

What are texts?

...shehasnootherneurologicsymptomsnonumbnessortinglingshedeniesanyvisualcha
ngespastmedicalhistorygallbladderremovalpastsurgicalhistorydiabetesrheumatoida
rthritishypertensiongerdandhypothyroidismmedicationsadvairalbuterolallopurinol
aspirinclobetasolfolicacidfosamaxlevoxyllisinoprilmetforminomeprazoleplaquenilp
rednisonetestosteroneverapamilallergiesnoknowndrugallergiessocialhistorythepati
entismarriedwithchildshe**doesnotsmoke**shedoesnotdrinkshedoesnotuserecreation
aldrugssheweighspoundsandisinchestallfamilyhistorynegativeforbrainaneurysmoro
theraneurysmitwasalsonegativeforheartdiseasehighcholesterolandhypertensionand
negativefordiabetesreviewofsystemsthepatientispositiveforhypertensionswellingint
hehandsorfeetlegpainwhilewalkingasthmapneumoniashortnessofbreathgastritisulc
ersdiabetesthyroiddiseaseurinarytractinfectionsandthosesymptomsrelatedtothepre
sentillnessthedetailsofthereviewofsystemswerereviewedwiththepatientandareinclu
dedintheneurosurgicalhealthhistoryquestionnairepainthepatienthasepisodicjointpa
inthatistreatedwithtylenolthepatientdoesnothaveanynutritionalconcernsshedoesno
thaveanysafetyconcernsphysicalexamination...

Natural language

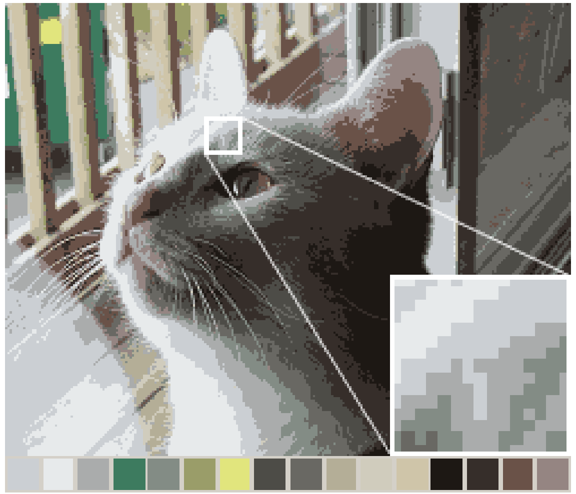


Molecules

Music notes

Coding

Images



What are texts? (cont.)

Two important properties of text:

1. Each token comes from a **finite** number of categories (not like Gaussian)
   - Token: the smallest divisible element in your algorithm
   - E.g., word, character, molecule, pixel, consonant/vowel…
   - Category: token representation (word embedding)
2. (Typically) These categories are **not ordered**

a	1
an	2
apple	3
the	4
…	…

What is a Language Model?

- Definition: A Language Model is a **probabilistic** characterization of the tokens

$$p(w_1, \dots, w_n)$$

- Generative model is a Language model that can **generate**!

Autoregressive Model

- Definition: An Autoregressive model is a generative model where the next token only depends on previous tokens

$$p(w_1, \dots, w_n) = p(w_1)p(w_2|w_1) \cdots p(w_n|w_1, \cdots, w_{n-1})$$

- Useful with **sequential** inputs: speech, text, etc.
  - …but not necessarily. Remember, it's just a model!

Language Modeling – example

- Let's calculate the probability of "the big dog"

$$P(\text{the, big, dog}) = P(\text{the})\, P(\text{big}|\text{the})\, P(\text{dog}|\text{the, big})$$

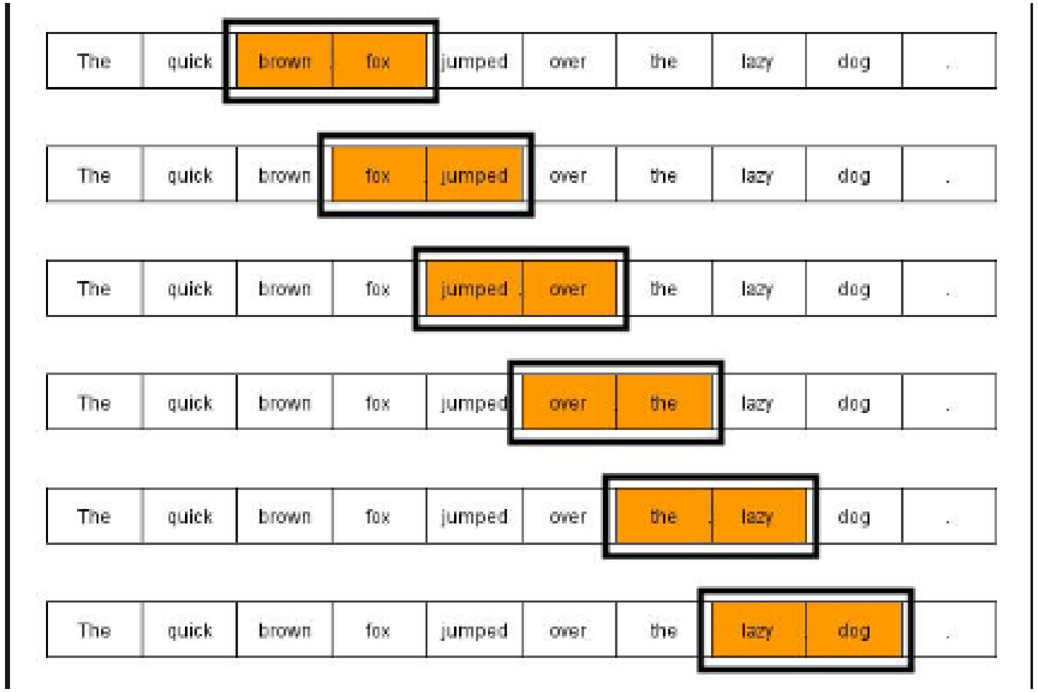
- Terminologies:
  - Unigrams: P(the)
  - Bigrams: P(big|the)
  - Trigrams: P(dog|the, big)

Estimate these probs using your text corpus…

Example 1: N-Gram

- N-Gram: the “go-to method” before deep learning
  - N-Gram is an **autoregressive** model
  - Sometimes, Bi-Gram is enough…

- Issues with N-Gram: cannot capture **long-range dependence**

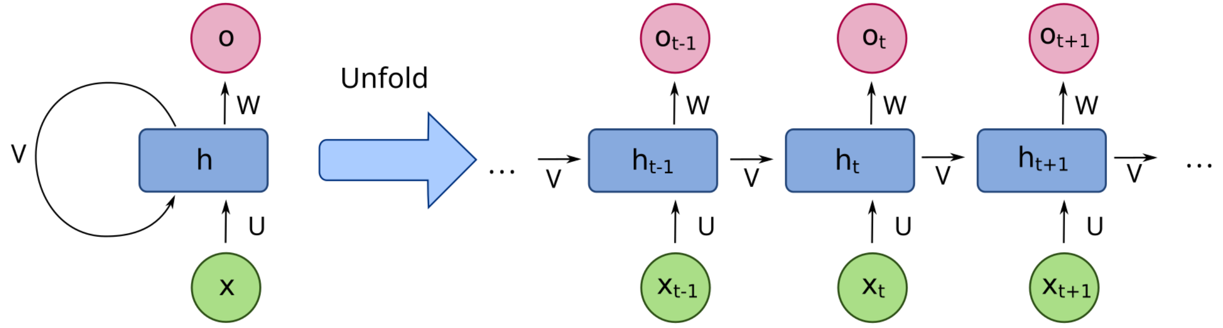


https://www.depends-on-the-definition.com/introduction-n-gram-language-models/

Example 2: RNN

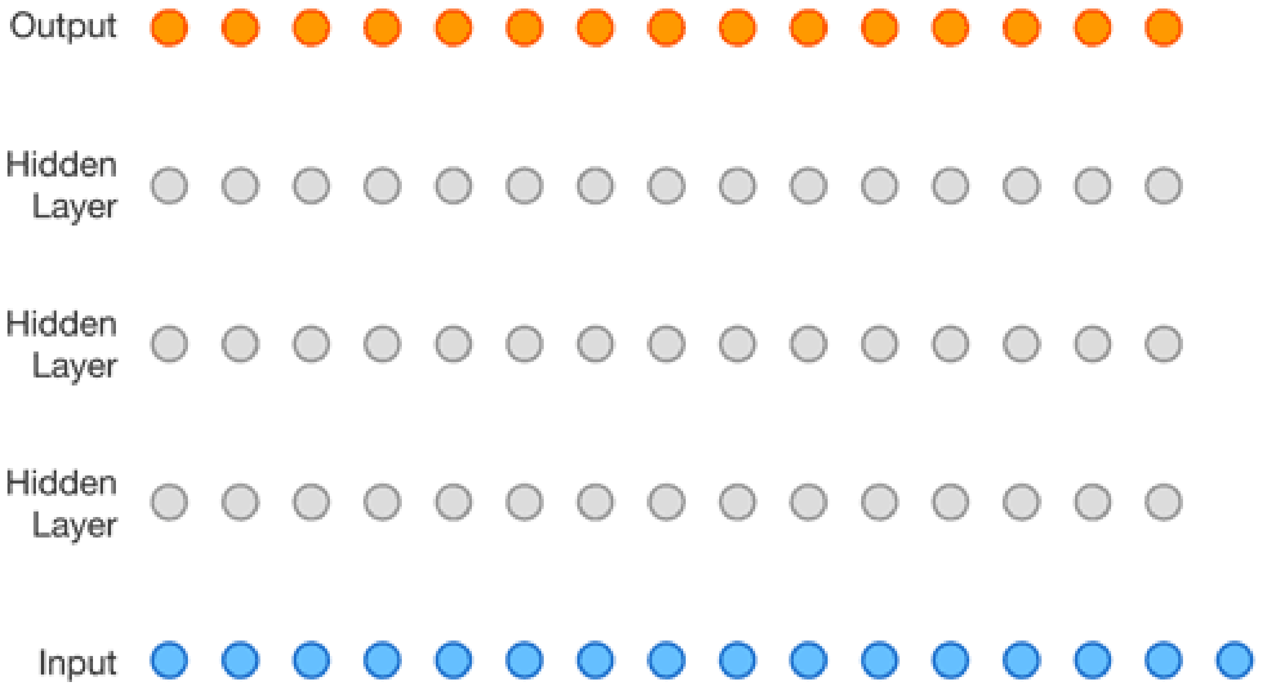
- A family of methods: RNN, LSTM, GRU, Bi-LSTM…

- Issues:
  1. Only covers mid-range dep (long-range still hard)
  2. Very inefficient to train (sequential nature)



Example 3: WaveNet

Generative model of speech signals



Text to Speech

- Parametric
- Concatenative
- WaveNet

- Unconditional
- Music

van den Oord et al, 2016c

N-Gram and BLEU Score

- BLEU (bilingual evaluation understudy): a quality measure of machine-translated text (2001)

$$BLEU_w(\hat{S};S) := BP(\hat{S};S) \cdot \exp\left(\sum_{n=1}^{\infty} w_n \ln p_n(\hat{S};S)\right)$$

- $\hat{S} = (\hat{y}_1, \ldots, \hat{y}_M)$: **candidate** corpus; $S = (S_1, \ldots S_M)$: **reference** corpus
- BP: brevity penalty (only for short candidates)
- p_n: (Idealy) captures how many **n-grams** in the reference are reproduced by the candidate sentence

From Wiki

Agenda

1. Text and Language Model
2. **Large Language Models**
3. Model Adaptations

Transformer

"[Transformer is] a model architecture **eschewing recurrence** and instead relying entirely on an **attention mechanism** to draw **global dependencies** between input and output."

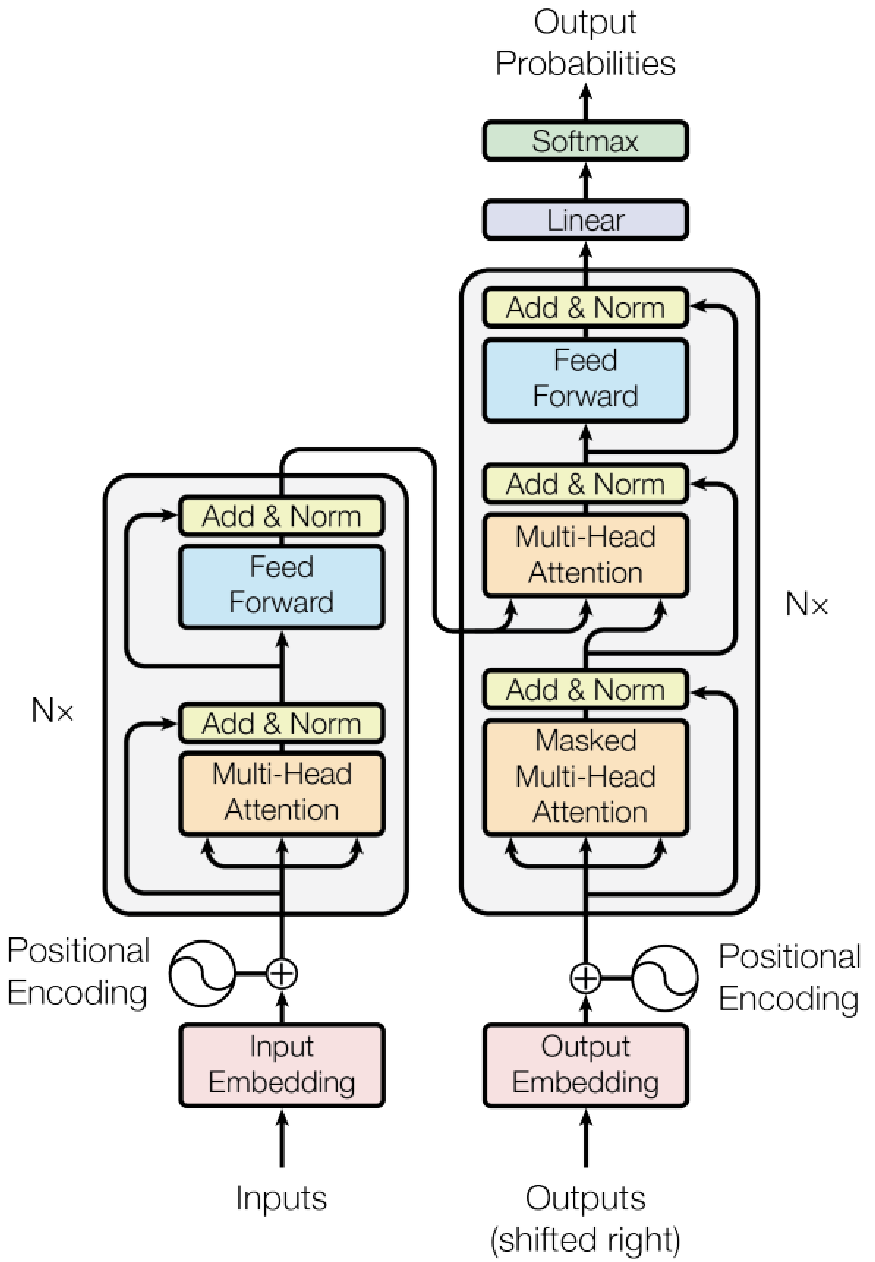


Figure 1: The Transformer - model architecture.

https://arxiv.org/pdf/1706.03762

Transformer (cont.)

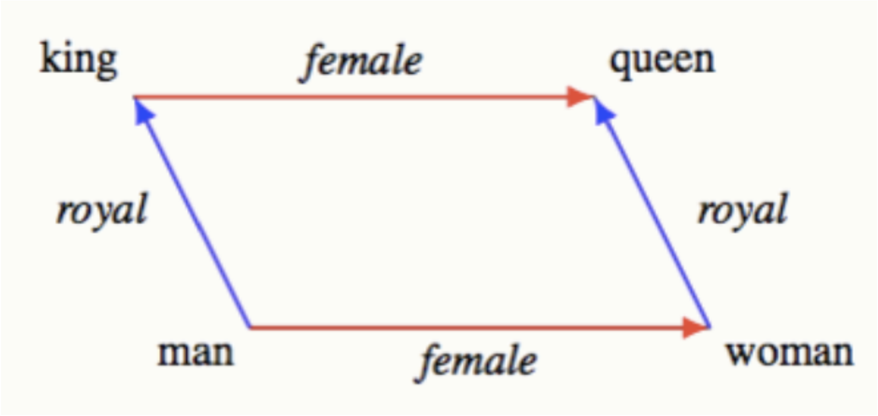
- Basic structure
  - Encoder: words -> hidden-state rep (**trained in parallel**)
  - Decoder: hidden-state rep -> probabilities (of words or labels)
- Transformer generates a contextual word embedding
  - Word Embedding: numerical representation of each word
  - Contextual: the mapping of a word depends on surrounding words
  - How? Thru the **self-attention** mechanism

(Static) Word Embedding Example

- Question: What is King - Man + Woman?

- Answer: **Queen**!

- …which is the answer from **Word2vec**



https://kawine.github.io/blog/nlp/2019/06/21/word-analogies.html

Transformer as Language Model

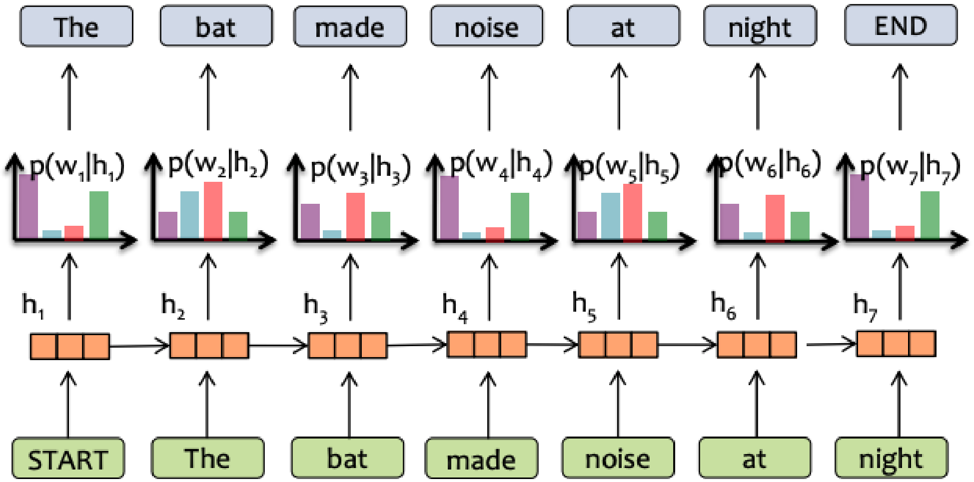


Diagram of RNN Generation

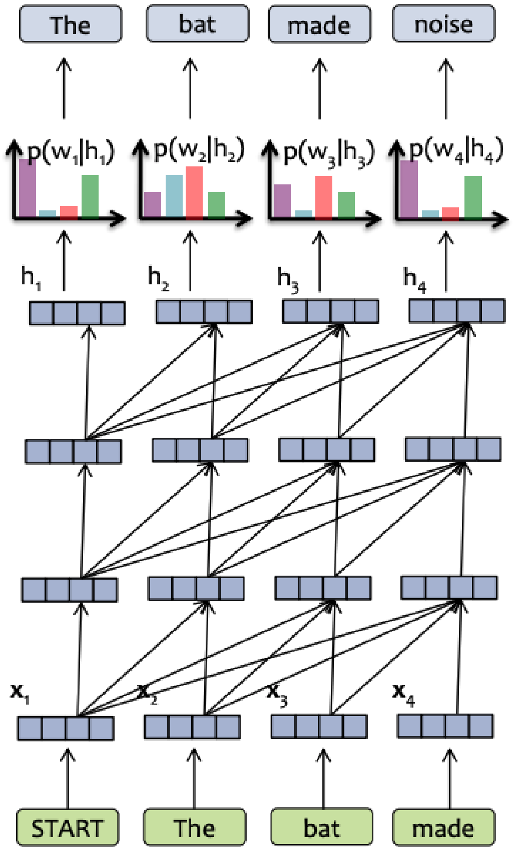
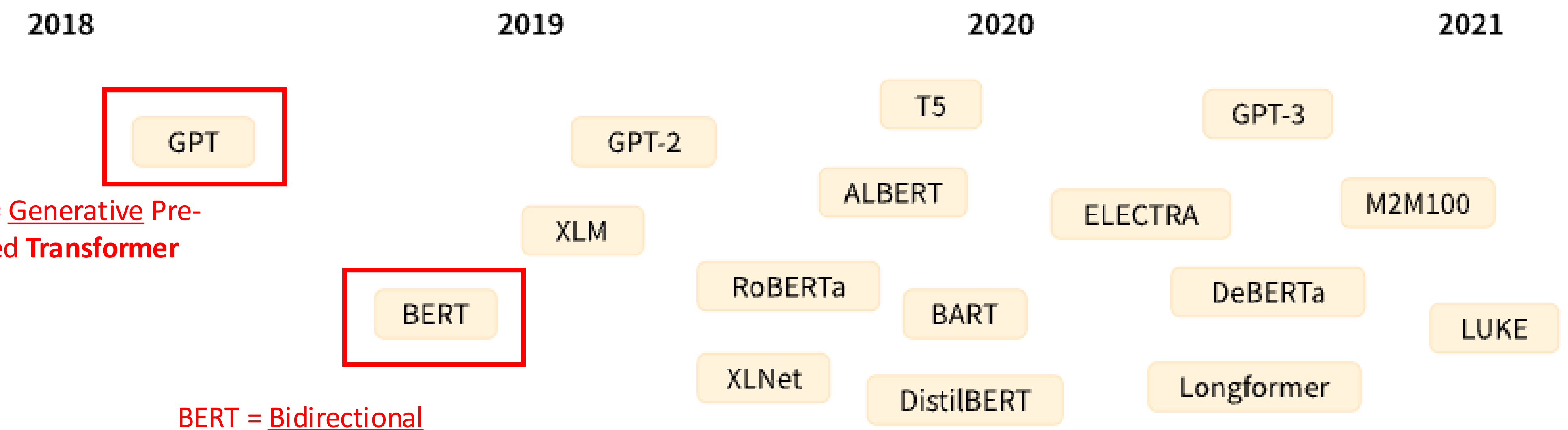


Diagram of GPT-style (decoder-only) Transformer Generation

https://www.cs.cmu.edu/~mgormley/courses/10423//slides/lecture2-transformer.pdf

Transformer (Large) Language Models



GPT = Generative Pre-trained **Transformer**

BERT = Bidirectional Encoder Representations from **Transformers**

https://huggingface.co/learn/llm-course/chapter1/4

GPT

- Transformer is just a model. How to use it?
- Generative Pre-Training (GPT): from OpenAI (Radford et al., 2018)
- (Pre-)training: conditional language modeling (CLM) -> **next-word** pred

$$L_1(\mathcal{U}) = \sum_i \log P(u_i | u_{i-k}, \ldots, u_{i-1}; \Theta)$$

- Fine-tuned for specific tasks…

$$P(y|x^1, \ldots, x^m) = \texttt{softmax}(h_l^m W_y). \qquad L_2(\mathcal{C}) = \sum_{(x,y)} \log P(y|x^1, \ldots, x^m).$$

- Later: GPT-x, ChatGPT (finetuned from GPT-3.5)

https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf

BERT

- Bidirectional Encoder Representations from Transformers (BERT): from Google (Devlin et al., 2019)
- (Pre-)training objectives
  1. Masked language modeling (MLM) -> use masks
  2. Next-sentence prediction (NSP)
- Then, fine-tuned for specific tasks
- Advantages (vs. GPT): light-weighted, faster, better for classification
- Later: RoBERTa, **DeBERTa (usually enough)**, ModernBERT

Types of Large Language Models

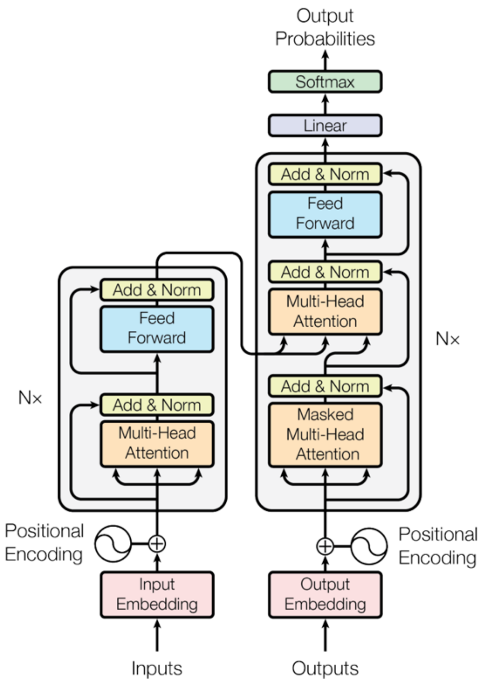
- BERT-like (Encoder-only) Models
  - **Bi-directional** structure
  - Useful for sentence classification (e.g., sentiment analysis)
- GPT-like (Decoder-only) Models (e.g., also Llama)
  - Uni-directional (i.e., **autoregressive**) structure
  - Suitable for generation
- Combined Models: T5, BART, CMLM, etc.
  - BERT-like encoder + GPT-like decoder
  - Useful for seq2seq tasks: summarization, translation, etc.

https://huggingface.co/learn/llm-course/chapter1/5

Combined Model Example

BERT

Encoder



GPT

Decoder

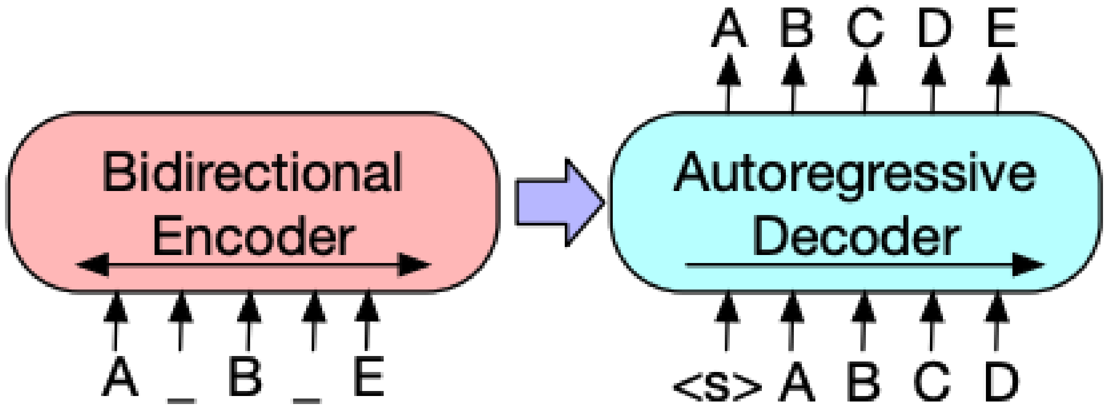


Diagram of BART

https://huggingface.co/learn/llm-course/chapter1/5

What can LLMs do?

1. Text generation (from a prompt)
2. Text classification
3. Summarization
4. Translation
5. Zero-shot classification
6. Feature extraction

All tasks have available **pipelines** on Hugging Face!

Language Generation

Custom prompt

To get an A+ in deep generative models, students have to

P(next word | previous words)

Completion

To get an A+ in deep generative models, students have to be willing to work with problems that are a whole lot more interesting than, say, the ones that most students work on in class. If you're a great student, the question above can be avoided and you'll be able to do great work, but if you're not, you will need to go beyond the basics before getting good.

Now to be clear, this advice is not just for the deep-learning crowd; it is good advice for any student who is taking his or her first course in machine learning.

The key point is that if you have a deep, deep brain of a computer scientist, that's just as important to you.

Radford et al., 2019
Demo from talktotransformer.com

Machine Translation

Conditional generative model P(English text | Chinese text)

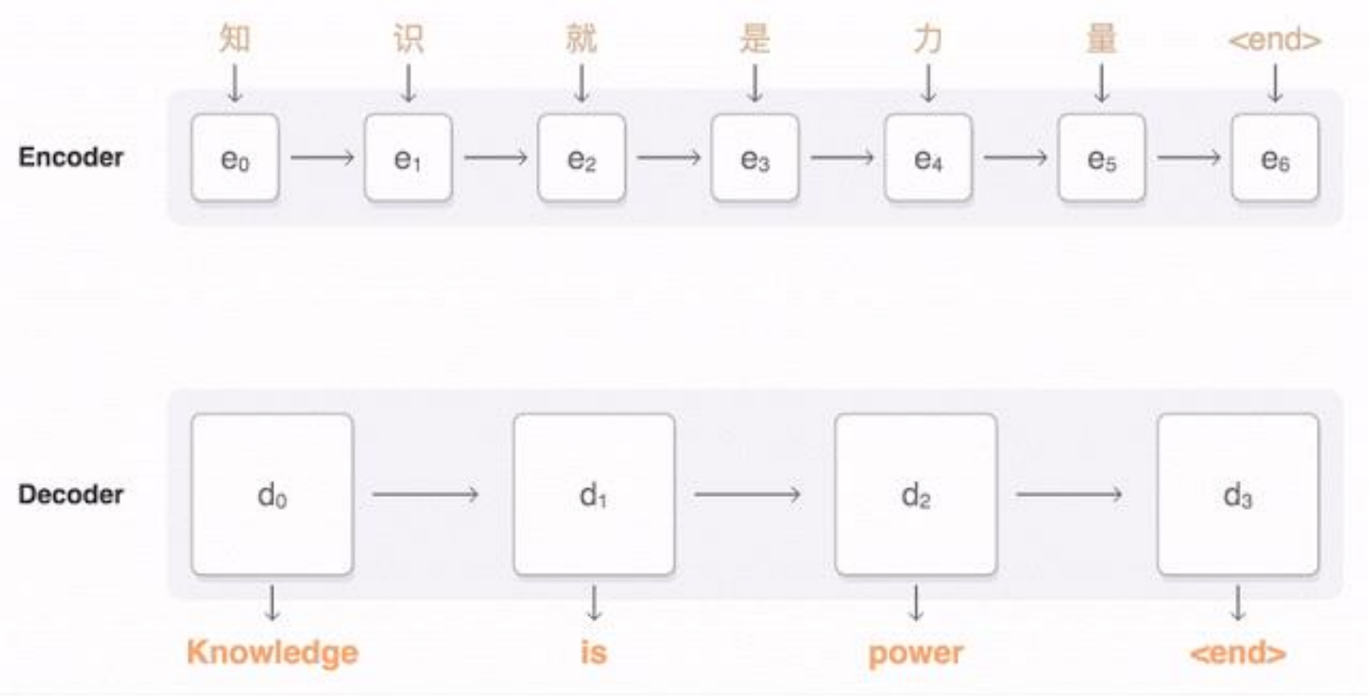


Figure from Google AI research blog.

Issue: Hallucinations

Definition: a tendency for LLMs to **fabricate information** which **sounds like facts**

❌ **Hallucinated Response:**

User: *Who won the Nobel Prize in Physics in 2022?*

LLM: *The 2022 Nobel Prize in Physics was awarded to Dr. Maria Thompson for her groundbreaking work on quantum teleportation.*

✅ **Reality:**

The **2022 Nobel Prize in Physics** was awarded to **Alain Aspect, John F. Clauser, and Anton Zeilinger** for experiments with entangled photons, establishing the violation of Bell inequalities and pioneering quantum information science.

Agenda

1. Text and Language Model
2. Large Language Models
3. **Model Adaptations**

- Resources
  - https://huggingface.co/learn/llm-course/chapter1
  - https://www.cs.cmu.edu/~mgormley/courses/10423/

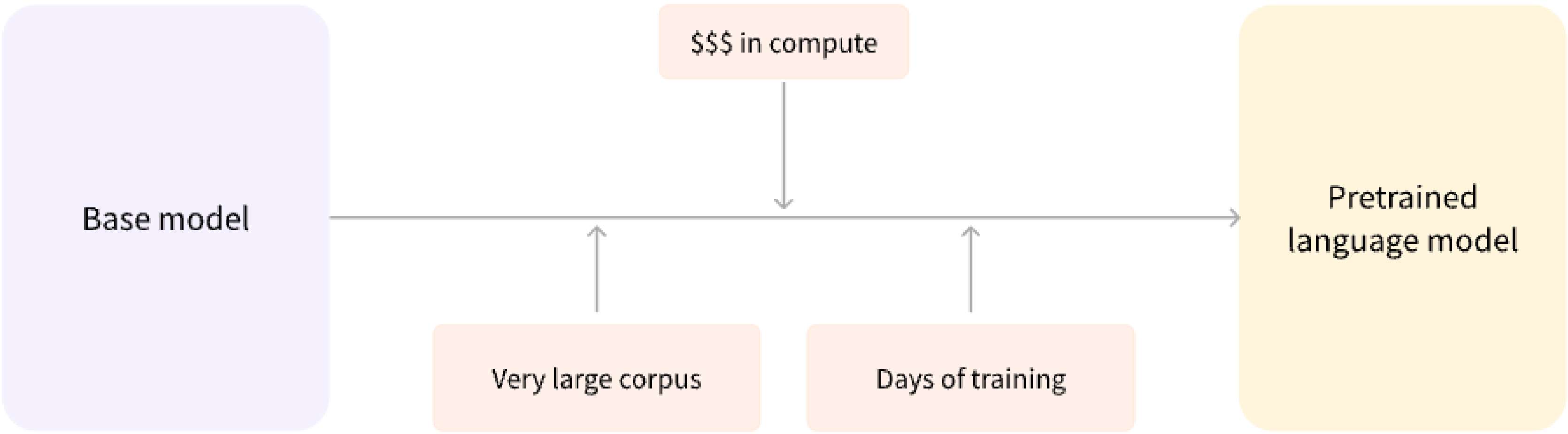
Why Model Adaptation?

Model	Creators	Year of release	Training Data (# tokens)	Model Size (# parameters)
GPT-2	OpenAI	2019	~10 billion (40Gb)	1.5 billion
GPT-3 (cf. ChatGPT)	OpenAI	2020	300 billion	175 billion
PaLM	Google	2022	780 billion	540 billion
Chinchilla	DeepMind	2022	1.4 trillion	70 billion
LaMDA (cf. Bard)	Google	2022	1.56 trillion	137 billion
LLaMA	Meta	2023	1.4 trillion	65 billion
LLaMA-2	Meta	2023	2 trillion	70 billion
GPT-4	OpenAI	2023	?	? (1.76 trillion)
Gemini (Ultra)	Google	2023	?	? (1.5 trillion)
LLaMA-3	Meta	2024	15 trillion	405 billion

https://www.cs.cmu.edu/~mgormley/courses/10423//slides/lecture10-peft.pdf

Model Pre-training and Fine-tuning

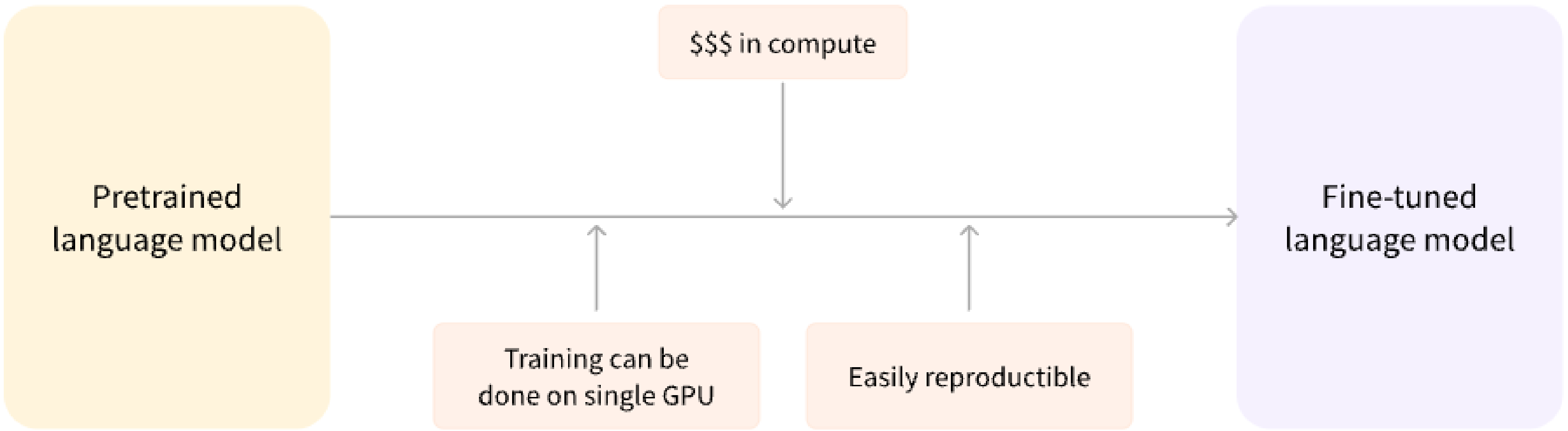
- Pre-training: Train a model **from scratch**
  - … which results in a **foundation** model
  - E.g., GPT, Stable Diffusion, …



https://huggingface.co/learn/llm-course/chapter1/4

Model Pre-training and Fine-tuning (cont.)

- Fine-tuning: Training based on a pre-trained model **for specific tasks**
  - … usually using a **customized dataset**



https://huggingface.co/learn/llm-course/chapter1/4

Parameter-Efficient Fine-Tuning (PEFT)

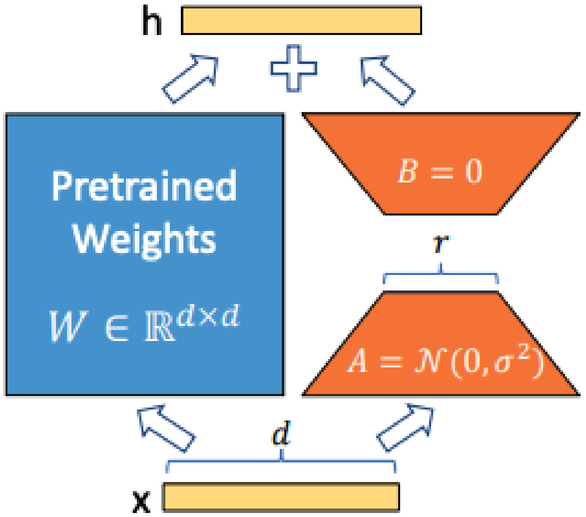


Figure 1: Our reparametrization. We only train A and B.

LoRA (Low-rank Adaptation) for Transformer

ControlNet for Diffusion

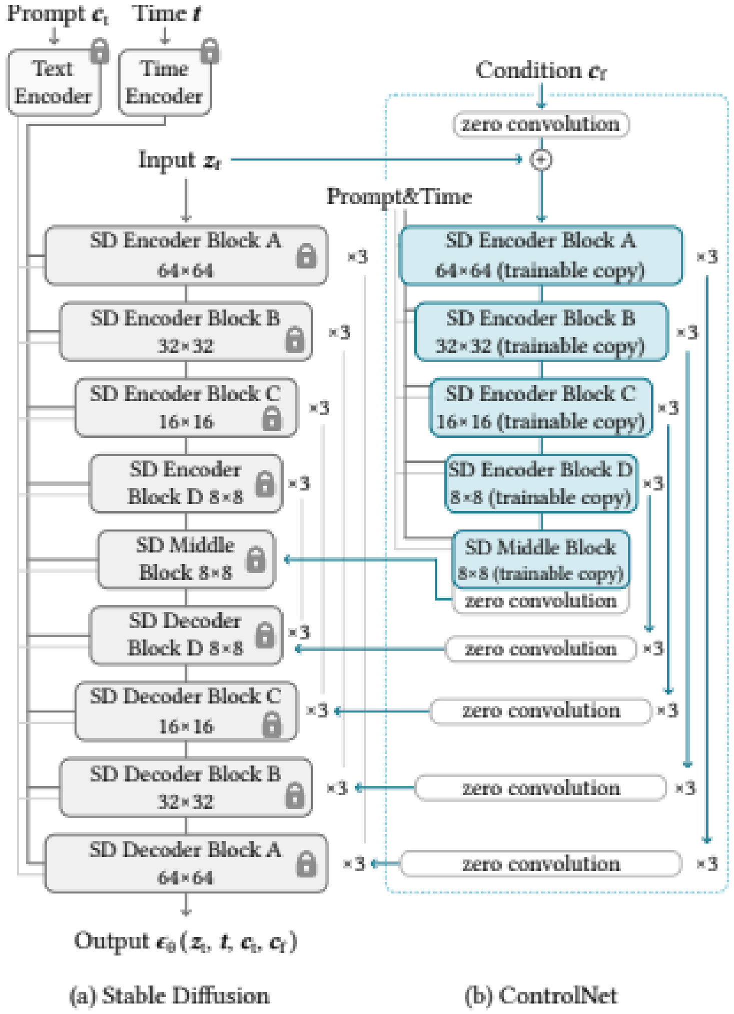


Figure 3: Stable Diffusion's U-net architecture connected with a ControlNet on the encoder blocks and middle block. The locked, gray blocks show the structure of Stable Diffusion V1.5 (or V2.1, as they use the same U-net architecture). The trainable blue blocks and the white zero convolution layers are added to build a ControlNet.

https://arxiv.org/pdf/2106.09685
https://arxiv.org/pdf/2302.05543

Can we avoid extra-training at all?

Option A: Supervised fine-tuning

- **Definition:** fine-tune the LLM on the training data using…
  - a standard supervised objective
  - backpropagation to compute gradients
  - your favorite optimizer (e.g. Adam)
- **Pro:** fits into the standard ML recipe
- **Pro:** still works if N is large
- **Con:** backpropagation requires ~3x the memory and computation time as the forward computation
- **Con:** you might not have access to the model weights at all (e.g. because the model is proprietary)

Option B: In-context learning

- **Definition:**
  1. feed training examples to the LLM as a prompt
  2. allow the LLM to infer patterns in the training examples during inference (i.e. decoding)
  3. take the output of the LLM following the prompt as its prediction
- **Con:** the prompt may be very long and Transformer LMs require $O(N^2)$ time/space where N = length of context
- **Pro:** no backpropagation required and only one pass through the training data
- **Pro:** does not require model weights, only API access

https://www.cs.cmu.edu/~mgormley/courses/10423//slides/lecture9-vae-in-context.pdf

Fine-tuning vs. In-context Learning

(a) RTE

		FT						
		125M	350M	1.3B	2.7B	6.7B	13B	30B
ICL	**125M**	−0.00	0.01	0.02	0.03	0.12	0.14	0.09
	350M	−0.00	0.01	0.02	0.03	0.12	0.14	0.09
	1.3B	−0.00	0.01	0.02	0.03	0.12	0.14	0.09
	2.7B	−0.00	0.01	0.02	0.03	0.12	0.14	0.09
	6.7B	−0.00	0.01	0.02	0.03	0.12	0.14	0.09
	13B	−0.04	−0.02	−0.01	−0.00	0.09	0.11	0.05
	30B	−0.11	−0.09	−0.08	−0.08	0.02	0.03	−0.02

(b) MNLI

		FT						
		125M	350M	1.3B	2.7B	6.7B	13B	30B
ICL	**125M**	−0.00	0.00	0.02	0.01	0.10	0.11	0.07
	350M	−0.00	0.00	0.02	0.01	0.10	0.11	0.07
	1.3B	−0.01	−0.00	0.01	0.01	0.10	0.11	0.07
	2.7B	−0.01	−0.00	0.01	0.01	0.09	0.10	0.07
	6.7B	−0.01	−0.01	0.01	0.00	0.09	0.10	0.06
	13B	−0.03	−0.03	−0.02	−0.02	0.07	0.08	0.04
	30B	−0.07	−0.07	−0.05	−0.06	0.03	0.04	0.00

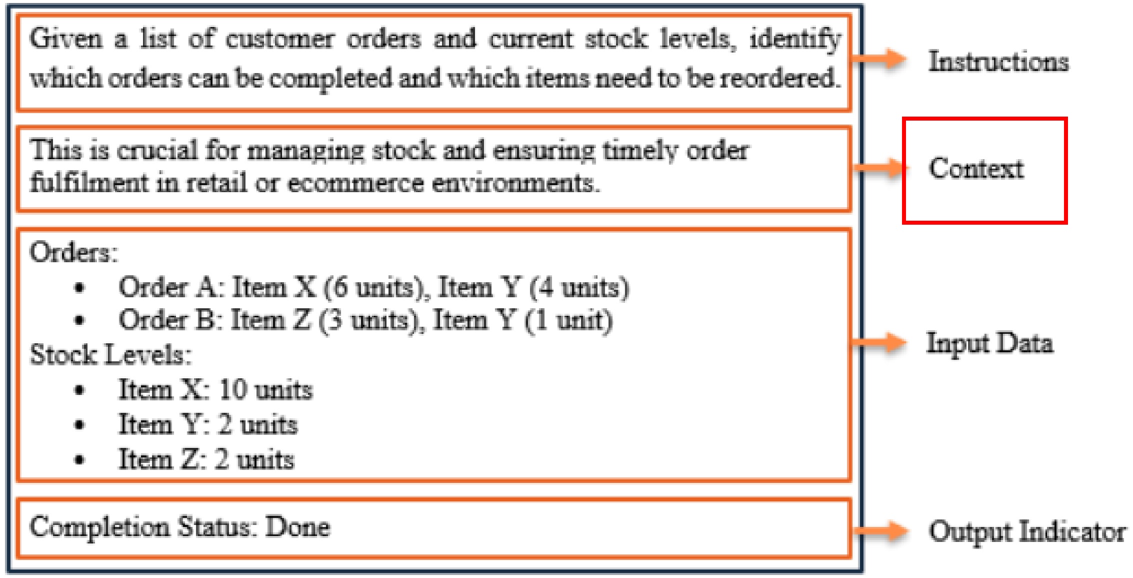
Table 1: Difference between average **out-of-domain performance** of ICL and FT on RTE (a) and MNLI (b) across model sizes. We use 16 examples and 10 random seeds for both approaches. For ICL, we use the `gpt-3` pattern. For FT, we use pattern-based fine-tuning (PBFT) and select checkpoints according to in-domain performance. We perform a Welch's t-test and color cells according to whether: ICL performs significantly better than FT, FT performs significantly better than ICL. For cells without color, there is no significant difference.

As of 2023…

https://aclanthology.org/2023.findings-acl.779.pdf

Prompt Engineering

Goal: Craft and refine your prompts to help the model generate specific outputs (an instance of ICL)



Providing context is essential

A prompt example that includes all four elements

Advanced Chain-of-Thought Prompting

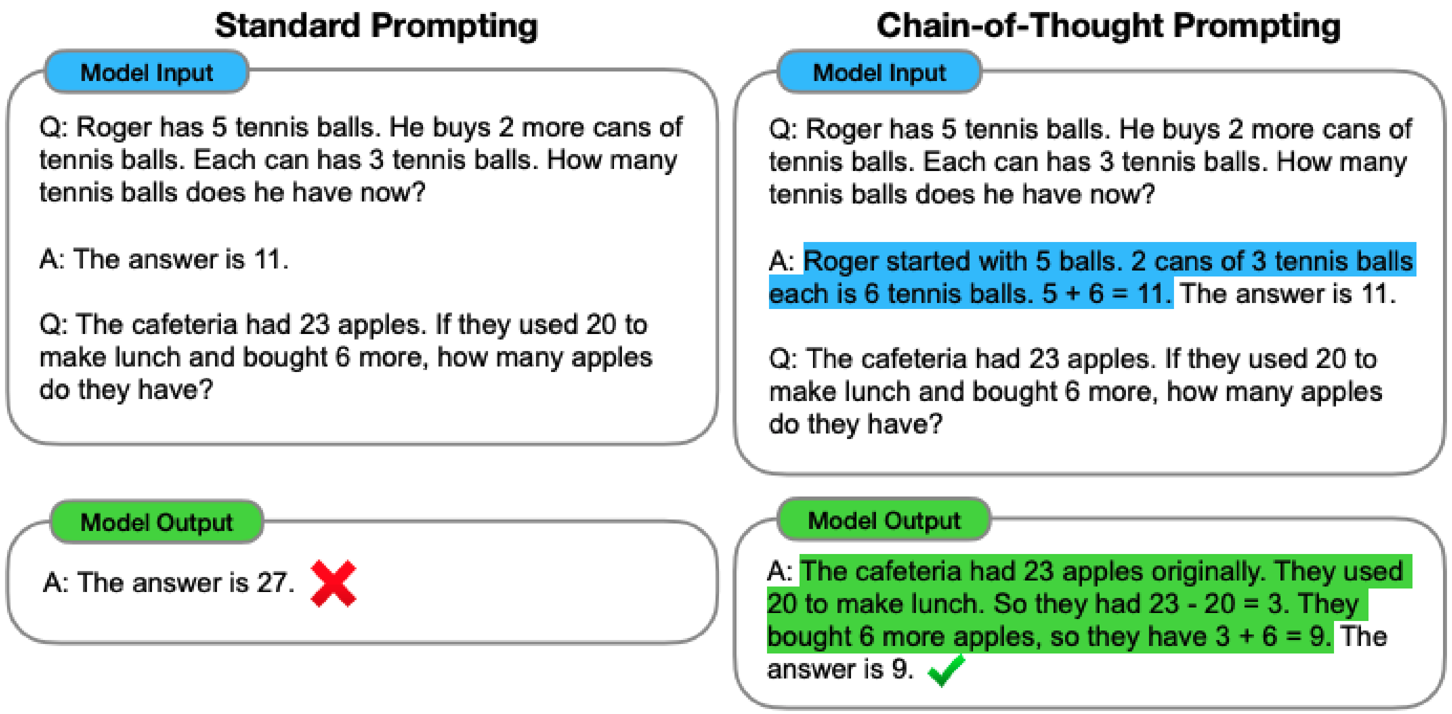


Figure 1: Chain-of-thought prompting enables large language models to tackle complex arithmetic, commonsense, and symbolic reasoning tasks. Chain-of-thought reasoning processes are highlighted.

https://arxiv.org/pdf/2201.11903

Homework

- Train your own GPT… (Ziyue)