Washington University in St. Louis
James McKelvey School of Engineering

CSE 561A: Large Language Models

Spring 2024

Lecture 1: Course Overview

Content

- **Course Logistics**
- Language Model Basics
- Covered Topics Preview

Course Logistics

- Instructor: Jiaxin Huang (jiaxinh@wustl.edu)
- Teaching Assistants:
  - Tovi Tu (jianhong.t@wustl.edu)
  - Nathan Suh (n.h.suh@wustl.edu)

- Course meeting times: 2:30pm – 3:50pm Tuesday / Thursday
- Location: Cupples I / 115

Course Logistics

- Course Syllabus: https://teapot123.github.io/CSE561A_2024sp/
- Canvas: https://wustl.instructure.com/courses/129974 (will be published soon)
- We will be using Canvas for announcements, discussions, and project report submissions.

Course Structure

- Advanced Research-Oriented Course
  - Pre-requisites: Students are expected to understand concepts in machine learning (CSE 417T/517A)
  - We will be teaching and discussing state-of-the-art papers about large language models
  - Lectures of fundamentals of Large Language Models (language model architecture and training framework)
  - Lectures of Large Language Model Capabilities, Applications and Issues
    - This part consists of a list of frontier research papers (will be released later), from which students will choose their interested papers to present in the class
    - Students who are not presenters are expected to participate in discussion and submit 3 preview questions
  - Guest lectures on frontier research topics

Grading

- 15% Class Participation
  - Regular class participation and discussion (10%)
  - Preview question submissions (5%)
- 30% Class Presentation
- 55% Final Project
  - 10% Project Proposal
  - 15% Mid-term Report
  - 10% Final Course Presentation (Group-based)
  - 5% Feedbacks for other groups’ final project presentations
  - 20% Final Project Report

In Class Presentation

- Starting from Week 3, each lecture will consist of one research topic of large language models, with 4 state-of-the-art papers. Each lecture will be covered by two students.
- Each student is required to do a 35-min presentation in class to cover two papers, followed by a 5-min Q&A/discussion session.
- Sign-up sheet for paper presentation will be released later this week.
- Remember to send over your slides to the instructor (and cc the TAs) before your presentation:
  - For Tuesday classes, send over your slides before the previous Friday 12:00PM
  - For Thursday classes, send over your slides before the previous Monday 12:00PM
- When it is not your turn to present, you can preview the paper in advance. Each student is required to submit a preview question for a paper one day before the presentation for **3** times (need to be on 3 different classes). You are also encouraged to raise that question in class.
  - Preview questions cannot be simple ones like “what is the aim of the paper?”

In Class Presentation

- How to present a paper:
  - Think about the context of the research: introduce the background of the research topic
  - What is the challenge and contribution of this paper, given the research background?
  - The method: from framework to technical details
  - What are some interesting experiment results and observations?
  - What could be done in the future?
  - Summarize the takeaways/highlights of this paper
  - Please control your time(35min)! We will give you notice when your time is nearly used up.

Final Project

- Students need to form groups of 2-3 people to do a large language model research project.
- Project proposal deadline: 2/19 11:59PM
- Midterm project report deadline: 3/18 11:59PM
- Final project presentation deadline: 4/17 11:59PM
  - We will use three lectures for project presentation: 4/18, 4/23, 4/25
- Final project report deadline: 4/26 11:59PM

Final Project

- There are typically two types of projects.
- 1) Designing a novel algorithm to train a medium-sized language model: BERT, GPT-2 for problems that you are interested in.
  - https://huggingface.co/models
- 2) Designing a novel algorithm to do inference on large language models (white box models such as LLaMA2 models, or black box models such as GPT-4, CLAUDE, etc.) to solve some type of complex problems, and analyze their limitations. (We may not be able to reimburse for the API costs, so you can choose to use free APIs such as CLAUDE)
  - https://platform.openai.com/docs/introduction
  - https://docs.anthropic.com/claude/reference/getting-started-with-the-api

Final Project Presentation

- Near the end of the semester, we will create a signup sheet for the final project presentation.
- We anticipate to distribute project presentations into three courses (4/18, 4/23, 4/25), and you will need to signup for a time slot.
- Length of project presentation: 15-20min depending on the number of groups
- Students will need to submit feedback scores for other groups' presentation (through Google Form).

Content

- Course Logistics
- **Language Model Basics**
- Covered Topics Preview

Language models

- The classic definition of a language model (LM) is a probability distribution over each token sequence $[w_1, w_2, \ldots, w_n]$, whether it's a good or bad one.
- Sally fed my cat with meat: P(I, feed, my, cat, with, meat) = 0.03,
- My cat fed Sally with meat: P(My, cat, fed, Sally, with, meat) = 0.005,
- fed cat meat my my with: P(fed, cat, meat, my, my, with) = 0.0001

Autoregressive language models

- The chain rule of probability:
- P(Sally, fed, my, cat, with, meat) = P(Sally)

  * P(fed | Sally)

  * P(my | Sally, fed)

  * P(cat | Sally, fed, my)

  * P(with | Sally, fed, my, cat)

  * P(meat | Sally, fed, my, cat, with)

Conditional probability

$$p(w_1, w_2, w_3, \ldots, w_N) =$$
$$p(w_1)\, p(w_2|w_1)\, p(w_3|w_1, w_2) \times \ldots \times p(w_N|w_1, w_2, \ldots w_{N-1})$$

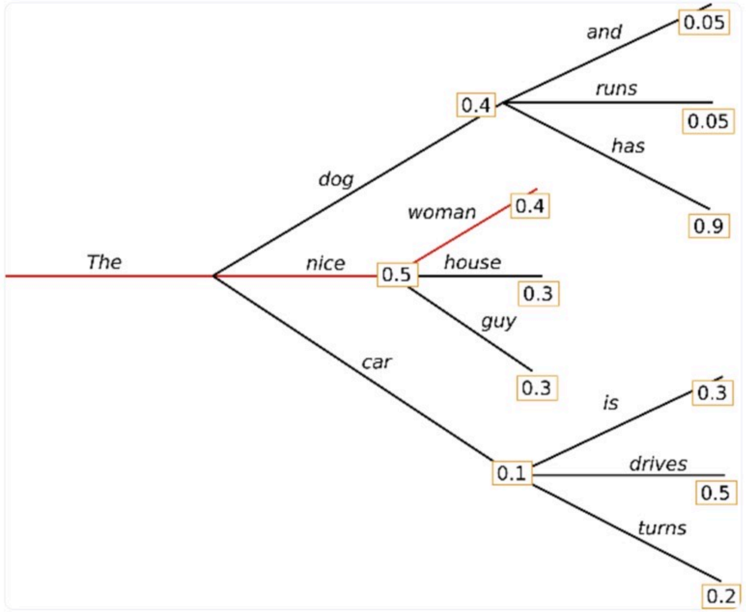
Generation

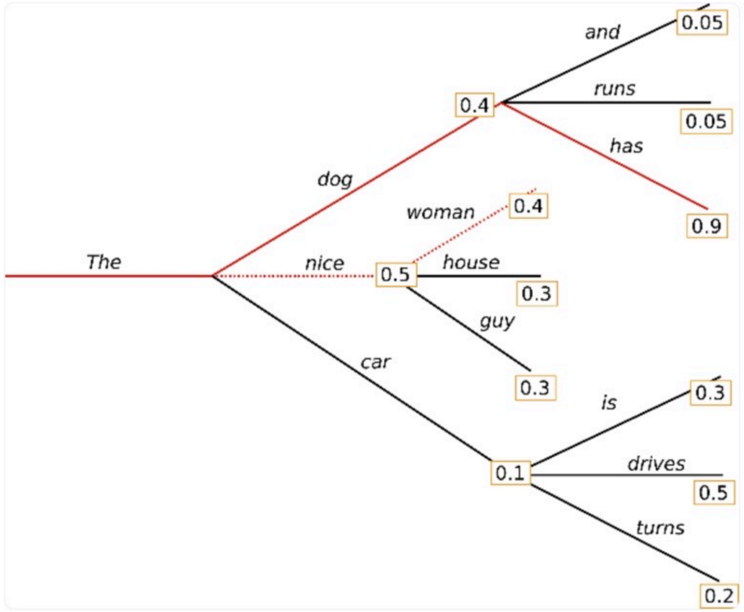
- If we already have a good language model, a given text prompt $w_{[1:n]}$, and we want the model to generate a good sentence completion with the length of L: How to find $w_{[n+1:n+L]}$ with the highest probability?
- Enumerate over all possible combinations?

- Next token prediction: generating the next token step by step, starting from w_{n+1} using $p(w_{n+1}|w_{[1:n]})$
- To select the next token with $p(w_{n+1}|w_{[1:n]})$, there are also different decoding approaches.

Different Decoding Approaches

- Greedy decoding: At each step, always select w_t with the highest $p(w_t|w_{[1:t-1]})$.
- Beam Search: Keep track of k possible paths at each step instead of just one. Reasonable beam size k: 5-10 .



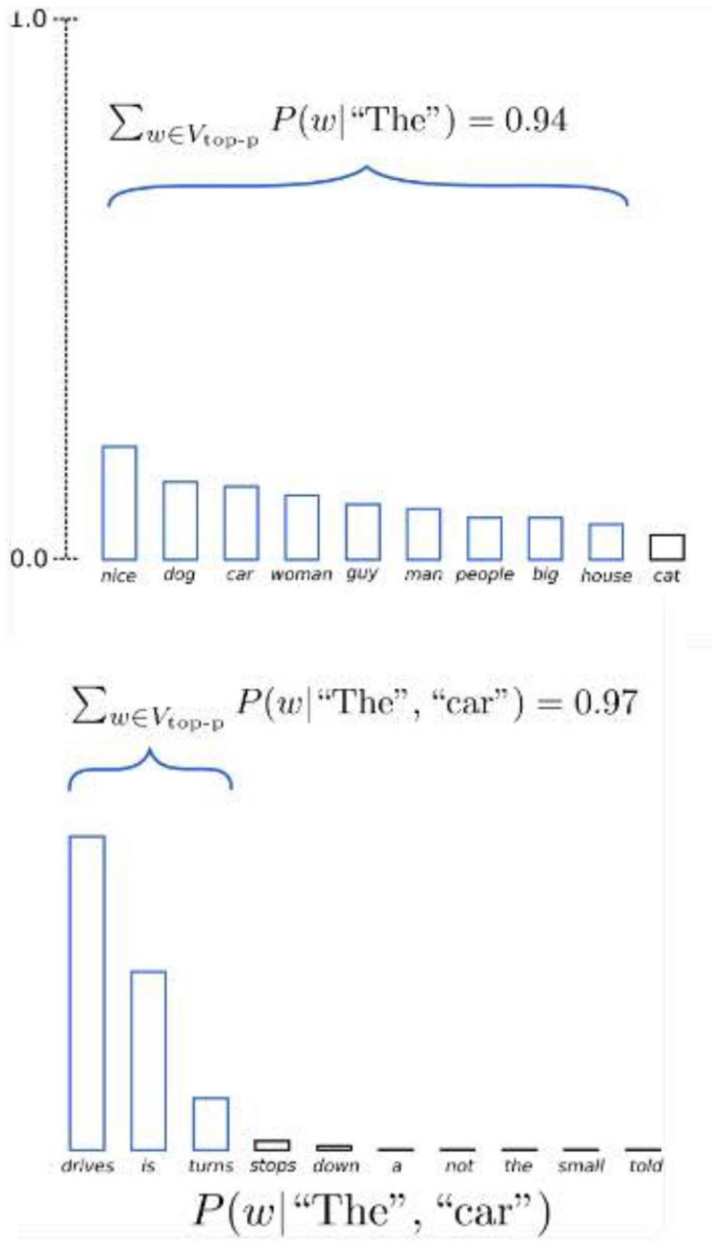


Different Decoding Approaches

- Top-k sampling: At each step, randomly sample the next token from $p\left(w_t \middle| w_{[1:t-1]}\right)$, but restrict to only the k most probable tokens.
- Allows you to control diversity:
  - Increase k gives you more creative / risky outputs.
  - Decrease k gives you safer outputs.

- Top-p sampling: At each step, randomly sample the next token from $p\left(w_t \middle| w_{[1:t-1]}\right)$, but restrict to the set of tokens with a cumulative probability of p
  - throw away long-tailed tokens

- Top-k and Top-p can be used together!



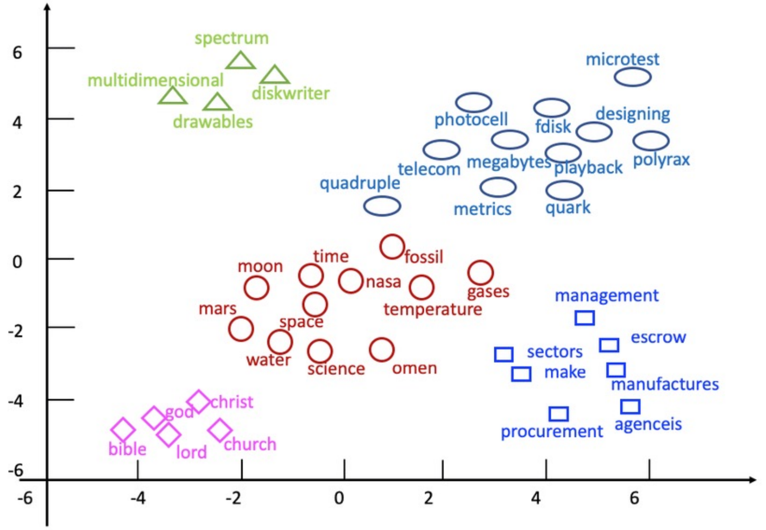
Q: How to train a good language model?

Q: How to train a good language model?

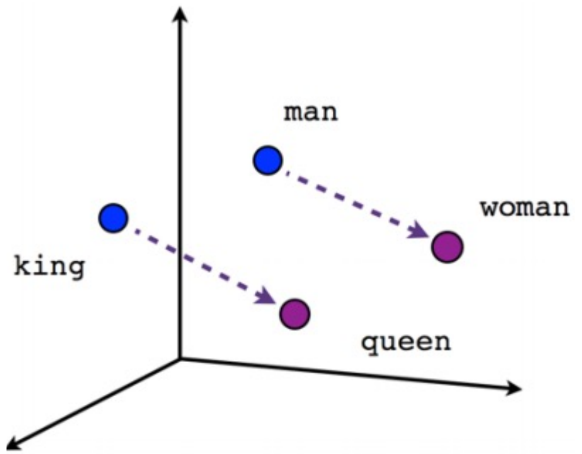
A: Maximizing the language model probability of an observed large corpus.

How to represent text?

- A milestone in NLP and ML:
  - Unsupervised learning of text representations—No supervision needed
  - Embed one-hot vectors into lower-dimensional space—Address "curse of dimensionality"
  - Word embedding captures useful properties of word semantics
  - Word similarity: Words with similar meanings are embedded closer
  - Word analogy: Linear relationships between words (e.g. king - queen = man - woman)



Word Similarity



Word Analogy

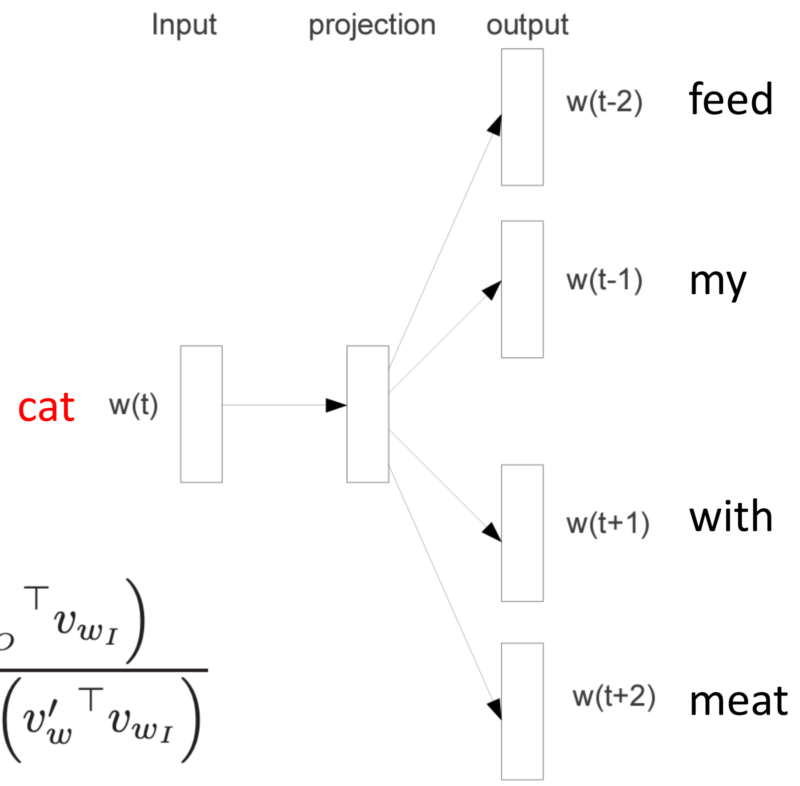
Distributed Representations: Word2Vec

- Assumption: If two words have similar contexts, then they have similar semantic meanings!
- Word2Vec Training objective:
- To learn word vector representations that are good at predicting the nearby words.

Co-occurred words in a **local context window**

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\le j\le c, j\ne 0}\log p(\boxed{w_{t+j}|w_t})$$

$$p(w_O|w_I) = \frac{\exp\left({v'_{w_O}}^\top v_{w_I}\right)}{\sum_{w=1}^{W} \exp\left({v'_w}^\top v_{w_I}\right)}$$



Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., & Dean, J. (2013). Distributed Representations of Words and Phrases and their Compositionality. NIPS.

Considering subwords: fastText

- fastText improves upon Word2Vec by incorporating subword information into word embedding

Tri-gram extraction

`<where>` → `<wh, whe, her, ere, re>`

- fastText allows sharing subword representations across words, since words are represented by the aggregation of their n-grams

Word2Vec probability expression

$$p(w_O|w_I) = \frac{\exp\left(\boxed{{v'_{w_O}}^\top v_{w_I}}\right)}{\sum_{w=1}^{W} \exp\left({v'_w}^\top v_{w_I}\right)} \dashrightarrow \sum_{g \in \mathcal{G}_w} \boxed{\mathbf{z}_g^\top} \mathbf{v}_c.$$

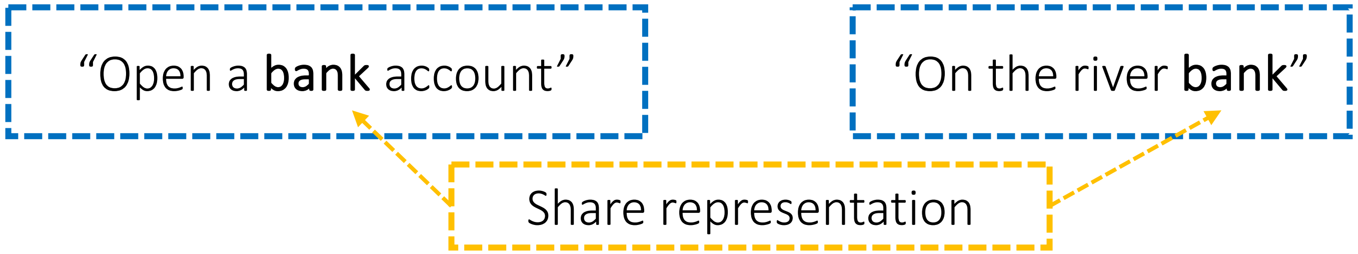
Represent a word by the sum of the vector representations of its n-grams

N-gram embedding

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2016). Enriching Word Vectors with Subword Information. Transactions of the Association for Computational Linguistics, 5, 135-146.

Limitations of Word2Vec embeddings

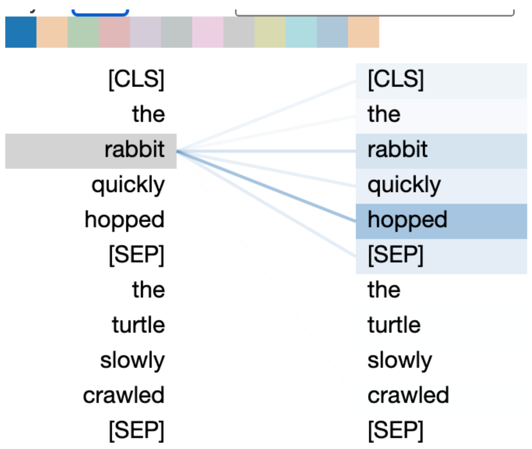
- 1) They are **context-free** embeddings: each word is mapped to only one vector regardless of its context!
  - E.g. “bank” is a polysemy, but only has one representation

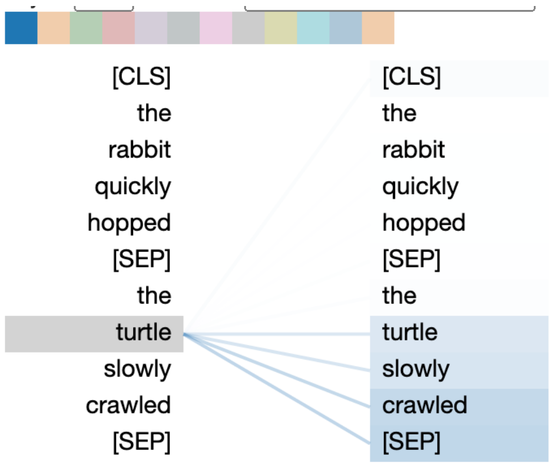


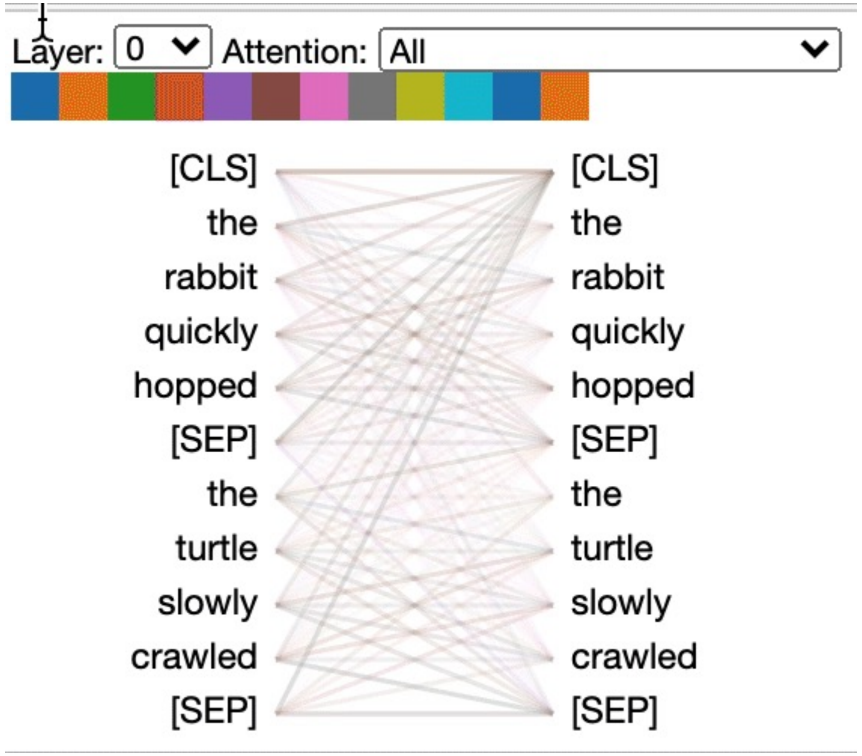
- 2) It does not consider the order of words
- 3) It treats the words in the context window equally

Attention is all you need (Transformer)

- Self-Attention: Each token attends to every other token in the sentence, but with different weights
- Demo: https://github.com/jessevig/bertviz

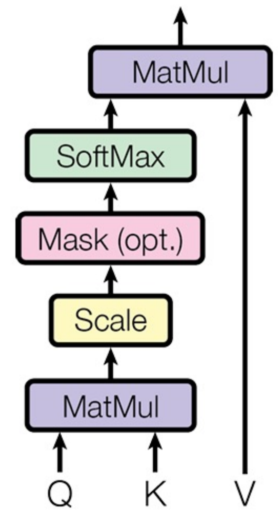




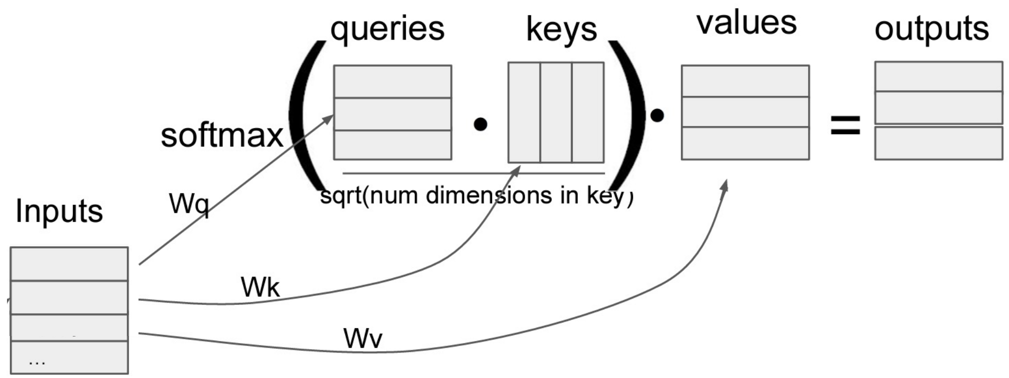


Self-Attention

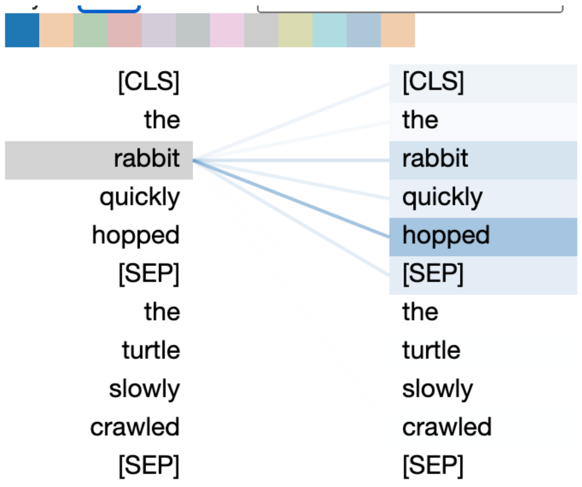
Scaled Dot-Product Attention



- To calculate the attention weight from a query word w_q (e.g, "rabbit") to another word w_k
- Each word is represented as a query, key and value vector. The vectors are obtained from the input embeddings multiplied by a weight matrix.



$$\mathrm{Attention}(Q,K,V) = \mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

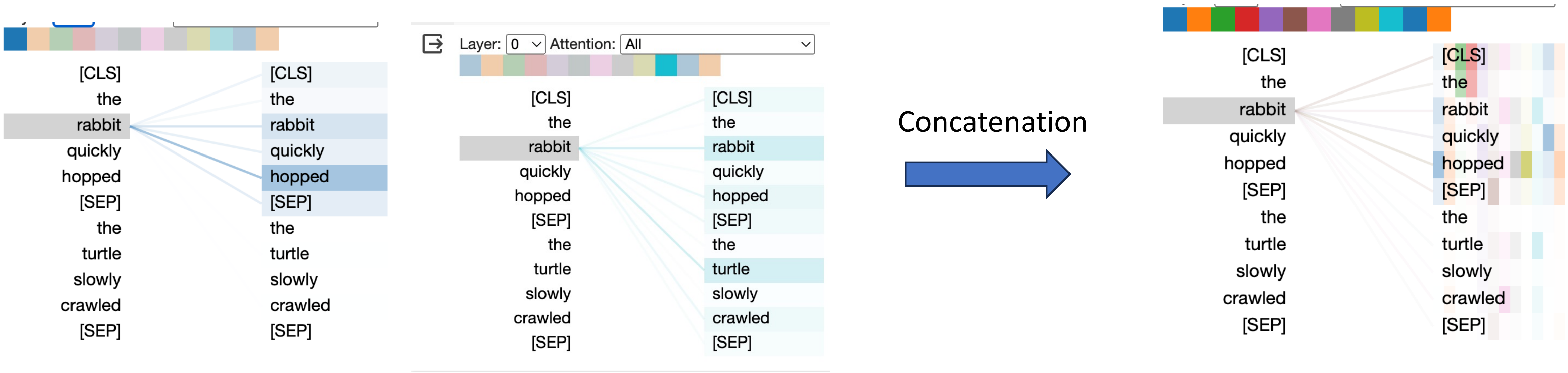


Multi-Head Attention

- Input: Multiple Independent sets of query, key, value matrix
- Output: Concatenate the outputs of attention heads
- Advantage: Each attention head focus on one subspace

$$\mathrm{MultiHead}(Q, K, V) = \mathrm{Concat}(\mathrm{head_1}, \ldots, \mathrm{head_h})W^O$$

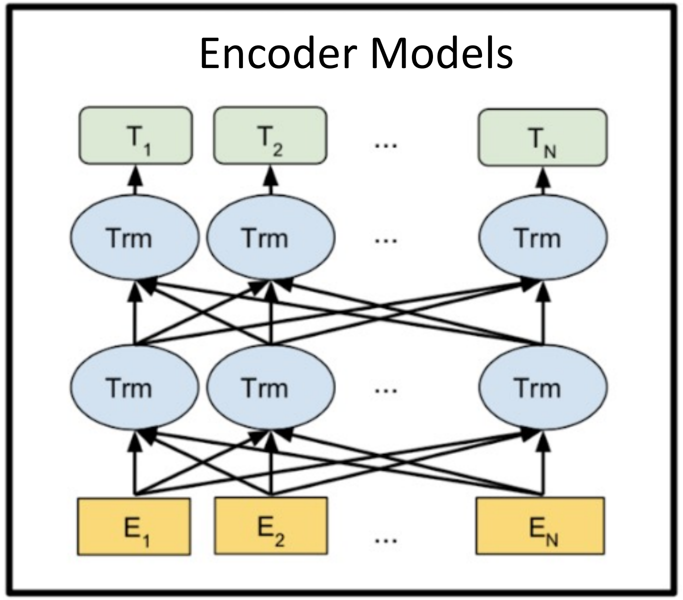
$$\text{where } \mathrm{head_i} = \mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

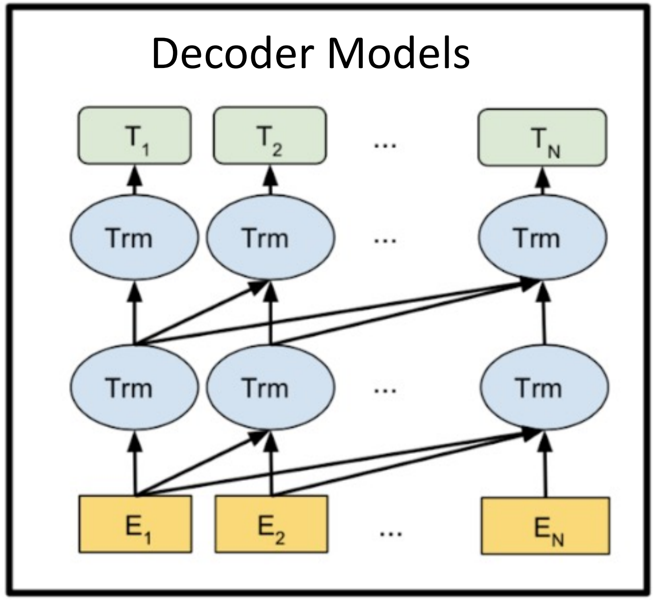


Content

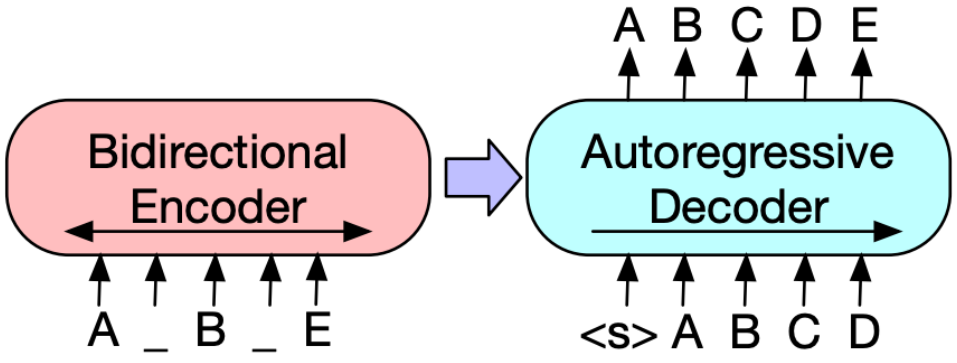
- Course Logistics
- Language Model Basics
- **Covered Topics Preview**

Language Model Architectures (will be covered in the next course)

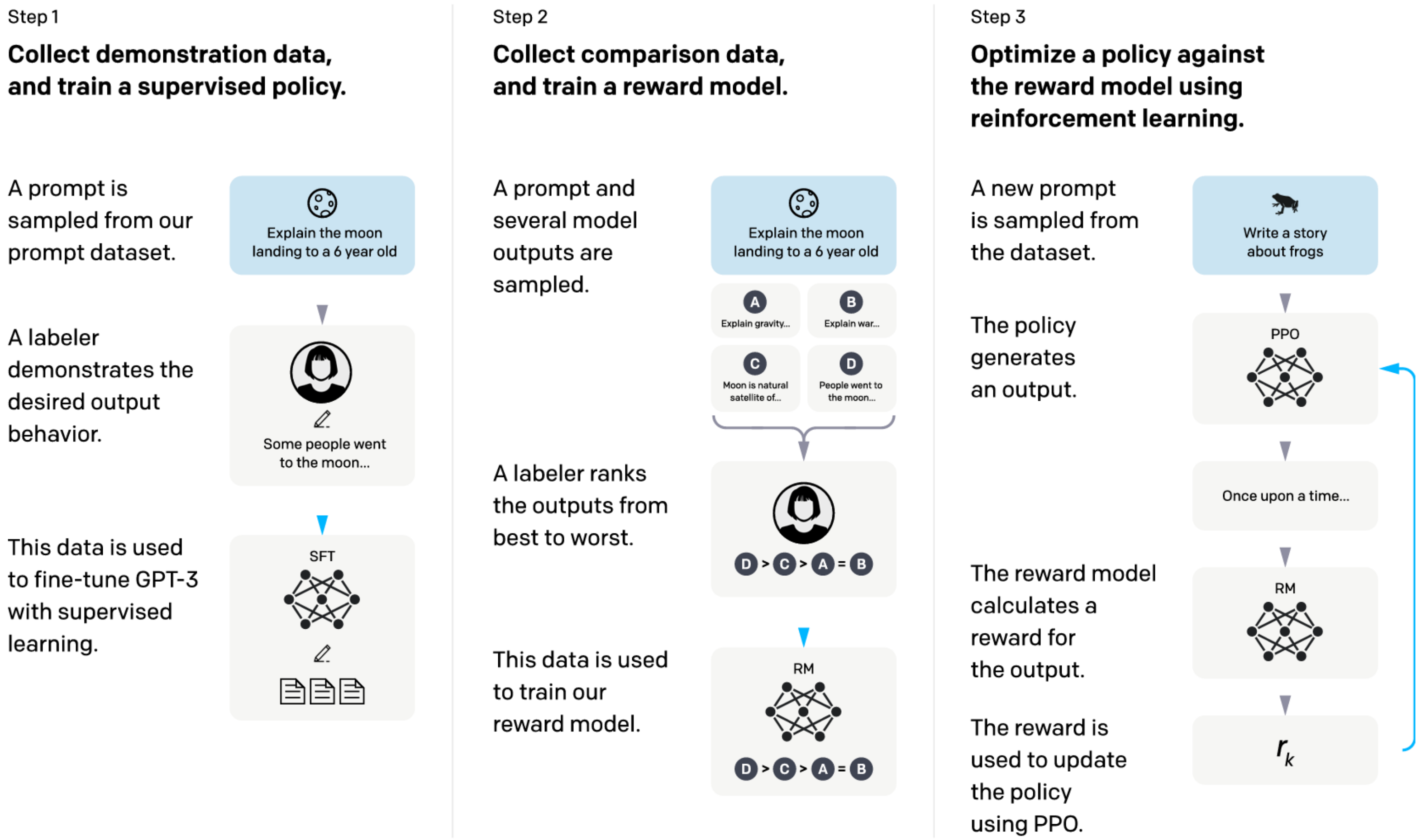




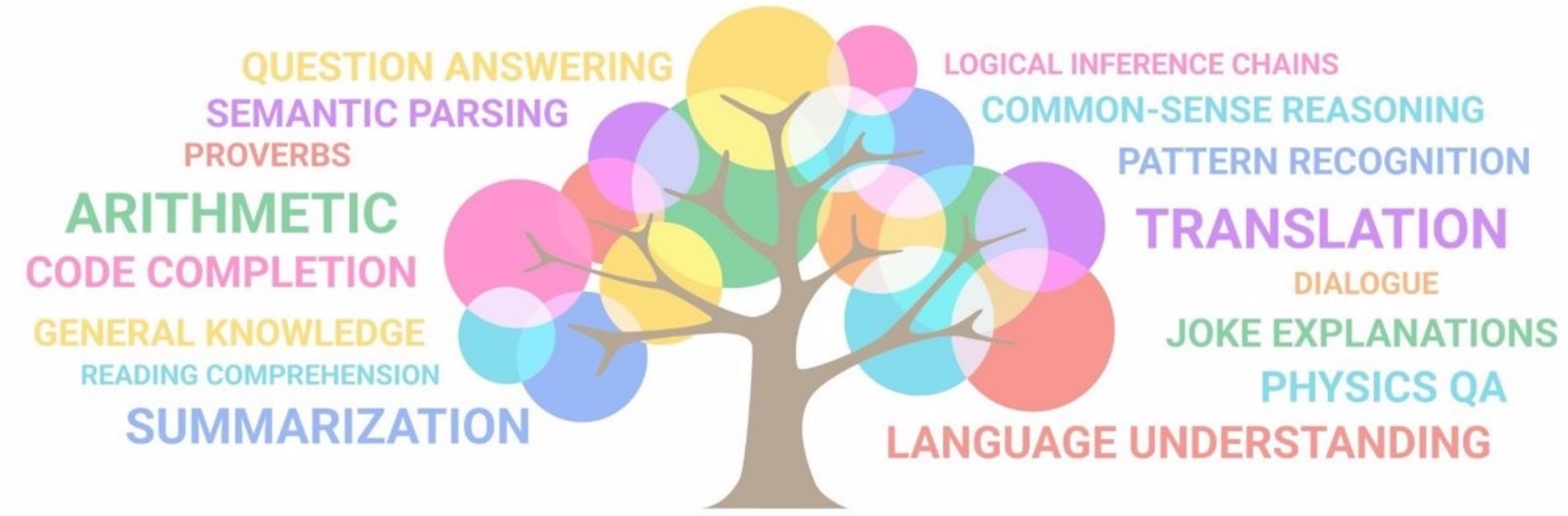
Encoder-Decoder Models

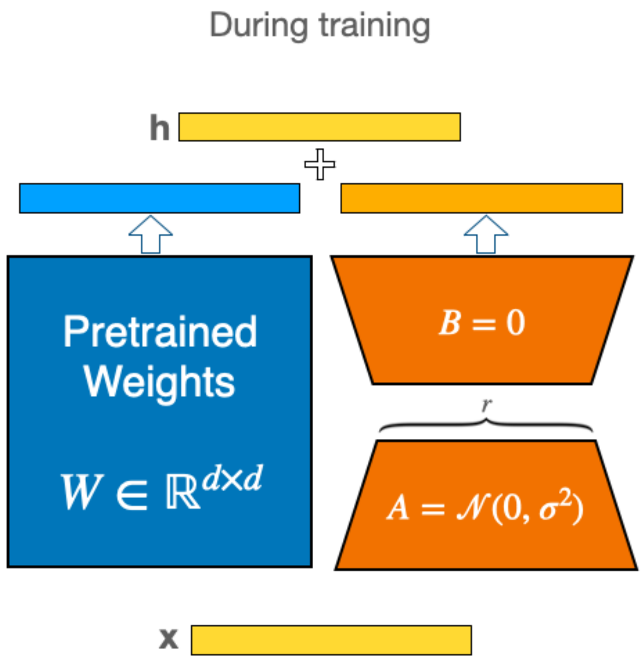


Large Language Model Pre-training Framework



Efficient Fine-Tuning





Unsupervised/Self-supervised;
On large-scale general domain corpus

→

Task-specific supervision;
On target corpus

Topics of Language Model Capabilities

Prompting and In-context Learning	Language Models are Few-Shot Learners (GPT-3) Emergent Abilities of Large Language Models Rethinking the Role of Demonstrations: What Makes In-Context Learning Work? Why Can GPT Learn In-Context? Language Models Implicitly Perform Gradient Descent as Meta-Optimizers
Language Model Reasoning (I)	Chain of Thought Prompting Elicits Reasoning in Large Language Models Self-Consistency Improves Chain of Thought Reasoning in Language Models Least-to-Most Prompting Enables Complex Reasoning in Large Language Models Tree of Thoughts: Deliberate Problem Solving with Large Language Models
Language Model Reasoning (II)	STaR: Bootstrapping Reasoning With Reasoning Large Language Models Can Self-Improve Progressive-Hint Prompting Improves Reasoning in Large Language Models Towards Understanding Chain-of-Thought Prompting: An Empirical Study of What Matters
Language Model Calibration/Uncertainty	How Can We Know When Language Models Know? On the Calibration of Language Models for Question Answering Teaching models to express their uncertainty in words Navigating the Grey Area: Expressions of Overconfidence and Uncertainty in Language Models Semantic Uncertainty: Linguistic Invariances for Uncertainty Estimation in Natural Language Generation

Solving Quantitative Reasoning Problems with Language Models

- Google's Minerva Model

Minerva

Question: Assume that the variance of the first n natural numbers is 10, and the variance of the first m even natural numbers is 16. Compute $m+n$.

Model output: Let σ^2 be the variance of the first m even natural numbers, and τ^2 be the variance of the first n natural numbers. Then, we have $\tau^2 = \frac{n^2-1}{12}$ and $\sigma^2 = \frac{m^2-1}{3}$. We solve the system of equations:

$$\frac{n^2-1}{12} = 10$$
$$\frac{m^2-1}{3} = 16.$$

This translates into the system

$$n^2 = 121$$
$$m^2 = 48+1 = 49.$$

Therefore, $n = \boxed{11}$ and $m = \boxed{7}$, so $n+m = \boxed{18}$.

Question: For every $a, b, b \neq a$ prove that

$$\frac{a^2+b^2}{2} > \left(\frac{a+b}{2}\right)^2.$$

Model output:

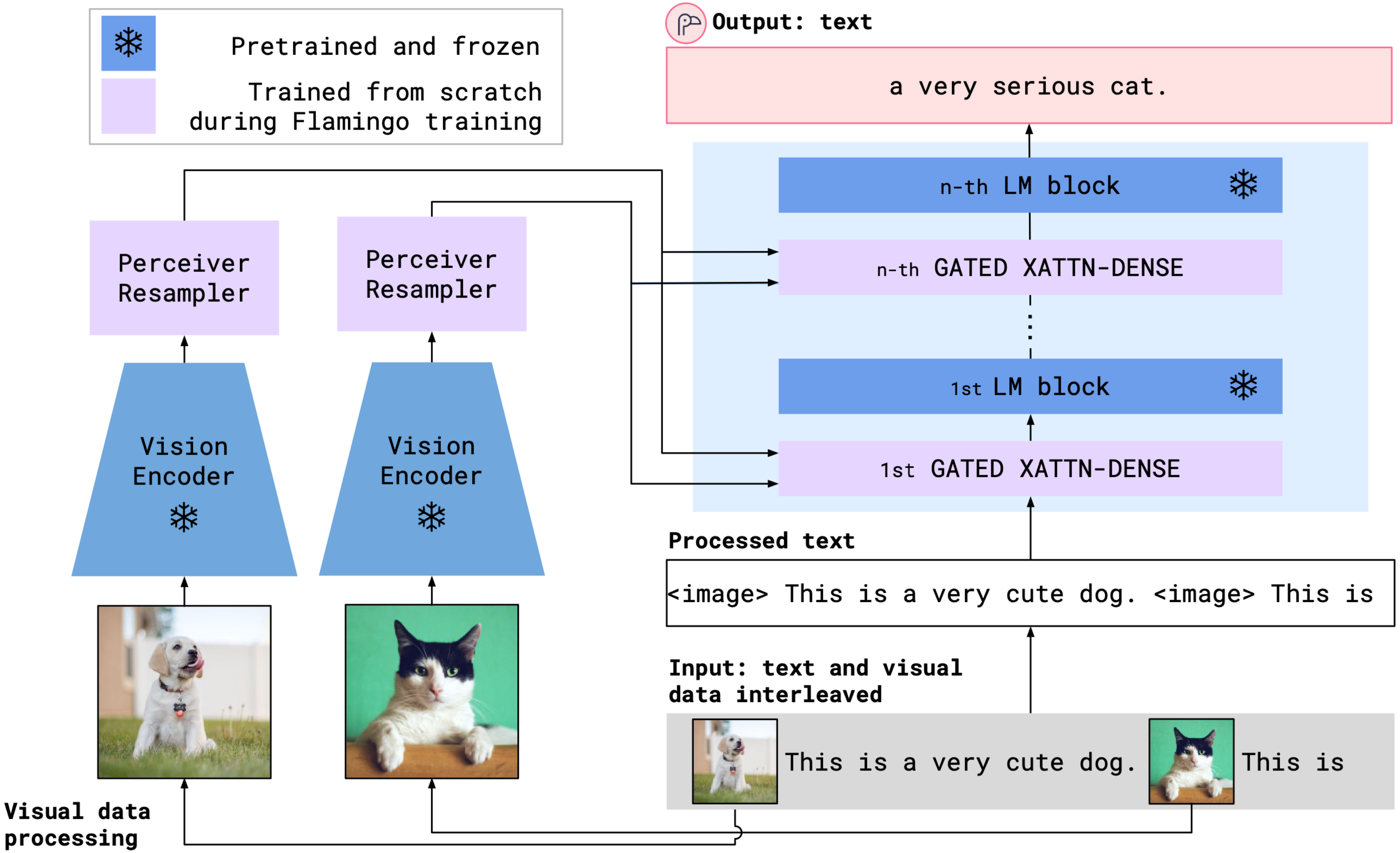
$$\frac{a^2+b^2}{2} > \left(\frac{a+b}{2}\right)^2$$
$$\iff \frac{a^2+b^2}{2} > \frac{a^2+b^2+2ab}{4}$$
$$\iff a^2+b^2 > \frac{a^2+b^2+2ab}{2}$$
$$\iff 2a^2+2b^2 > a^2+b^2+2ab$$
$$\iff a^2+b^2 > 2ab$$
$$\iff a^2+b^2-2ab > 0$$
$$\iff (a-b)^2 > 0$$

which is true, because the square of a real number is positive.

Language Models as Agents

Code Language Models	InCoder: A Generative Model for Code Infilling and Synthesis Code Llama: Open Foundation Models for Code Teaching Large Language Models to Self-Debug LEVER: Learning to Verify Language-to-Code Generation with Execution
Multimodal Language Models	Flamingo: a Visual Language Model for Few-Shot Learning VisionLLM: Large Language Model is also an Open-Ended Decoder for Vision-Centric Tasks Visual Instruction TuningNExT-GPT: Any-to-Any Multimodal LLM
Language Models as Agents	ReAct: Synergizing Reasoning and Acting in Language Models Toolformer: Language Models Can Teach Themselves to Use Tools AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation ART: Automatic multi-step reasoning and tool-use for large language models Reflexion: Language Agents with Verbal Reinforcement Learning
Evaluation of Language Models	Proving Test Set Contamination in Black Box Language Models Holistic Evaluation of Language Models Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena FActScore: Fine-grained Atomic Evaluation of Factual Precision in Long Form Text Generation

Integration of pre-trained vision model and language model



Issues and Future Directions of Language Models

Language Model Bias	Men Also Like Shopping: Reducing Gender Bias Amplification using Corpus-level Constraints Self-Diagnosis and Self-Debiasing: A Proposal for Reducing Corpus-Based Bias in NLP Whose Opinions Do Language Models Reflect? Red Teaming Language Models with Language Models	
Privacy	Extracting Training Data from Large Language Models Large Language Models Can Be Strong Differentially Private Learners Quantifying Memorization Across Neural Language ModelsSILO Language Models: Isolating Legal Risk In a Nonparametric Datastore	
Security	Universal and Transferable Adversarial Attacks on Aligned Language Models DecodingTrust: A Comprehensive Assessment of Trustworthiness in GPT Models Poisoning Language Models During Instruction Tuning GPT-4 Is Too Smart To Be Safe: Stealthy Chat with LLMs via Cipher	
Explorations of Large Language Models	Weak-To-Strong Generalization:Eliciting Strong Capabilities With Weak Supervision Hungry Hungry Hippos: Towards Language Modeling with State Space Models PaLM-E: An Embodied Multimodal Language Model When Large Language Models Meet Personalization: Perspectives of Challenges and Opportunities	

Bias of Language Models

- Different language models may have different political views.