

CSE 561A: Large Language Models

Fall 2025

Lecture 1: Course Overview

Jiaxin Huang

Content

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

Course Logistics

- Course meeting times: 2:30pm – 3:50pm Tuesday / Thursday
- Location: Jubel / 121
- TAs:
 - Zheyuan Wu (w.zheyuan@wustl.edu)
 - Isle Song (s.xiaodao@wustl.edu)

Course Logistics

- Course Syllabus: https://teapot123.github.io/CSE561A_2025fl/
- Canvas: <https://wustl.instructure.com/courses/155129> (will be published soon)
- We will be using Canvas for announcements, and project report submissions, and Piazza for discussions.

Course Structure

- Advanced Research-Oriented Course
 - We will be teaching and discussing **state-of-the-art research papers** about large language model architectures, training and inference.
 - Students will choose papers from a list of frontier research papers (https://teapot123.github.io/CSE561A_2025fl/) to present in the class
 - Guest lectures on frontier research topics
- Warning on pre-requisites: Please be aware that this is a **fast-paced, research-driven course**, not an introductory course to LLMs. The curriculum is tailored **for advanced students (PhD candidates) doing state-of-the-art LLM-related research**.
- Students without a strong machine learning research background and understanding of fundamental NLP concepts (sequence modeling, contextualized representation, etc.) will find the pace and technical depth of the material exceptionally challenging.

Grading

- 20% Class Participation
 - Submit 1 preview question before each paper presentation class (every class counts for 1 point)
 - Guest Lecture
- 25% Paper Presentation
- 55% Final Project (Group with 2-3 students)
 - 10% Project Proposal
 - 10% Mid-term Report
 - 10% Final Course Presentation (Zoom)
 - 5% Feedbacks for other groups' final project presentations
 - 20% Final Project Report

Paper 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 **three** students, who are required to prepare a 60-min presentation in class to cover 4 papers.
 - Distribute the presentation among yourselves
 - Suggestion: 15 min presentation + 1 min Q&A for each paper
- **Sign-up sheet for picking out your presented paper** will be released later this week.
- **Remember to send over your slides to the instructor (and cc the TA) before your presentation:**
 - For Tuesday classes, send over your slides before the previous Friday 12:00PM to both the instructor (jiaxinh@wustl.edu) and Zheyuan (w.zheyuan@wustl.edu)
 - For Thursday classes, send over your slides before the previous Monday 12:00PM to both the instructor (jiaxinh@wustl.edu) and Isle (s.xiaodao@wustl.edu)
 - Late submissions -> not well-prepared, point penalty

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

In Class Presentation

- More tips to do presentations
 - Get familiar with your material. **Don't read scripts for the whole time.**
 - Make eye contact with audiences.
 - Make your voice loud enough so that everyone can hear you clearly
 - Please control your time (15min for each paper)! We will give you notice when your time is nearly used up.

Preview Question Submission

- When it is **not your turn** to present, you need to preview the paper in advance.
- Each student is required to submit 1 preview question for a paper one day before the presentation. You are also encouraged to raise that question in class.
 - Preview questions cannot be simple ones like “what is the aim of the paper?”, or “what is the difference between this and previous methods?”

Research Project

- Students need to form groups of 2-3 people to do a large language model research project.
- Project proposal deadline: 9/15 11:59PM
- Midterm project report deadline: 10/20 11:59PM
- Final project presentation deadline: 12/1 11:59PM
 - We will use two lectures for project presentation: 12/2, 12/4
- Final project slides deadline: 12/12 11:59PM

Research 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.
 - Find open-weight models here: <https://huggingface.co/models>
- 2) Designing a novel algorithm to do inference on large language models (Qwen, Llama, DeepSeek series, or GPT, Gemini, CLAUDE) to solve some type of complex problems, and analyze their limitations. (We may not be able to reimburse for the API costs)
 - <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 use 2 classes (12/2, 12/4) for every group to present their projects via Zoom.
- Length of project presentation: 5-8min depending on the number of groups
- Students will need to submit feedback scores for other groups' presentation (through Google Form).

Content

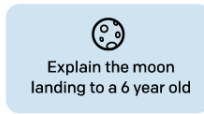
- Course Logistics
- **Covered Topics Preview**
- Language Model Basics

Large Language Model Pre-training and Post-training Framework

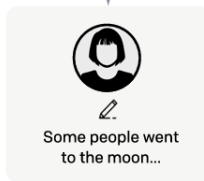
Step 1

Collect demonstration data, and train a supervised policy.

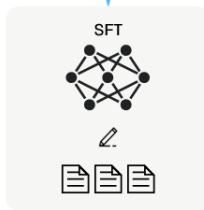
A prompt is sampled from our prompt dataset.



A labeler demonstrates the desired output behavior.



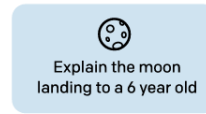
This data is used to fine-tune GPT-3 with supervised learning.



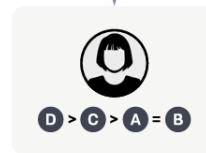
Step 2

Collect comparison data, and train a reward model.

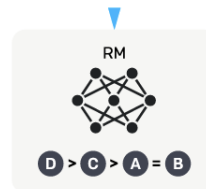
A prompt and several model outputs are sampled.



A labeler ranks the outputs from best to worst.



This data is used to train our reward model.



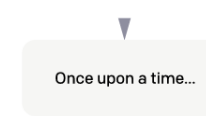
Step 3

Optimize a policy against the reward model using reinforcement learning.

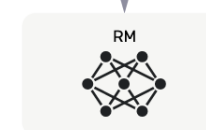
A new prompt is sampled from the dataset.



The policy generates an output.



The reward model calculates a reward for the output.

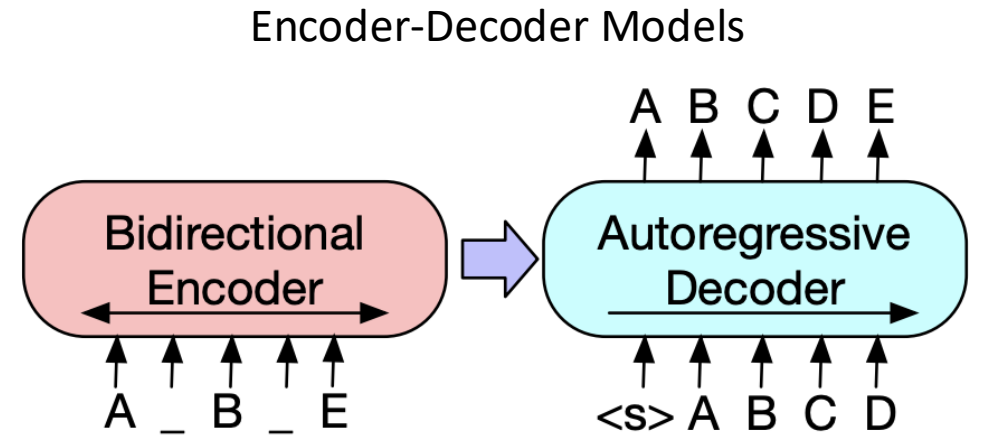
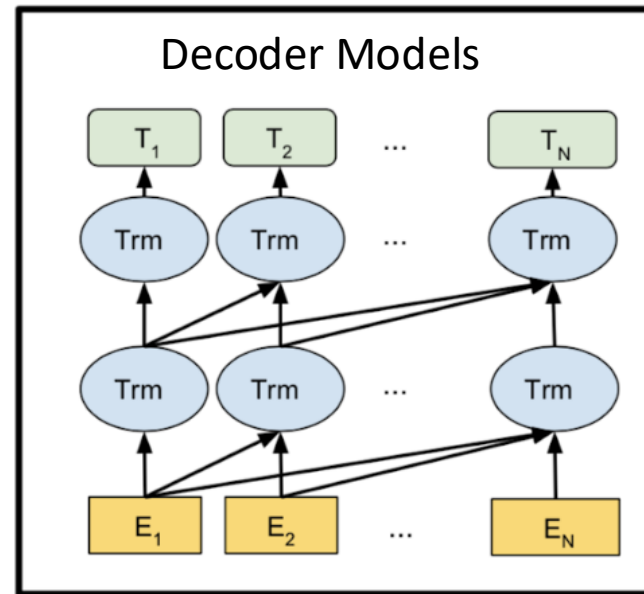
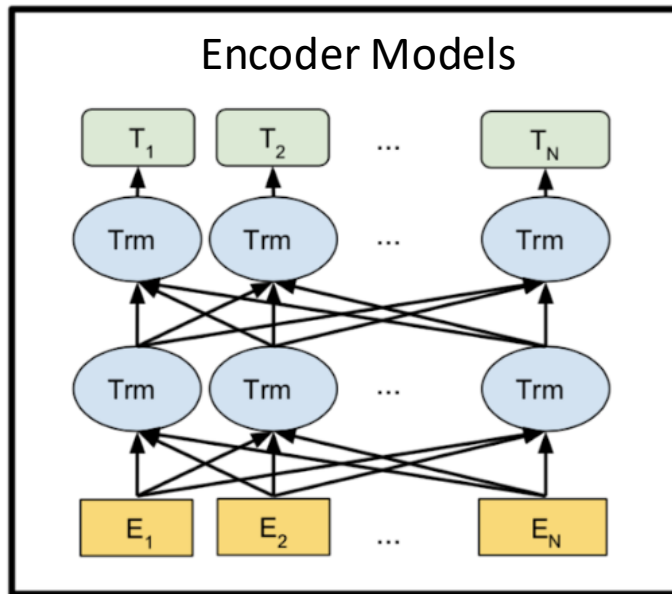


The reward is used to update the policy using PPO.



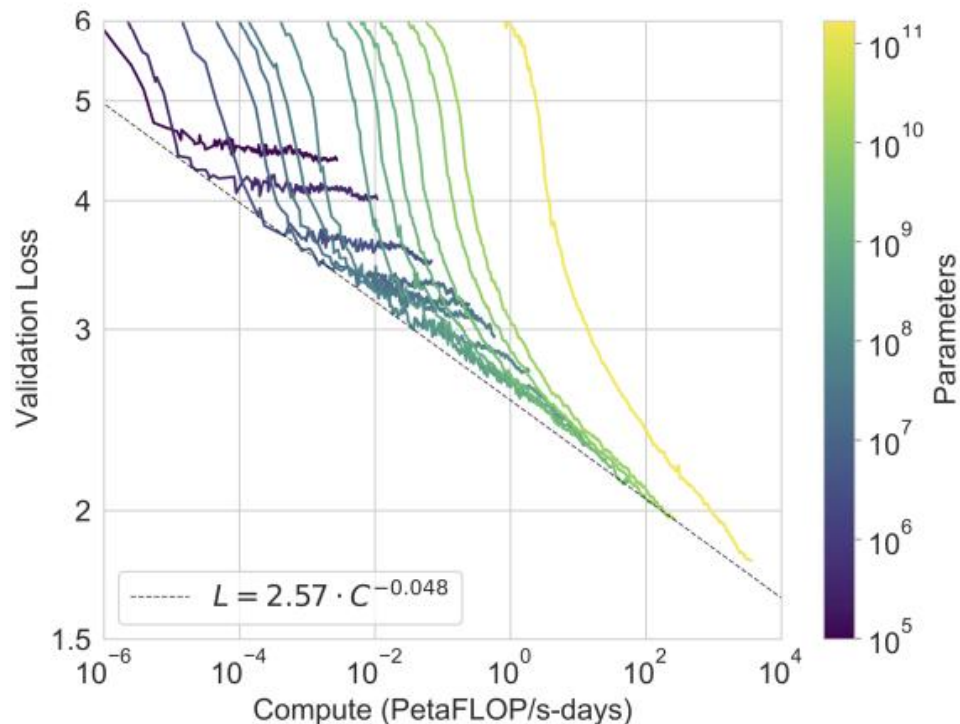
InstructGPT: Training language models to follow instructions with human feedback. (Ouyang et. al, 2022)

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



Scaling Up Language Models

- Performance on validation set (cross entropy loss on standard language modeling task) follows a power-law trend with respect to the amount of computation in training



Zero-shot

The model predicts the answer given only a natural language description of the task. No gradient updates are performed.

```
1 Translate English to French: ← task description
2 cheese => ..... ← prompt
```

One-shot

In addition to the task description, the model sees a single example of the task. No gradient updates are performed.

```
1 Translate English to French: ← task description
2 sea otter => loutre de mer ← example
3 cheese => ..... ← prompt
```

Few-shot

In addition to the task description, the model sees a few examples of the task. No gradient updates are performed.

```
1 Translate English to French: ← task description
2 sea otter => loutre de mer ← examples
3 peppermint => menthe poivrée
4 plush girafe => girafe peluche
5 cheese => ..... ← prompt
```

Language Model Post-Training: Instruction Tuning

Collecting from multiple public NLP datasets

Training mixtures:

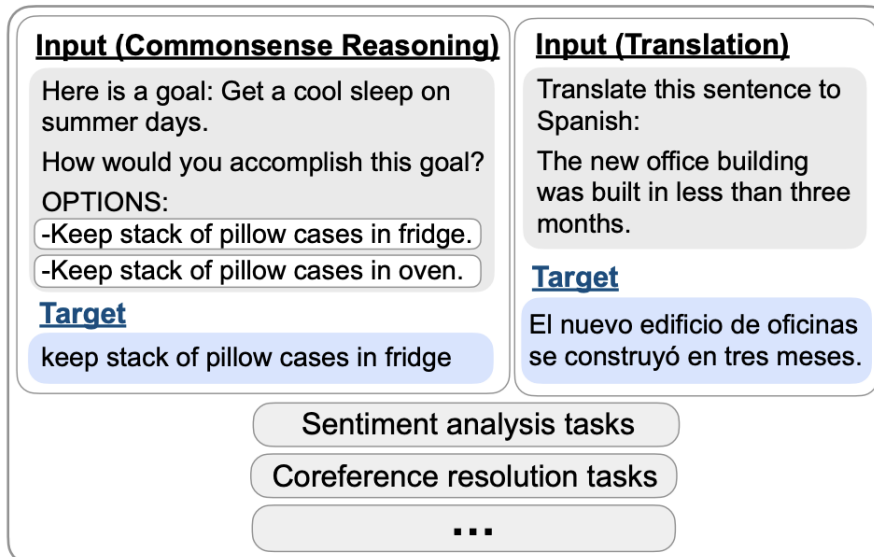
QA (Question Answering tasks),
structure-to-text,
summarization
Sentiment analysis, topic
classification, paraphrase
identification

Held-out test set:

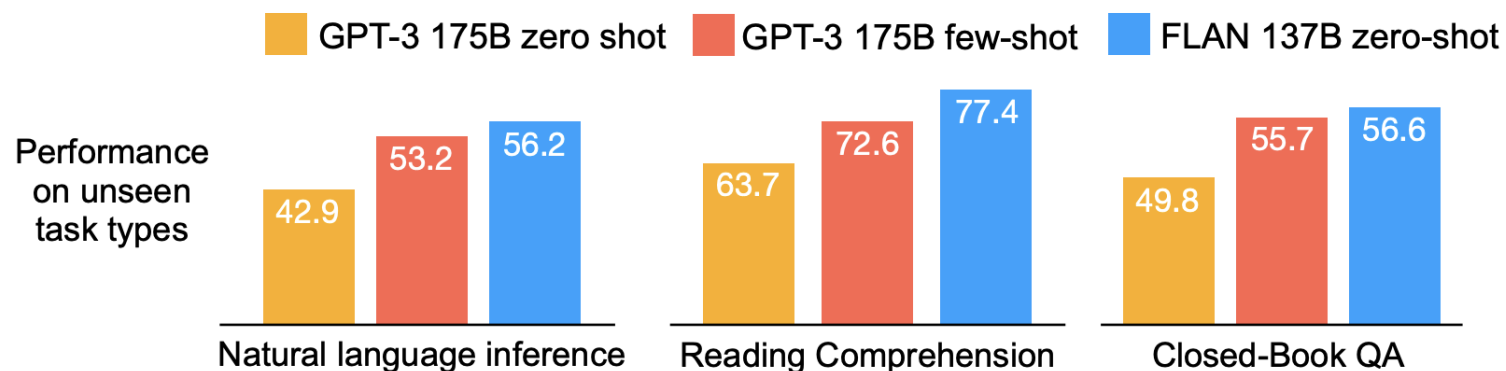
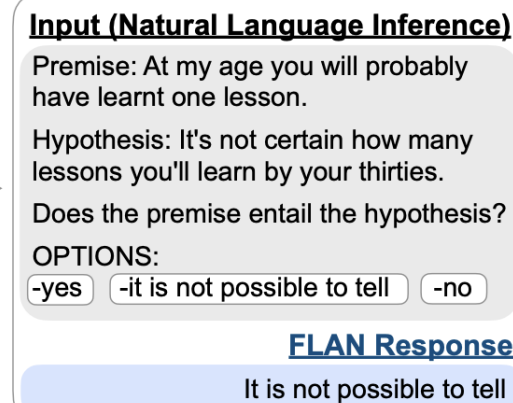
Sentence completion, BIG-Bench

Natural language inference,
coreference resolution, word
sense disambiguation

Finetune on many tasks (“instruction-tuning”)

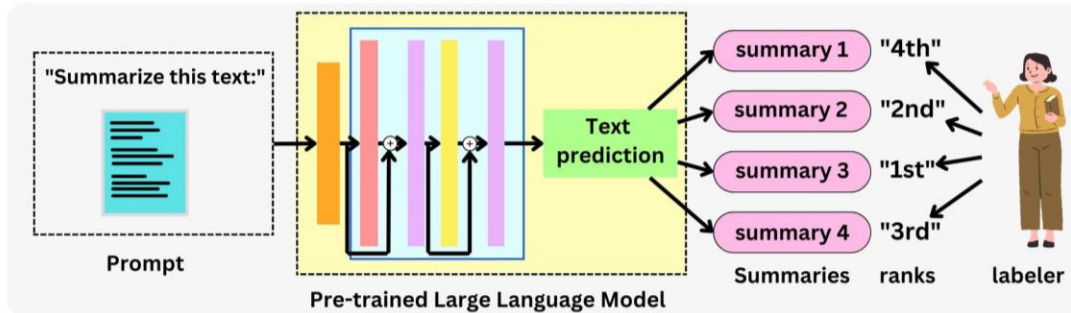


Inference on unseen task type

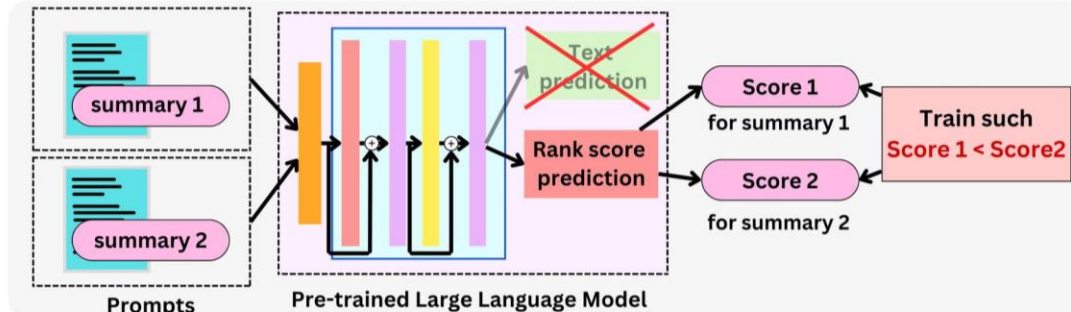


Language Model Post-Training: Reinforcement Learning from Human Feedback

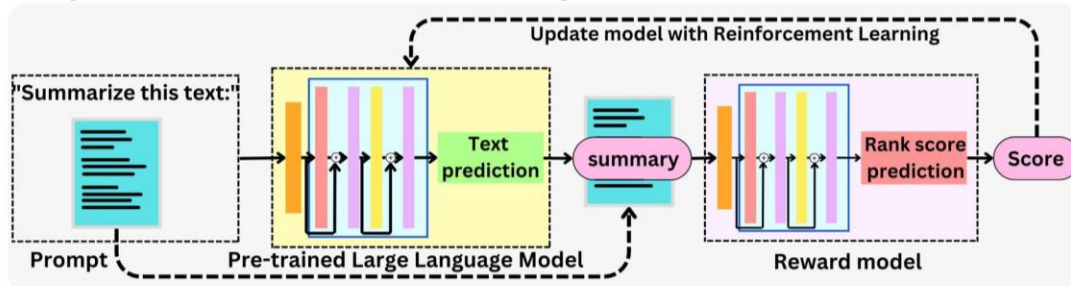
Step 1: Rank model outputs with human labeler TheAiEdge.io



Step 2: Train Reward model to learn to rank output



Step 3: Use Reward model to update model with RL



- Agent: language model
- Action: predict the next token
- Policy π_{θ} : the output distribution of the next token
- Reward: a reward model r_{ϕ} trained by human evaluations on model responses, so no more human-in-the-loop is needed

Topics: Language Model Reasoning

- Today's LLMs can solve very hard math problems, even those in AIME'25.

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:

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

This translates into the system

$$\begin{aligned}n^2 &= 121 \\ m^2 &= 48 + 1 = 49.\end{aligned}$$

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

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

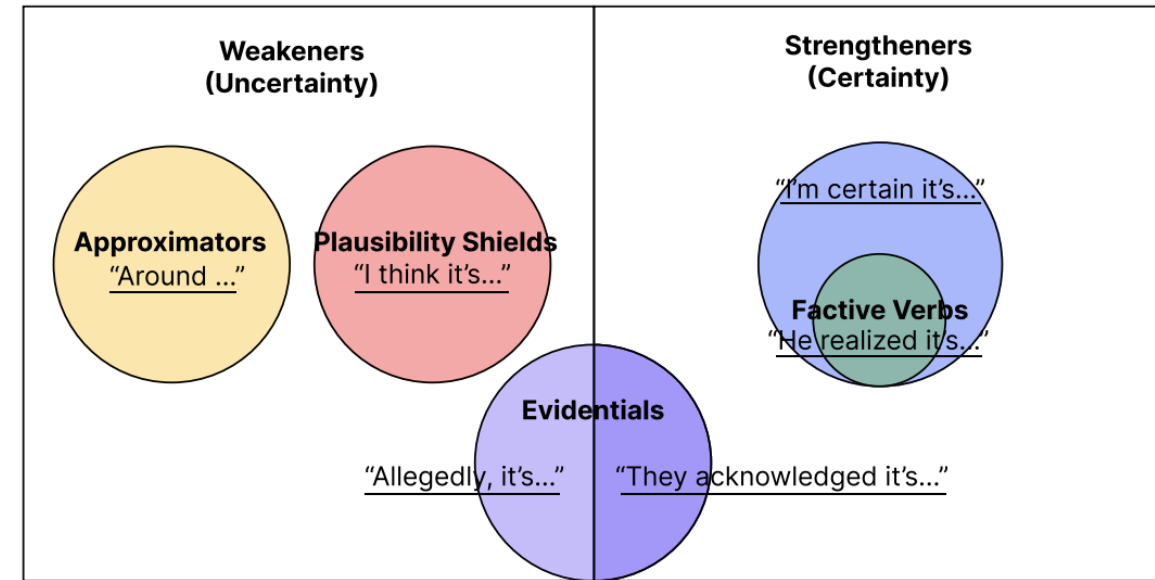
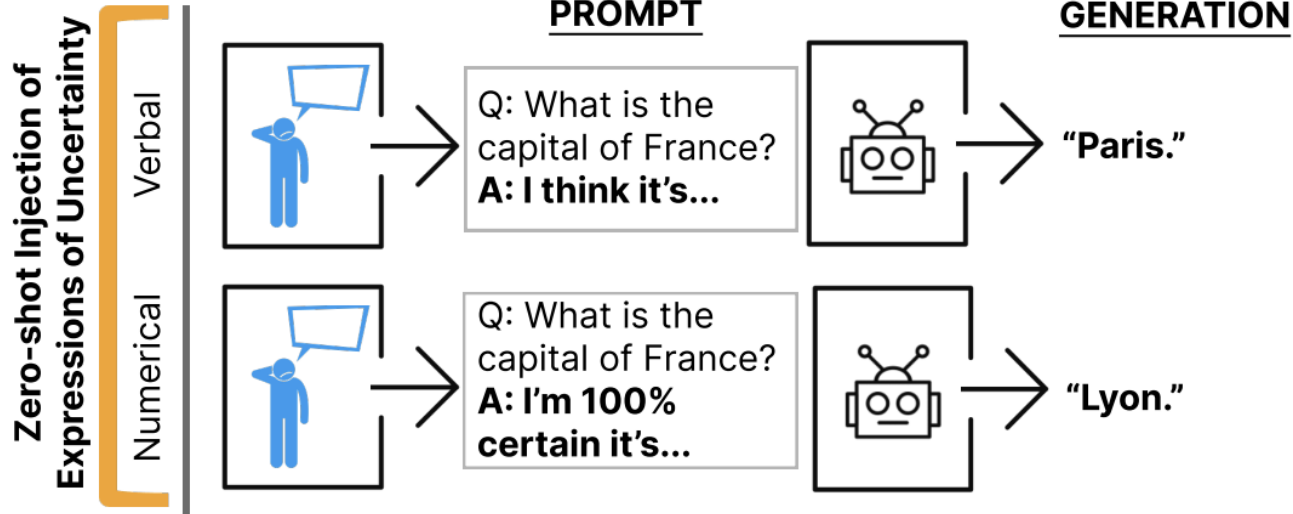
$$\frac{a^2 + b^2}{2} > \left(\frac{a+b}{2}\right)^2.$$

Model output:

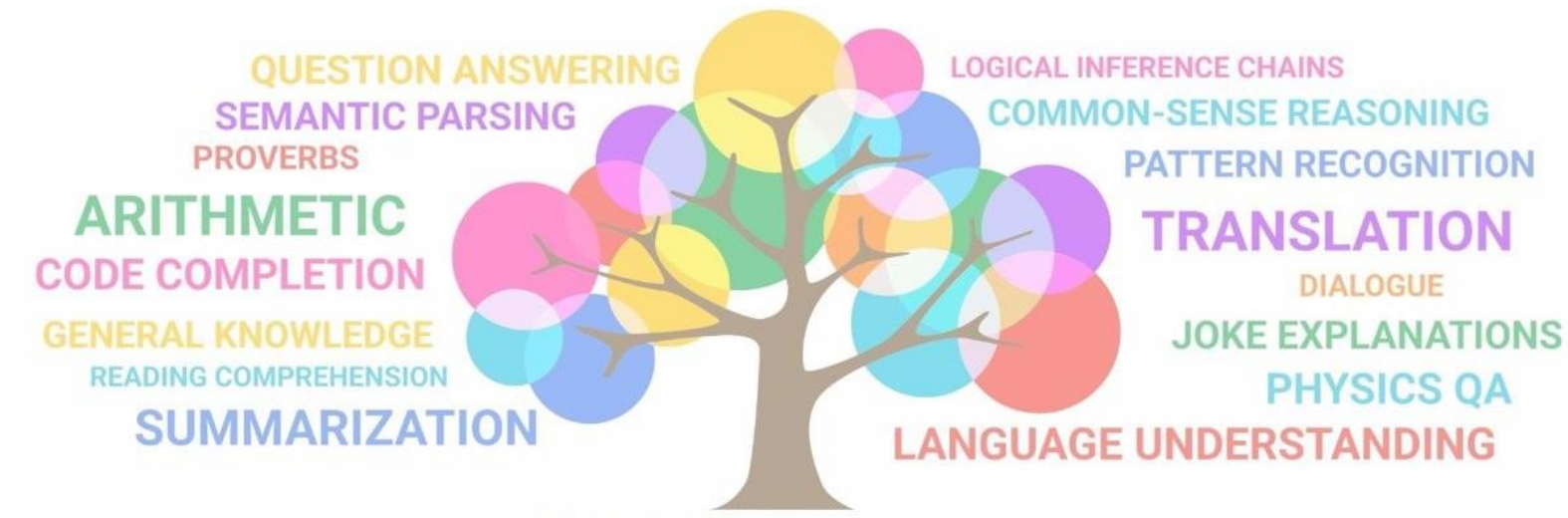
$$\begin{aligned}\frac{a^2+b^2}{2} &> \left(\frac{a+b}{2}\right)^2 \\ \iff \frac{a^2+b^2}{2} &> \frac{a^2+b^2+2ab}{4} \\ \iff a^2 + b^2 &> \frac{a^2+b^2+2ab}{2} \\ \iff 2a^2 + 2b^2 &> a^2 + b^2 + 2ab \\ \iff a^2 + b^2 &> 2ab \\ \iff a^2 + b^2 - 2ab &> 0 \\ \iff (a-b)^2 &> 0\end{aligned}$$

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

Topics: Language Model Calibration



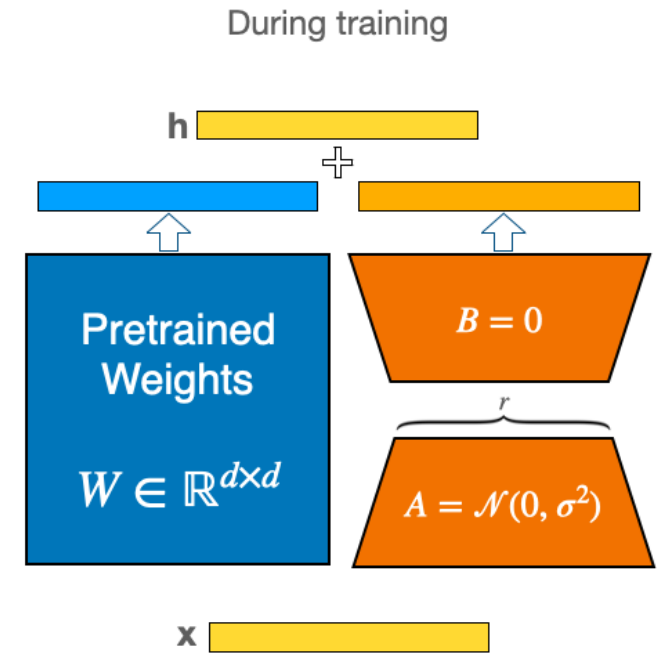
Topics: Efficient Fine-Tuning



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

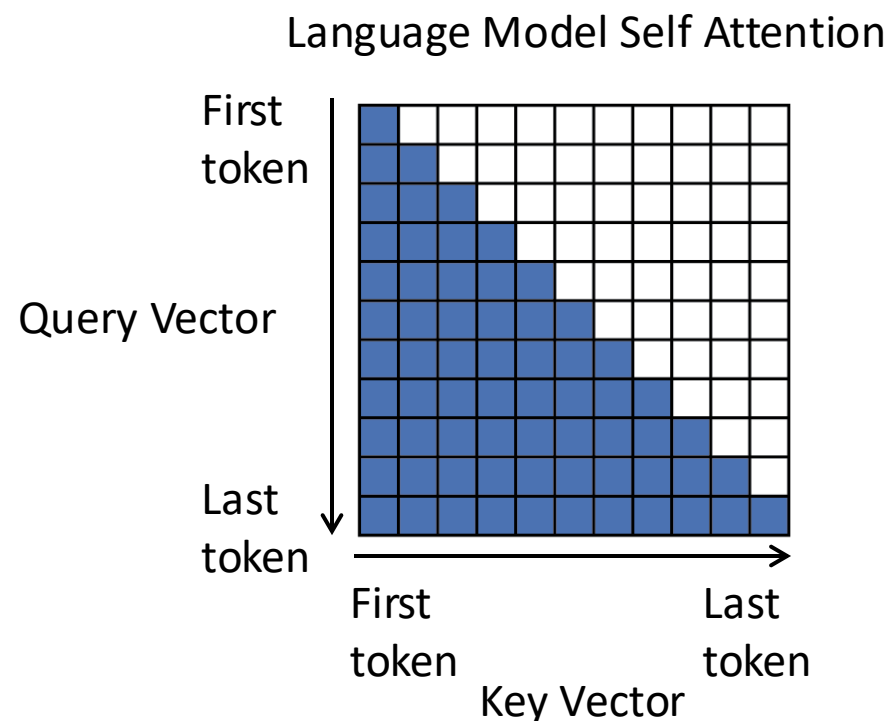
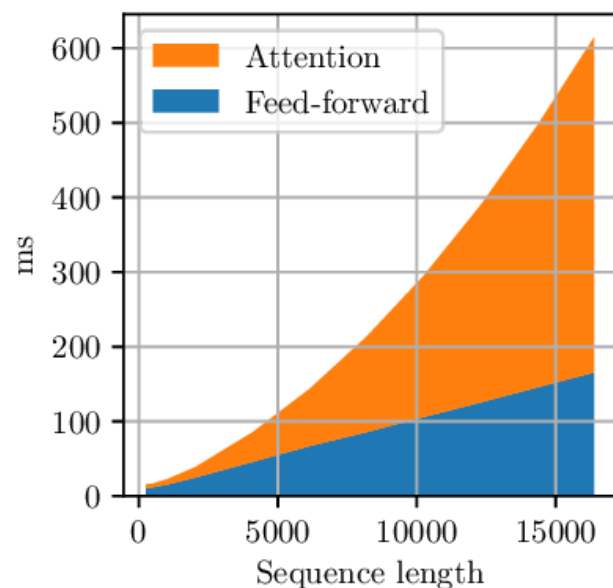


Task-specific supervision;
On target corpus








Efficient LLM Inference

- LLMs are computationally expensive
- Doing inference on a 7B LLM with full precision requires an 80GB GPU, and full-finetuning can use 4x80GP GPUs!
- LLM computation and memory costs grow quadratically with sequence length
- Efficient inference algorithms are needed for real-time resource-constrained environments such as mobile devices



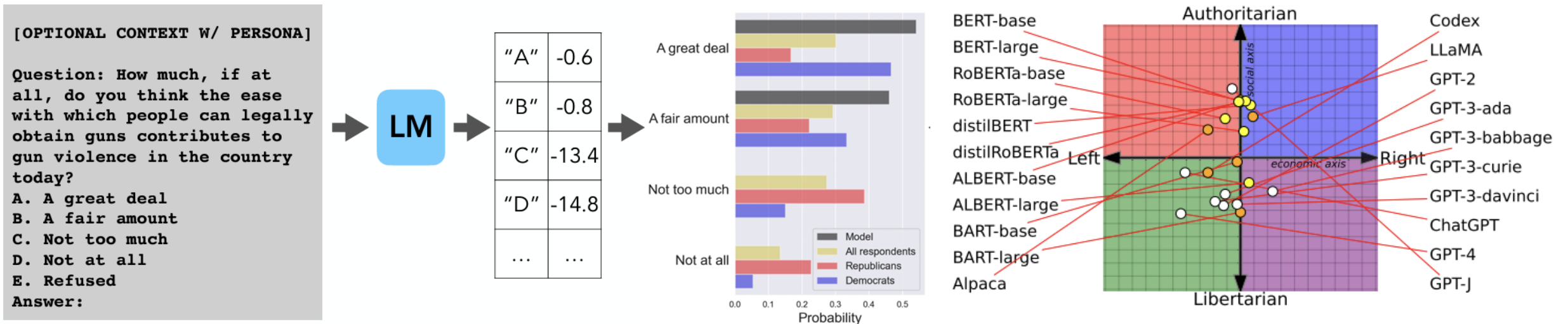
Topics: Multimodal Language Model

- Understand a combination of texts and images

 <p>This is a picture of two teddy bears on the moon.</p> <p>What are they doing?</p> <p>They are having a conversation.</p> <p>What object are they using?</p> <p>It looks like a computer.</p> <p>Is this surprising?</p> <p>Yes, it is surprising.</p> <p>Why is this picture surprising to you?</p> <p>I think it is surprising because teddy bears are not usually found on the moon.</p>	 <p>What is the common thing about these three images?</p> <p>They are all flamingos.</p> <p>What is the difference between these three images?</p> <p>The first one is a cartoon, the second one is a real flamingo, and the third one is a 3D model of a flamingo.</p>	 <p>This is an apple with a sticker on it.</p> <p>What does the sticker say?</p> <p>The sticker says "iPod".</p> <p>Where is the photo taken?</p> <p>It looks like it's taken in a backyard.</p> <p>Do you think it is printed or handwritten?</p> <p>It looks like it's handwritten.</p> <p>What color is the sticker?</p> <p>It's white.</p>	 <p>This is a cityscape. It looks like Chicago.</p> <p>What makes you think this is Chicago?</p> <p>I think it's Chicago because of the Shedd Aquarium in the background.</p>  <p>What about this one? Which city is this and what famous landmark helped you recognise the city?</p> <p>This is Tokyo. I think it's Tokyo because of the Tokyo Tower.</p>
---	---	---	---

Topics: Bias of Language Models

- Different language models may have different political views.



Content

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

What are language models?

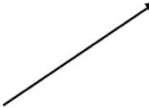
Language models

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

Autoregressive language models

- The chain rule of probability:
- $P(\text{Sally, fed, my, cat, with, meat}) = P(\text{Sally})$
 - * $P(\text{fed} \mid \text{Sally})$
 - * $P(\text{my} \mid \text{Sally, fed})$
 - * $P(\text{cat} \mid \text{Sally, fed, my})$
 - * $P(\text{with} \mid \text{Sally, fed, my, cat})$
 - * $P(\text{meat} \mid \text{Sally, fed, my, cat, with})$

Conditional probability

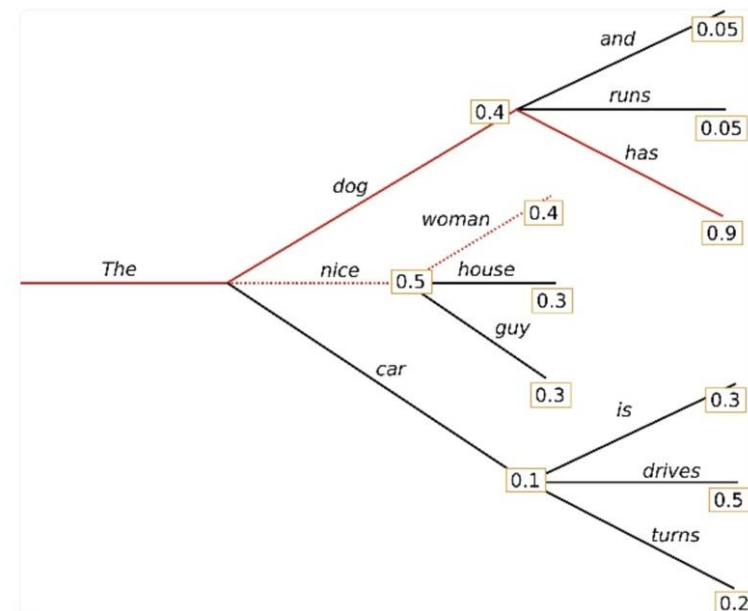
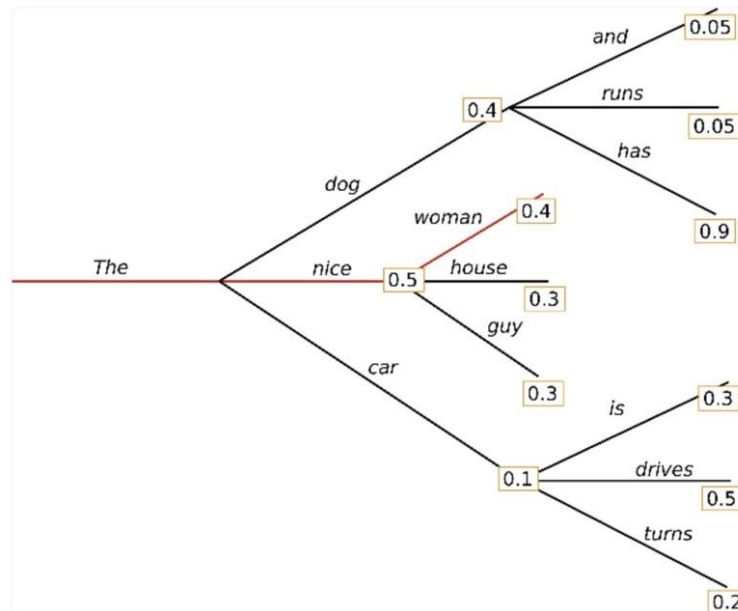
$$p(w_1, w_2, w_3, \dots, w_N) = p(w_1) p(w_2|w_1) p(w_3|w_1, w_2) \times \dots \times p(w_N|w_1, w_2, \dots, w_{N-1})$$


Sequence generation with language model

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

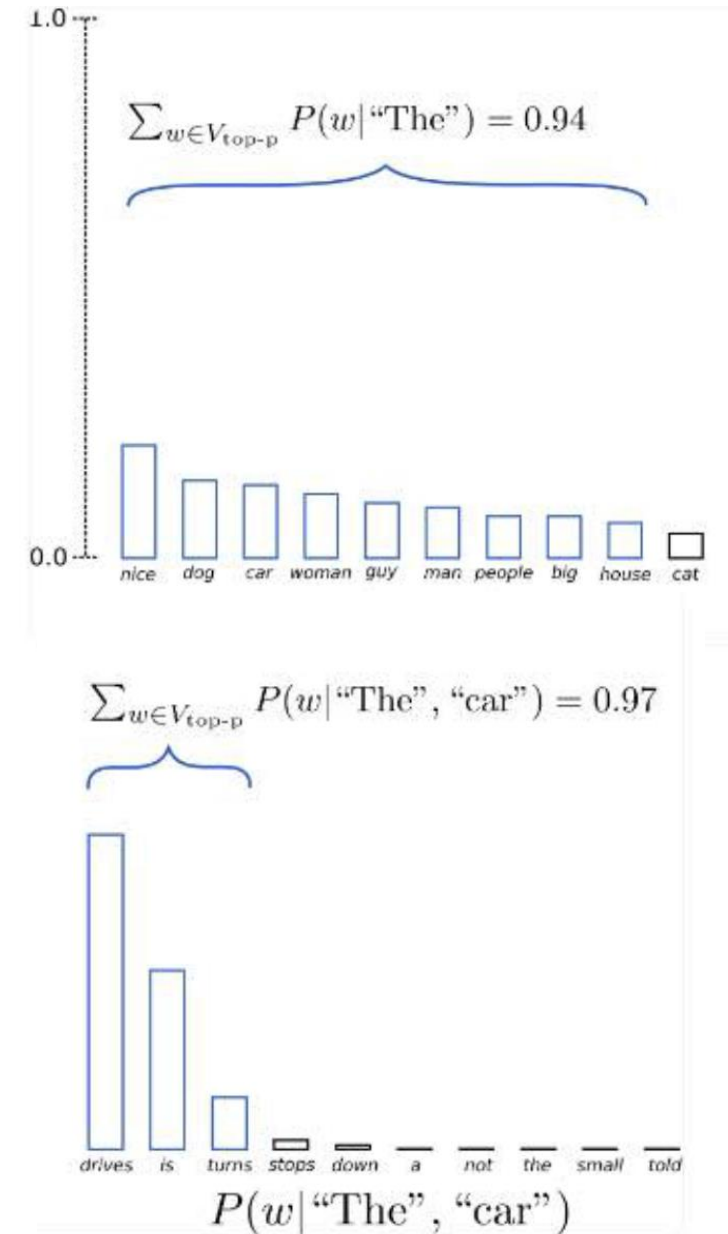
Different Decoding Approaches

- 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 $p(w_t | w_{[1:t-1]})$, 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 $p(w_t | w_{[1:t-1]})$, 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.

N-gram Language Models

- Bigram models

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

i	want	to	eat	chinese	food	lunch	spend
2533	927	2417	746	158	1093	341	278

$$P(\text{food} \mid \text{chinese}) = 82 / 158 = 0.519$$

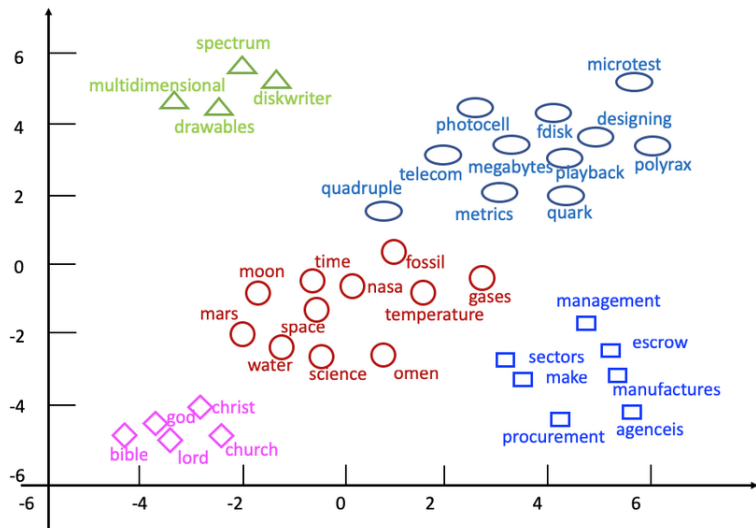
$$P(\text{chinese} \mid \text{want}) = 6 / 927 = 0.00647$$

Curse of Dimensionality

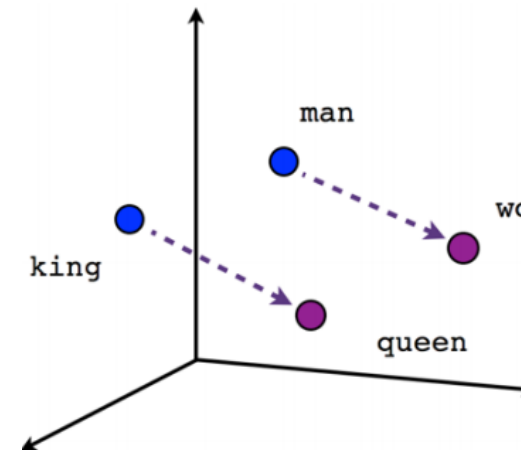
- Limitation of N-gram models
 - Limited Context Length: N-grams have a finite context window of length N , which means they cannot capture long-range dependencies or context beyond the previous $N-1$ words
 - Sparsity: As N increases, the number of possible N-grams grows exponentially, leading to sparse data and increased computational demands
 - Suppose vocabulary size is V , the number of possible N-grams increases to V^N .
 - Usually V (vocabulary size) could be more than ten thousand. Representing each word as a one-hot vector is inefficient.
 - “Dogs” and “cats” are more similar, compared to “dogs” and “rectangular”.

How to represent text more efficiently?

- Word Embedding: 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

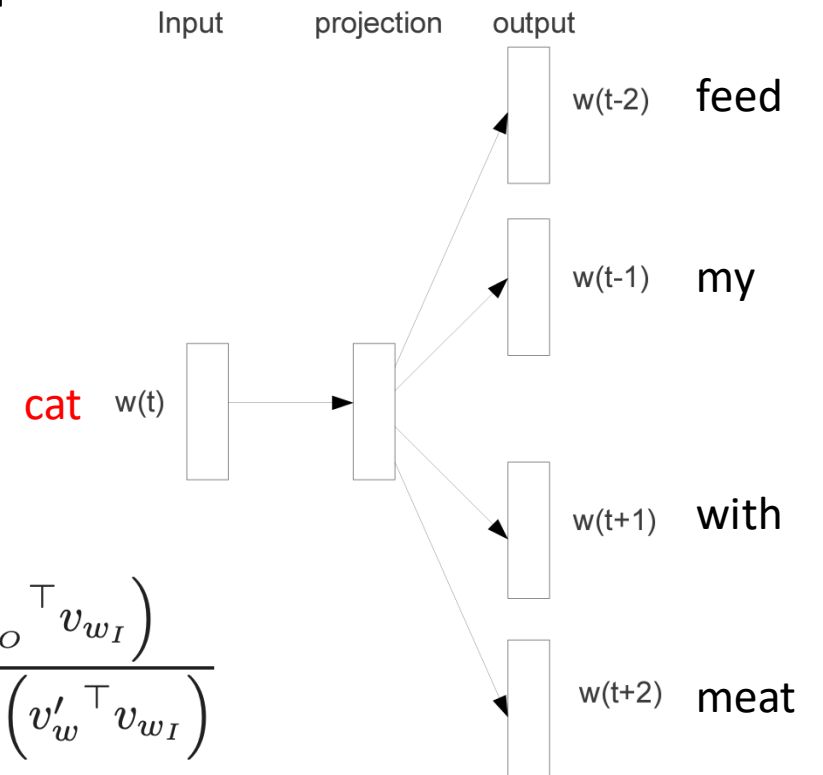
Distributed Representations: Word2Vec

- 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

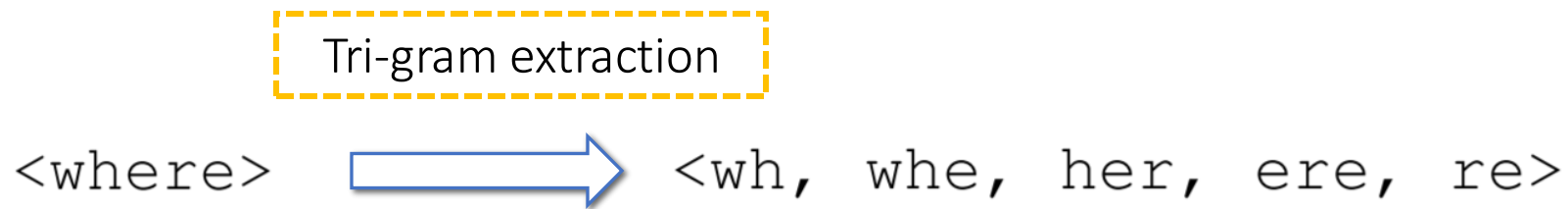
$$\frac{1}{T} \sum_{t=1}^T \sum_{-c \leq j \leq c, j \neq 0} \log p(w_{t+j} | w_t)$$

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



Considering Subwords: fastText

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



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

Word2Vec probability expression

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

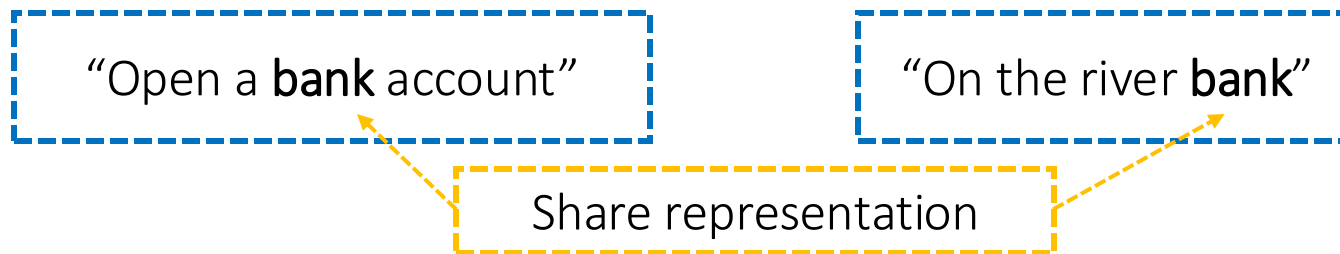
$$\sum_{g \in \mathcal{G}_w} \mathbf{z}_g^\top \mathbf{v}_c$$

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

N-gram embedding

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

Next Lecture: Self-Attention and Transformers