Lecture 18: LLMs: Prompting and Finetuning

Instructor: Swabha Swayamdipta

USC CSCI 544 Applied NLP

Oct 29, Fall 2024

Some slides adapted from Dan Jurafsky and Chris Manning and Qinyuan Ye and Justin Cho

Announcements

- Today: Tue, 10/29 - Project proposal
- Thu, 10/31 - Lecture + Paper Presentation I
- Tue, 11/5 - Lecture + Paper Presentation II
- Thu, 11/7 - Quiz 4 + Paper Presentation III
- Tue, 11/12 - Quiz 5 + Paper Presentation IV
- Thu, 11/14 Guest lecture on LLM Pretraining by Prof. Willie Neiswanger on 11/14 + HW4 due
  - Questions from lecture materials will be included in final exam
- Quizzes 4 and 5 - all topics after the midterms
  - Consider these as practice tests for final exams
- Paper Presentation:
  - Presenter will be announced couple of hours before the presentation (so please prepare accordingly!)
    - The presenter is responsible for the team's grade, so please ensure your teammates to be prepared!
  - Google Slides: Max 3 slides (share by 11:59 PM the night before)
  - Total time: 5 minutes (3 min presentation + 2 min QA)
  - Content:
    - Slide 1: Main Research Question in the paper,
    - Slide 2: Main Results Summarized,
    - Slide 3: How this influences your project.

Lecture Outline

- Announcements
- Recap: Decoding Algorithms
- Evaluating Generated Language
- LLMs
  - Pretraining
  - Post-training with Supervised Finetuning:
    - Instruction Tuning
  - Interacting with LLMs: Prompting
  - Post-training with Alignment with Human Feedback:
    - Preference Tuning: RLHF [Next Class]

Recap:
Natural Language Generation
- Search Algorithms

Beam Search Decoding

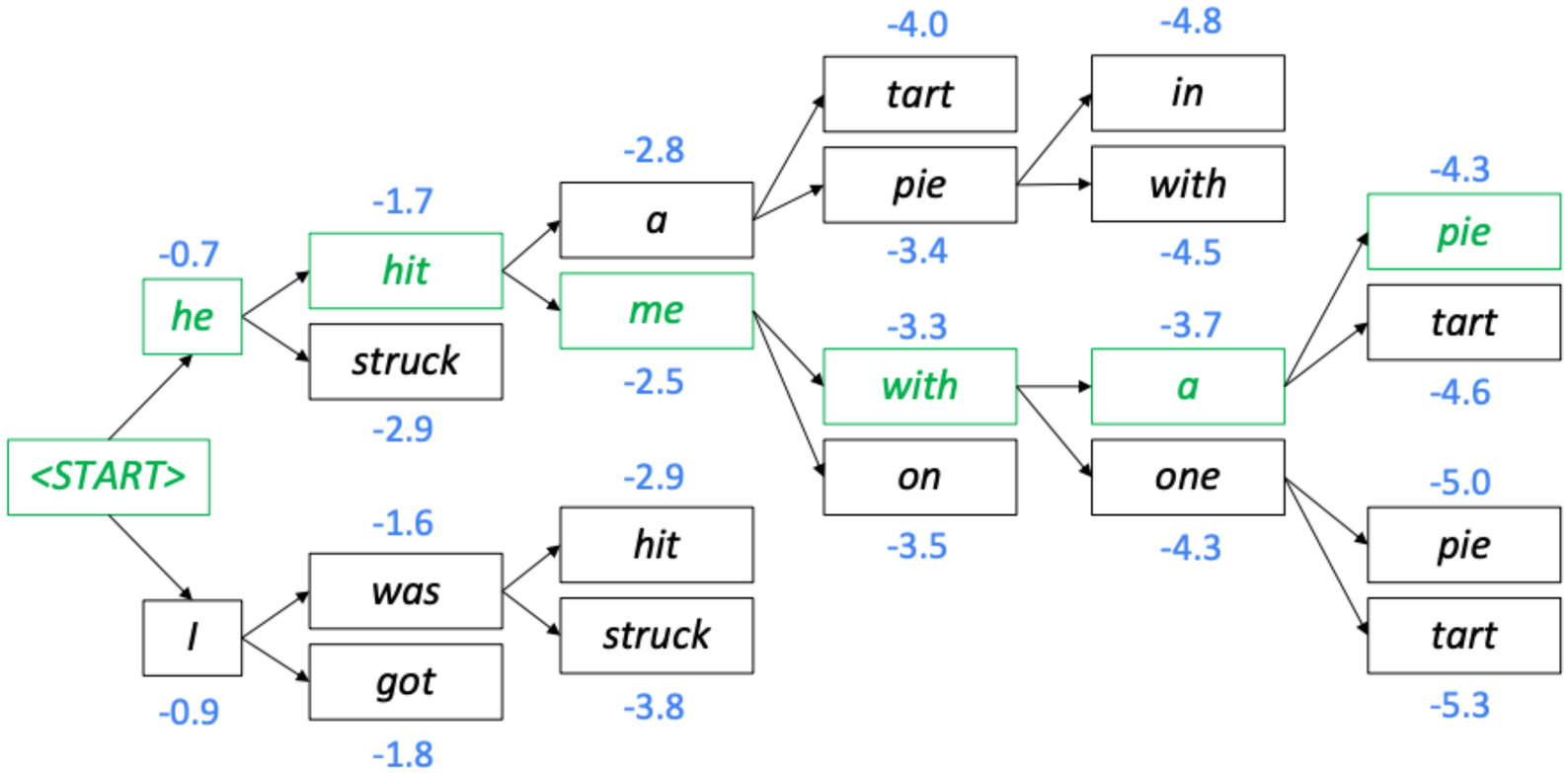
- Core idea: On each step of decoder, keep track of the k most probable partial translations (which we call hypotheses)
  - k is the beam size (in practice around 5 to 10, in NMT)
- A hypothesis has a score which is its log probability:

$$\text{score}(y_1, \ldots, y_t) = \log P_{\text{LM}}(y_1, \ldots, y_t | x) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$$

  - Scores are all negative, and higher score is better
  - We search for high-scoring hypotheses, tracking top k on each step
- Beam search is not guaranteed to find optimal solution
- But much more efficient than exhaustive search!

Beam Search Decoding: Example

Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$



Backtrack to obtain the full hypothesis

Slide credit: Chris Manning

Beam Search Decoding: Stopping Criterion

- Greedy Decoding is done until the model produces an </s> token
  - For e.g. <s> he hit me with a pie </s>
- In Beam Search Decoding, different hypotheses may produce </s> tokens at different time steps
  - When a hypothesis produces </s>, that hypothesis is complete.
  - Place it aside and continue exploring other hypotheses via beam search.
- Usually we continue beam search until:
  - We reach time step T (where T is some pre-defined cutoff), or
  - We have at least n completed hypotheses (where n is pre-defined cutoff)

Beam Search Decoding: Longer Hypotheses

- We have our list of completed hypotheses. Now how to select top one?
- Each hypothesis $y_1, \ldots, y_t$ on our list has a score

$$\text{score}(y_1, \ldots, y_t) = \log P_{\text{LM}}(y_1, \ldots, y_t | x) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$$

- Problem with this: longer hypotheses have lower score
- Fix: Normalize by length. Use this to select top one instead

$$\frac{1}{t} \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$$

Maximization Based Decoding

- Either greedy or beam search
- Beam search can be more effective with large beam width, but also more expensive
- Another key issue:

Generation can be bland or repetitive (also called degenerate)

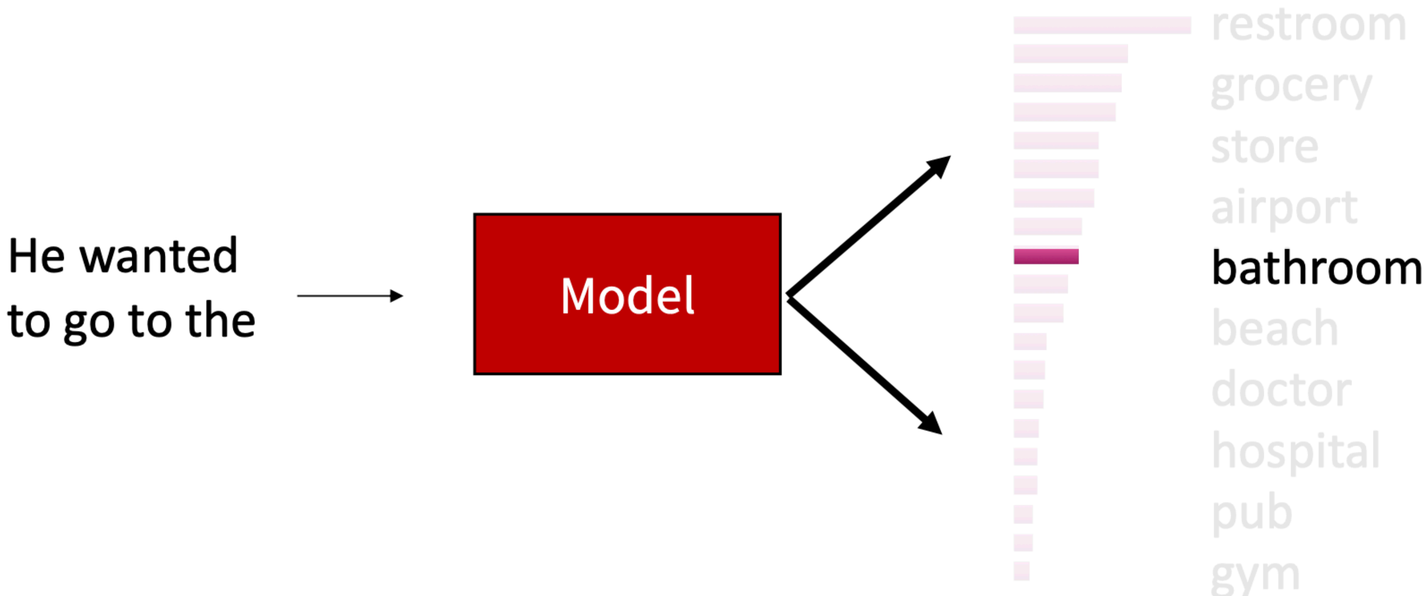
Context: In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.

Continuation: The study, published in the Proceedings of the National Academy of Sciences of the United States of America (PNAS), was conducted by researchers from the **Universidad Nacional Autónoma de México (UNAM)** and **the Universidad Nacional Autónoma de México (UNAM/Universidad Nacional Autónoma de México/ Universidad Nacional Autónoma de México/ Universidad Nacional Autónoma de México/ Universidad Nacional Autónoma de México...**

Holtzmann et al., 2020

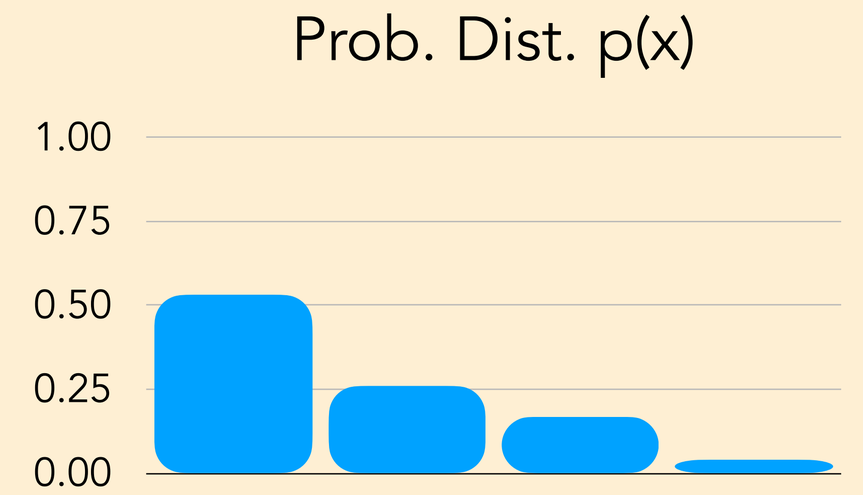
Solution: Don't Maximize, Pick a Sample

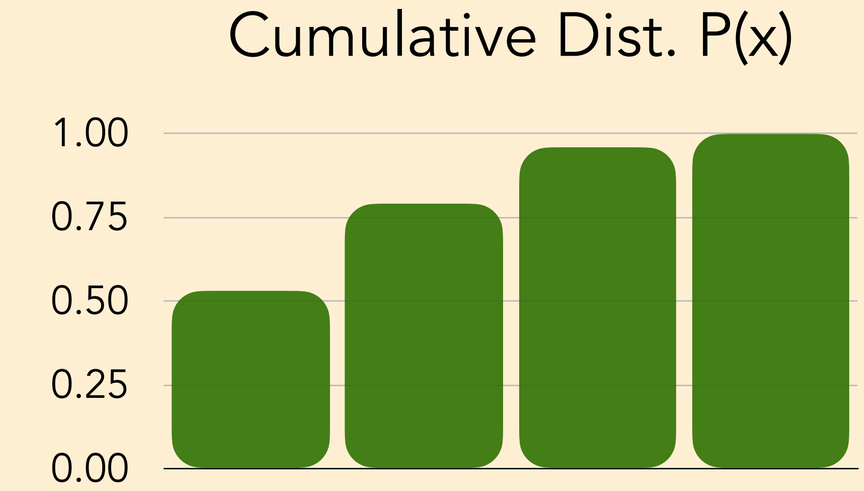
- Sample a token from the distribution of tokens.
- But this is not a truly random sample, it is a sample for the learned model distribution
  - Respects the probabilities, without going just for the maximum probability option
  - Or else, you would get something meaningless
  - Many good options which are not the maximum probability!



Sampling from a Distribution

- Choose x by randomly sampling from $p(x)$
  - Construct cumulative distribution function $P(x)$
  - Sample value q between 0-1 from a uniform random distribution
  - Pick x with largest $P(x)$ such that $P(x) \leq q$

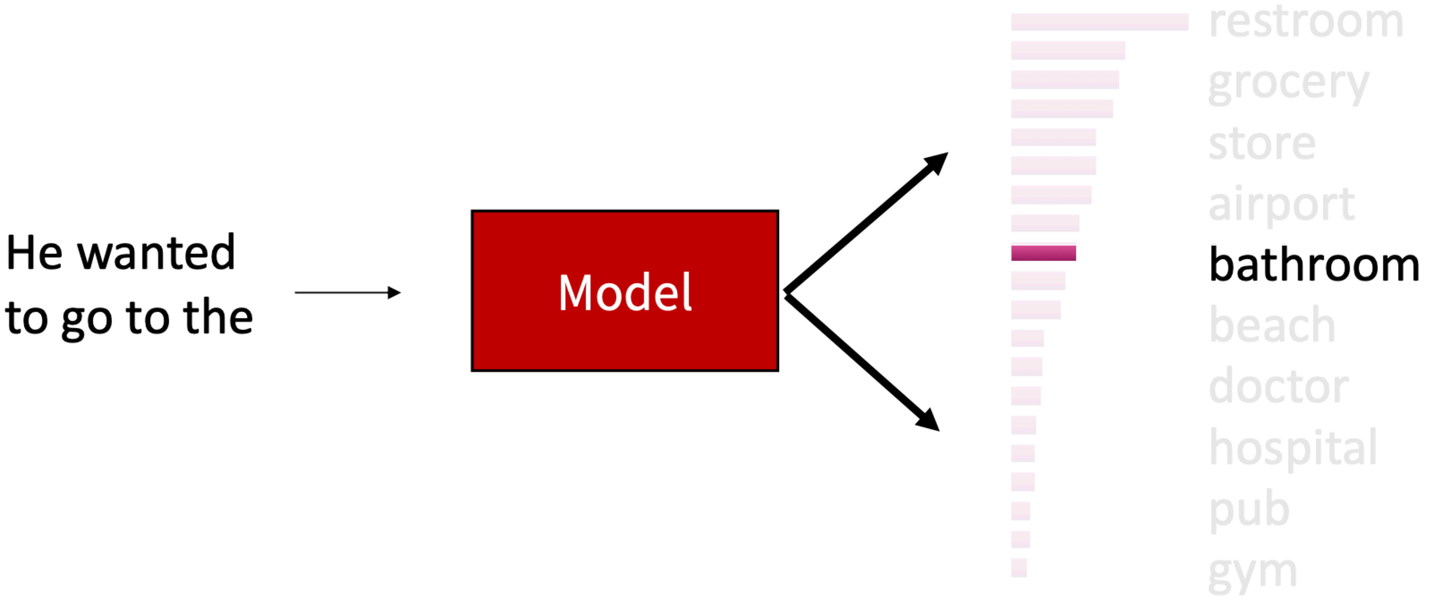




Pure / Ancestral Sampling

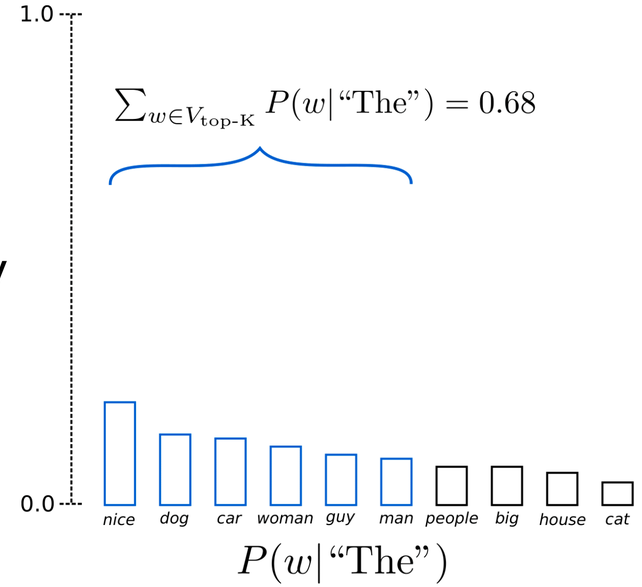
$$y_t \sim P_t(w) = \frac{\exp(S_w)}{\sum_{v \in V} \exp(S_v)}$$

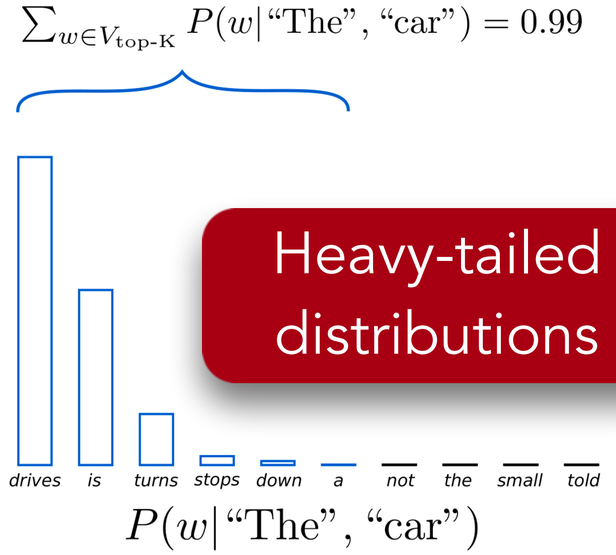
- Sample directly from P_t
- Still has access to the entire vocabulary
- But if the model distributions are of low quality, generations will be of low quality as well
- Often results in ill-formed generations
  - No guarantee of fluency



Top-K Sampling

- Problem: Ancestral sampling makes every token in the vocabulary an option
  - Even if most of the probability mass in the distribution is over a limited set of options, the tail of the distribution could be very long and in aggregate have considerable mass
  - Many tokens are probably really wrong in the current context. Yet, we give them individually a tiny chance to be selected.
  - But because there are many of them, we still give them as a group a high chance to be selected.
- Solution: Top-K sampling
  - Only sample from the top K tokens in the probability distribution





Heavy-tailed distributions

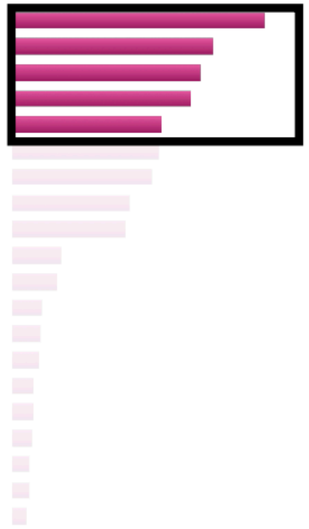
Fan et al., ACL 2018; Holtzman et al., ACL 2018

Image Source: Huggingface

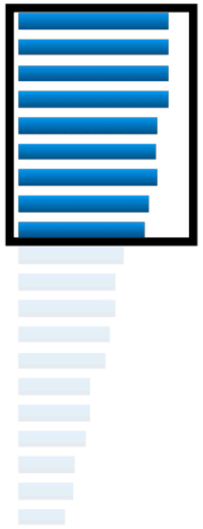
Nucleus (Top-P) Sampling

- Top-P sampling: Different threshold for different contexts
  - Sample from all tokens in the top P cumulative probability mass (i.e., where mass is concentrated)
  - Varies K depending on the uniformity of P_t

$P_t^1(y_t = w \mid \{y\}_{<t})$



$P_t^2(y_t = w \mid \{y\}_{<t})$

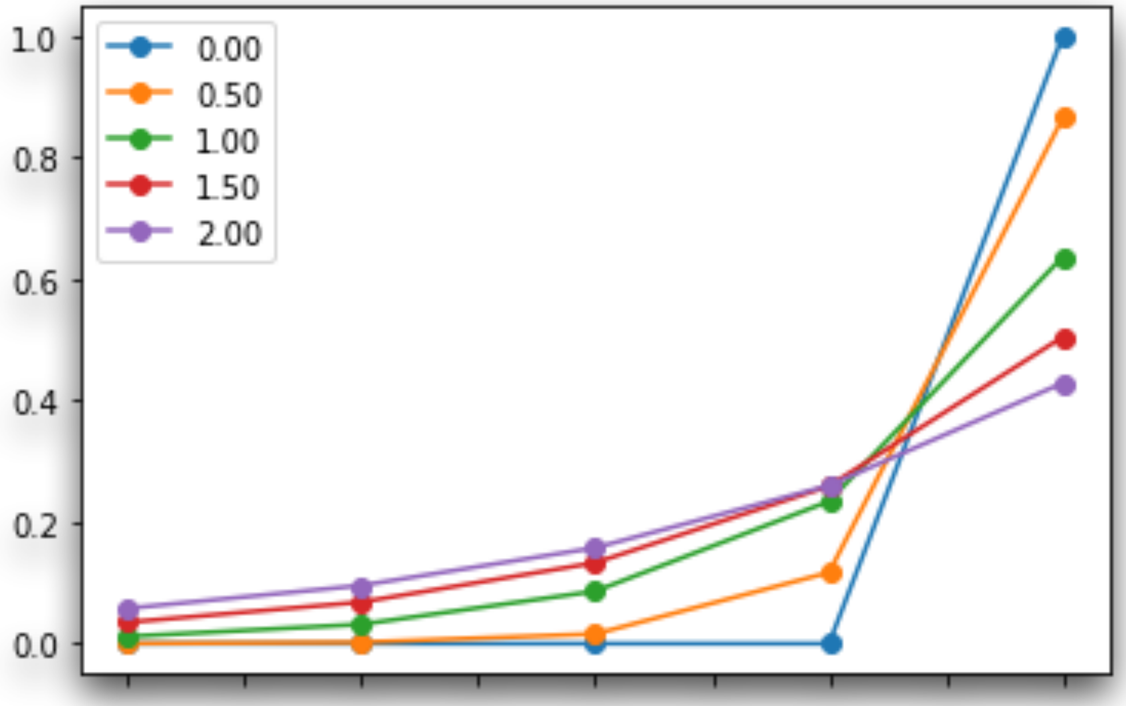


$P_t^3(y_t = w \mid \{y\}_{<t})$

Holtzman et al., ICLR 2020

Temperature Scaling

- We can apply a temperature hyperparameter τ to the softmax to rebalance P_t
- Unlike truncation-based sampling, temperature reshapes the probability distribution
- Most current approaches use this decoding: Ancestral sampling with temperature scaling



$$P(y_t = w) = \frac{\exp(S_w/\tau)}{\sum_{v \in V} \exp(S_v/\tau)}$$

Temperature is a hyperparameter for decoding: It can be tuned for both beam search and ancestral sampling.

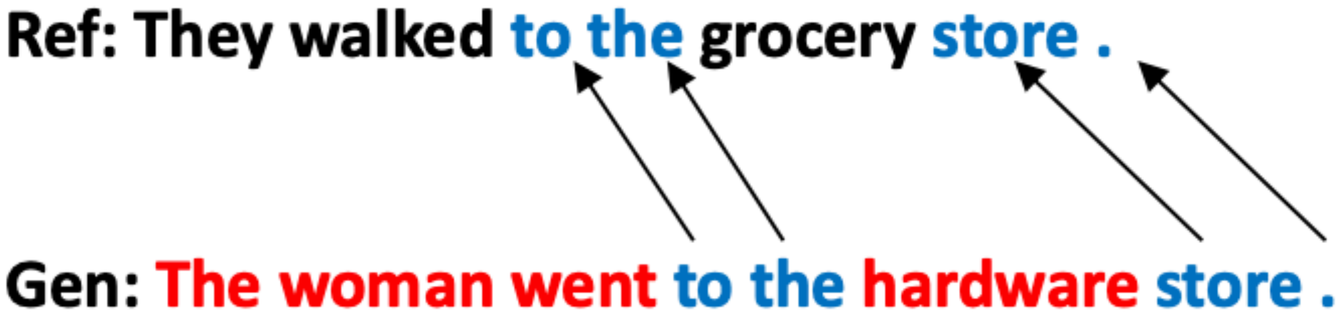
Lecture Outline

- Announcements
- Recap: Decoding Algorithms
- Evaluating Generated Language
- LLMs
  - Pretraining
  - Post-training with Supervised Finetuning:
    - Instruction Tuning
  - Interacting with LLMs: Prompting
  - Post-training with Alignment with Human Feedback:
    - Preference Tuning: RLHF [Next Class]

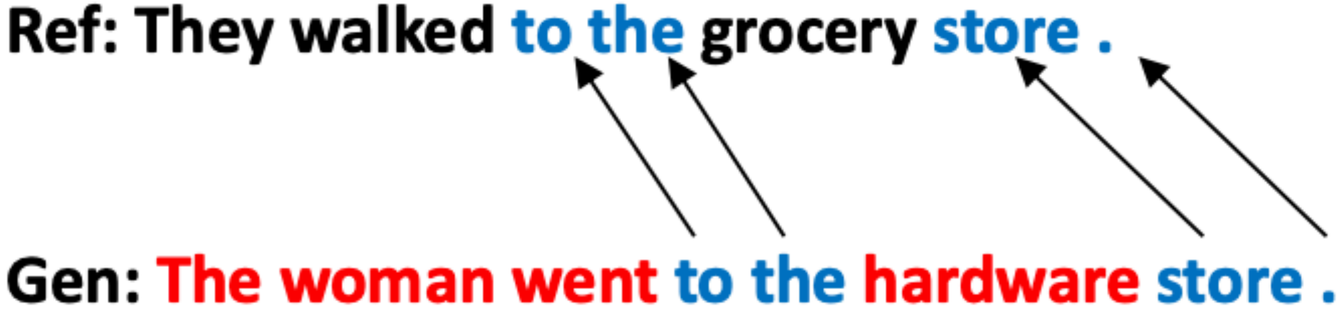
Evaluating Generations

Evaluation Strategies

- With Reference
  - Lexical Matching
  - Semantic Matching
- Without Reference
  - Perplexity
  - Model-Based Metrics
  - Advanced: Distributional Matching
  - Simplest, Most Reliable Strategy to-date: Human Evaluation
  - Even simpler and least reliable: Auto Evaluation



Reference-Based Metrics



- Only possible for close-ended generation tasks
- Compute a score that indicates the lexical similarity between generated and gold-standard (human-written) text
- Fast and efficient and widely used
- n-gram overlap metrics (e.g., BLEU, ROUGE, etc.)

BLEU

- Stands for Bilingual Evaluation Understudy
- BLEU compares the machine-written translation to one or several human-written translation(s), and computes a similarity score based on:
  - Geometric mean of n-gram precision (usually for 1, 2, 3 and 4-grams)
  - Plus a penalty for too-short system translations
- BLEU is useful but imperfect
  - There are many valid ways to translate a sentence
  - So a good translation can get a poor BLEU score because it has low n-gram overlap with the human translation
- Precision-based metric

Papineni et al., 2002

Precision, Recall and F-1

- True Positives, True Negatives, False Positives and False Negatives

$$\textbf{Precision} = \frac{TP}{TP + FP}$$

Of all the items in the prediction, how many match the ground truth

$$\textbf{Recall} = \frac{TP}{TP + FN}$$

Of all the items in the ground truth, how many are correctly predicted

$$F_1 = \frac{2 * PR}{P + R}$$

Harmonic Mean of Precision and Recall

Different value for different classes!

BLEU: Details

- Purely **precision-based** rather than combining precision and recall.
- BLEU score for a corpus of candidate translation sentences is a function of
  - the n-gram word precision over all the sentences
  - combined with a brevity penalty computed over the corpus as a whole.
- Consider a corpus composed of a single sentence
  - The unigram precision for this corpus is the percentage of unigram tokens in the candidate translation that also occur in the reference translation, and ditto for bigrams and so on, up to 4-grams
  - It computes this n-gram precision for unigrams, bigrams, trigrams, and 4-grams and takes the geometric mean
- Because BLEU is a word-based metric, it is very sensitive to word tokenization, making it impossible to compare different systems if they rely on different tokenization

$$p_n = \frac{\sum_{C \in \{Candidates\}} \sum_{n\text{-}gram \in C} Count_{clip}(n\text{-}gram)}{\sum_{C' \in \{Candidates\}} \sum_{n\text{-}gram' \in C'} Count(n\text{-}gram')}.$$

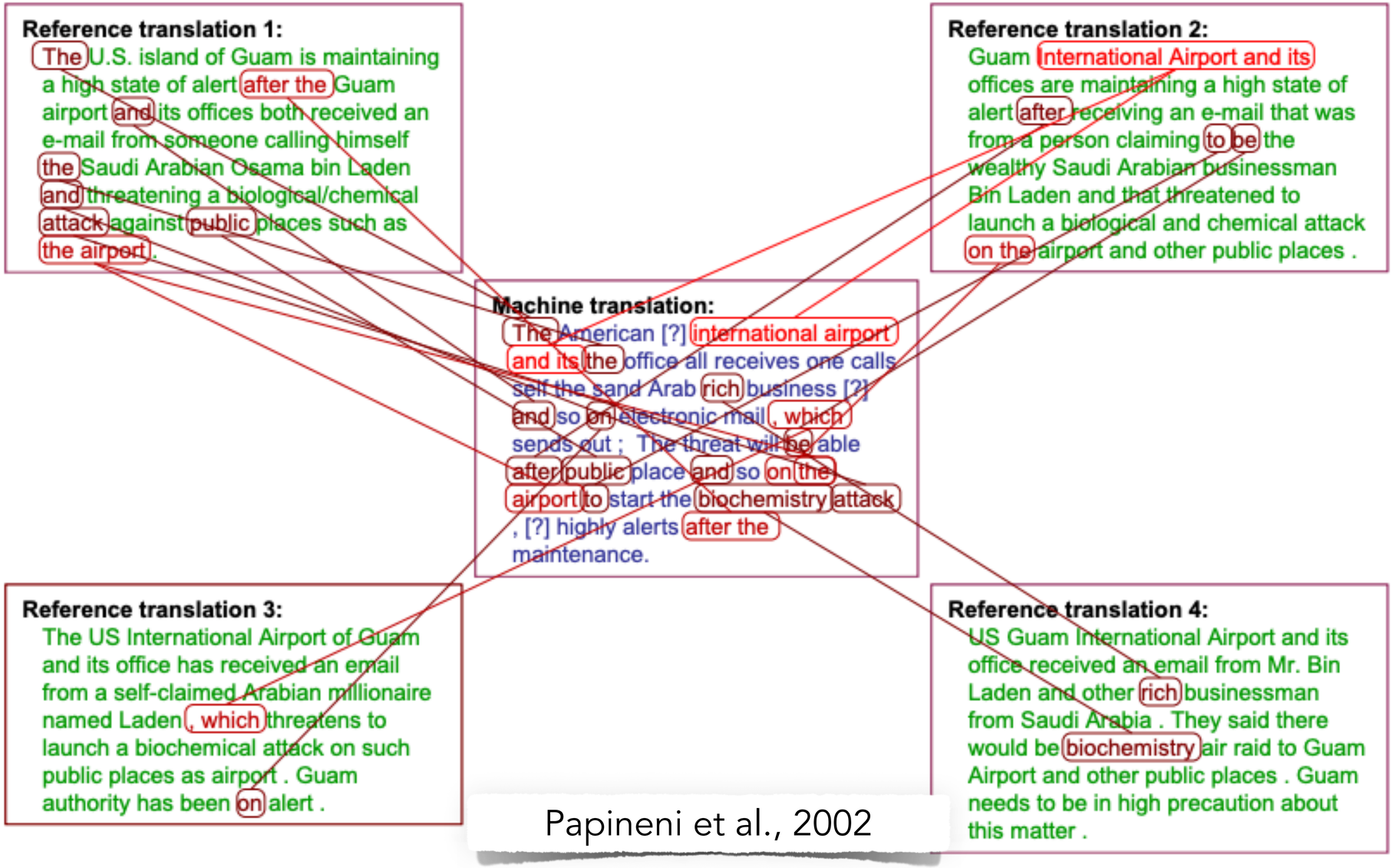
$$\mathrm{BP} = \begin{cases} 1 & \text{if } c > r \\ e^{(1-r/c)} & \text{if } c \leq r \end{cases}.$$

Then,

$$\mathrm{BLEU} = \mathrm{BP} \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right).$$

Papineni et al., 2002

BLEU: Example



Papineni et al., 2002

ROUGE

- Stands for “Recall-Oriented Understudy for Gisting Evaluation”
- Originally created for evaluating automatic summarization as well as machine translation
- Comparing an automatically produced summary or translation against a set of reference summaries (typically human-produced)
- Four variants:
  - ROUGE-N
  - ROUGE-L
  - ROUGE-S
  - ROUGE-W

ROUGE: A Package for Automatic Evaluation of Summaries (Lin, 2004)

ROUGE: Details

- **ROUGE-N**: measures **unigram**, **bigram**, **trigram** and higher order n-gram overlap
  - n-gram recall between a candidate summary and a set of reference summaries
- **ROUGE-L**: measures **longest matching sequence** of words using LCS.
  - Does not require consecutive matches but in-sequence matches that reflect sentence level word order.
  - Since it automatically includes longest in-sequence common n-grams, you don't need a predefined n-gram length.
- **ROUGE-S**: Is any pair of words in a sentence in order, allowing for arbitrary gaps.
  - Also be called skip-gram concurrence.
  - For example, **skip-bigram** measures the overlap of word pairs that can have a maximum of two gaps in between words. As an example, for the phrase *"cat in the hat"* the skip-bigrams would be *"cat in, cat the, cat hat, in the, in hat, the hat"*.
- **ROUGE-W**: Weighted Longest Common Subsequence

ROUGE: A Package for Automatic Evaluation of Summaries (Lin, 2004)

Evaluating Generation: Other Options

- Perplexity!
- Model-based Metrics (BERTScore, BARTScore, Word Mover's Distance, BLEURT)
  - Use learned representations of words and sentences to compute semantic similarity between generated and reference texts
  - No more n-gram bottleneck because text units are represented as embeddings!
  - The embeddings are pretrained, distance metrics used to measure the similarity can be fixed
- Automatic metrics fall short of matching human decisions
- So, Human Evaluation!

Human Evaluation

- Ask humans to evaluate the quality of generated text
  - Along specific axes: fluency, coherence / consistency, factuality and correctness, commonsense, etc.
  - Mostly done via crowdsourcing
- Human judgments are regarded as the gold standard
- Of course, we know that human eval is slow and expensive
- Beyond the cost of human eval, it's still far from perfect:
  - Humans Evaluation is hard:
    - Results are inconsistent / not reproducible
    - Can be subjective!
    - Misinterpret your question
    - Precision not recall

Rising Popularity: Automatic Evaluation

AlpacaFarm: A Simulation Framework for Methods that Learn from Human Feedback

Yann Dubois* Stanford
Xuechen Li* Stanford
Rohan Taori* Stanford
Tianyi Zhang* Stanford
Ishaan Gulrajani Stanford
Jimmy Ba University of Toronto
Carlos Guestrin Stanford
Percy Liang Stanford
Tatsunori B. Hashimoto Stanford

Cheap and theoretically consistent with human evaluation. BUT… reliability? Models evaluating their own generations may lead to weird mode collapsing effect

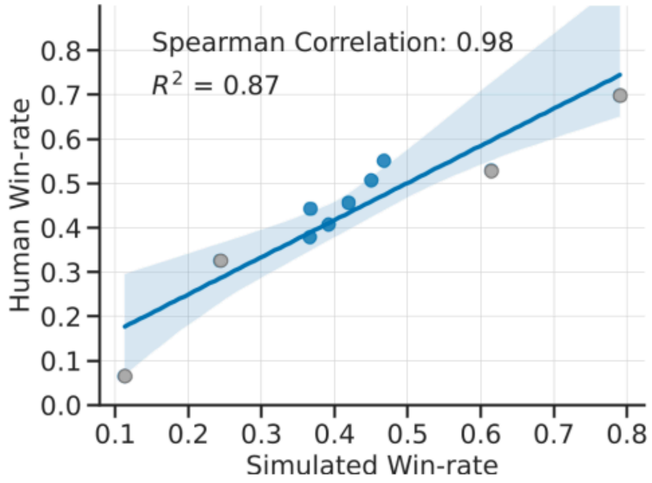


Figure 3: The ranking of methods trained and evaluated in AlpacaFarm matches that of methods trained and evaluated in the human-based pipeline. Each point represents one method M (e.g. PPO). The x-axis shows the simulated evaluation (win-rates measured by $p_{\text{sim}}^{\text{eval}}$) on methods trained in simulation M_{sim}. The y-axis shows human evaluation (win-rates measured by p_{human}) on methods trained with human feedback M_{human}. Gray points show models that we did not train, so their x and y values only differ in the evaluation (simulated vs human). Without those points, we have $R^2 = 0.83$ and a Spearman Correlation of 0.94.

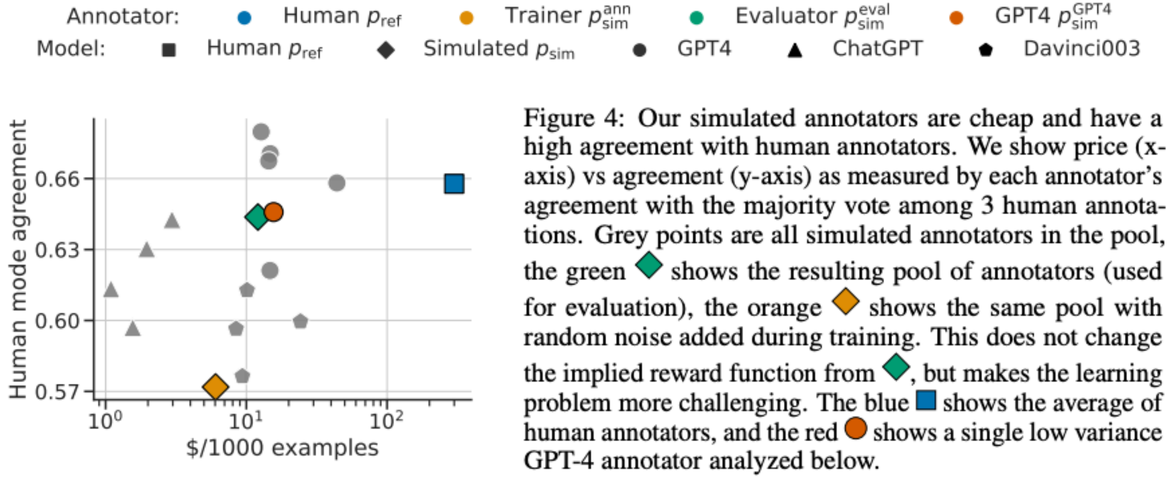


Figure 4: Our simulated annotators are cheap and have a high agreement with human annotators. We show price (x-axis) vs agreement (y-axis) as measured by each annotator's agreement with the majority vote among 3 human annotations. Grey points are all simulated annotators in the pool, the green ◆ shows the resulting pool of annotators (used for evaluation), the orange ◆ shows the same pool with random noise added during training. This does not change the implied reward function from ◆, but makes the learning problem more challenging. The blue ■ shows the average of human annotators, and the red ● shows a single low variance GPT-4 annotator analyzed below.

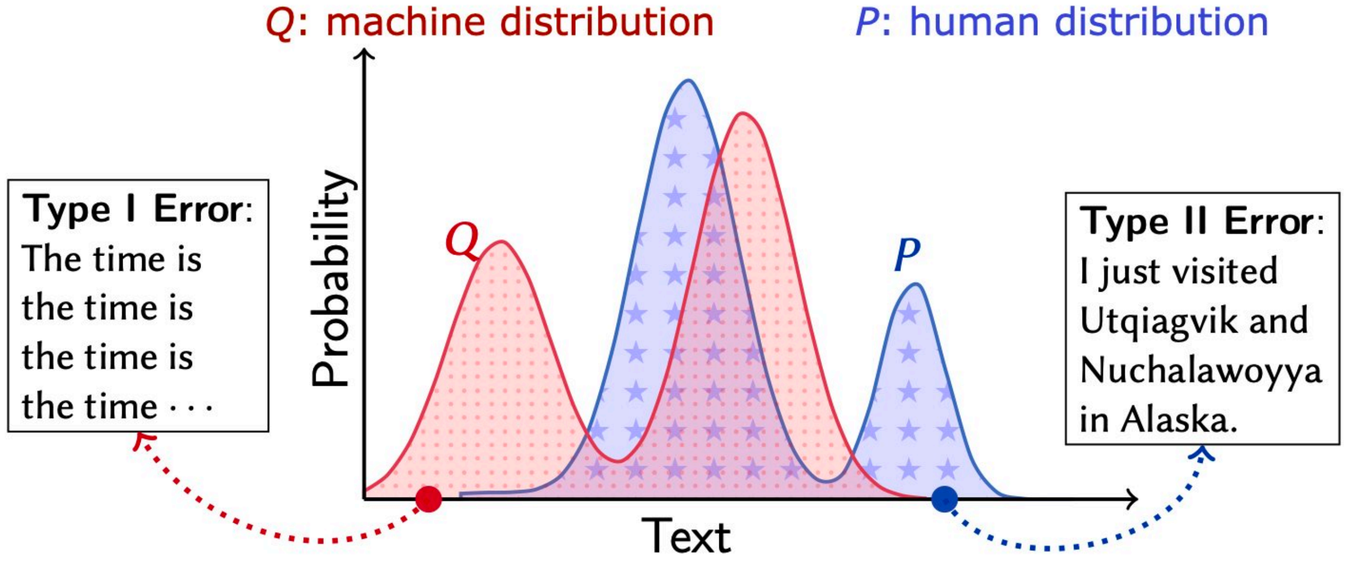
Evaluating Systems without References

- Compare human / natural language distributions to model-generated language distributions
- Divergence between these two distributions can be measured by MAUVE

MAUVE: Measuring the Gap Between Neural Text and Human Text using Divergence Frontiers

Krishna Pillutla[1] **Swabha Swayamdipta**[2] **Rowan Zellers**[1] **John Thickstun**[3]
Sean Welleck[1,2] **Yejin Choi**[1,2] **Zaid Harchaoui**[4]

[1]Paul G. Allen School of Computer Science & Engineering, University of Washington
[2]Allen Institute for Artificial Intelligence
[3]Department of Computer Science, Stanford University
[4]Department of Statistics, University of Washington



Natural Language Generation: Parting Thoughts

- Once trained, language models can be very powerful
  - The power only increases with scale
- So much so that most of our tasks in natural language can be seen as sequence completion tasks
  - Decoding Algorithms thus play a critical role
- How can you make LLMs do tasks (follow instructions)? **Instruction-Tuning and Preference-Tuning**
- **Prompting (or In-Context / Few-Shot Learning):** the ability to do many tasks with no gradient updates and no / a few examples, by simply:
  - Specifying the right sequence prediction problem
  - You can get interesting zero-shot behavior if you're creative enough with how you specify your task!

Lecture Outline

- Announcements
- Recap: Decoding Algorithms
- Evaluating Generated Language
- LLMs
  - Pretraining
  - Post-training with Supervised Finetuning:
    - Instruction Tuning
  - Interacting with LLMs: Prompting
  - Post-training with Alignment with Human Feedback:
    - Preference Tuning: RLHF [Next Class]

Large Language Models

What are Large Language Models?

- Models with many, many, parameters
  - Deeper layers. The largest T5 model had 11B parameters. GPT-3 has 175B parameters.
  - More context. GPT-2 has a context length of 1024 tokens, GPT-4 and Llama have 8192!
  - LLM are generally trained by filling the full context window with text. If documents are shorter than this, multiple documents are packed into the window with a special end-of-text token between them.
- Models trained on many, many tokens
  - Training data size is measured in # tokens
  - Batch size of the largest GPT-3 model is 3.2M tokens

OpenAI model's version	**GPT-3** (ada, babbage, curie, davinci)	**GPT-3.5** (gpt-3.5-turbo, gpt-3.5-turbo-0301,text-davinci-003, text-davinci-002)*	**GPT-4-8K**
Context length (max request)	2,049	4,096	8,192
Number of English words	~1,500	~3,000	~6,000
Number of single-spaced pages of English text	3	6	12

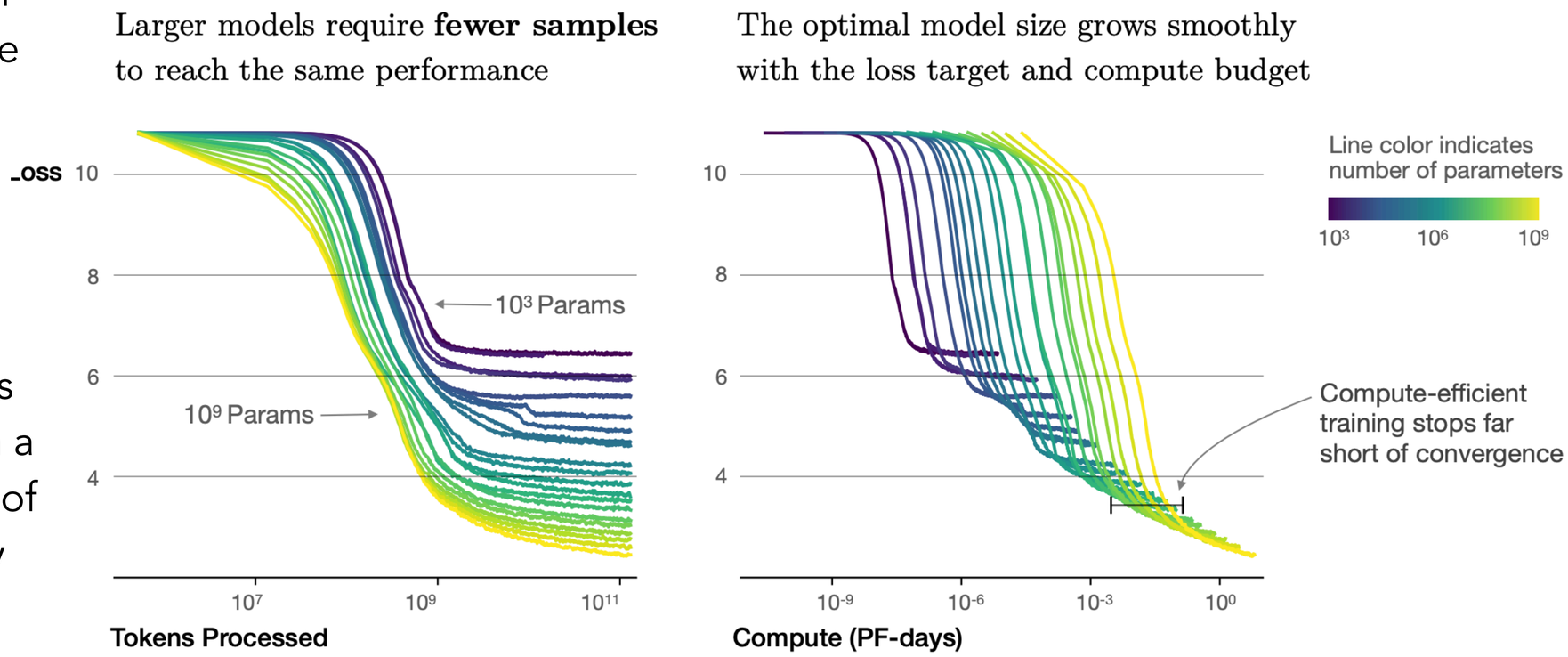
Source: Neoteric

Model	Size (# Parameters)	Training Tokens
LaMDA (Thoppilan et al., 2022)	137 Billion	168 Billion
GPT-3 (Brown et al., 2020)	175 Billion	300 Billion
Jurassic (Lieber et al., 2021)	178 Billion	300 Billion
Gopher (Rae et al., 2021)	280 Billion	300 Billion
MT-NLG 530B (Smith et al., 2022)	530 Billion	270 Billion
Chinchilla	70 Billion	1.4 Trillion

The Scaling "Laws" of LLMs

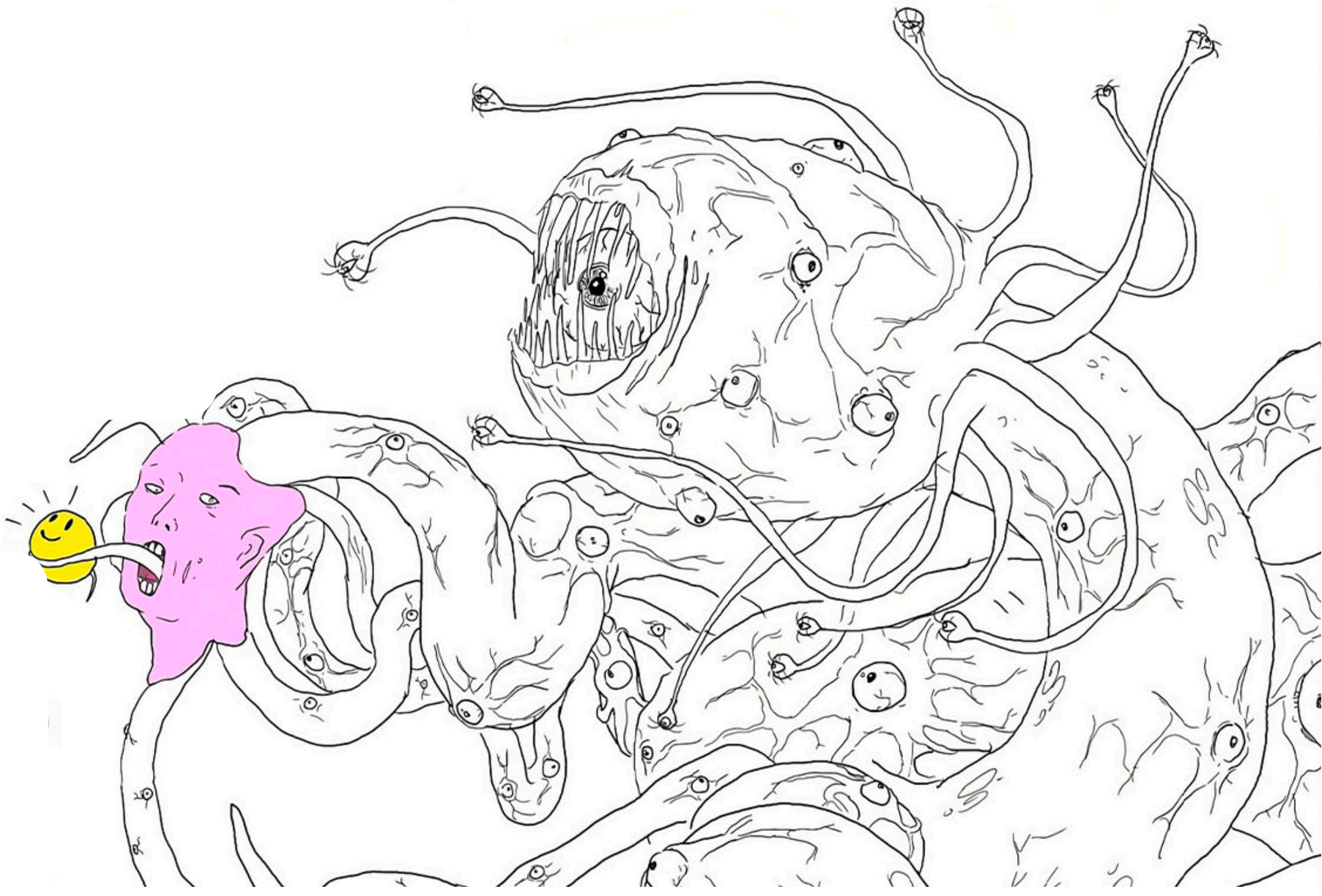
- Predictive rules of model performance, given parameter size, data size, computation
- The loss of an LLM scales as a power-law with each of these three properties of model training

- Other architectural details such as network width or depth have minimal effects within a wide range
- Determines the optimal allocation of a fixed compute budget
  - For instance, it can help us train very large models on a relatively modest amount of data and stop significantly before convergence
  - Or smaller models on larger data (e.g. Chinchilla LM)



Kaplan et al., 2020 https://arxiv.org/abs/2001.08361

Training LLMs



A significant, yet small part of the LM training phase (Next Class)

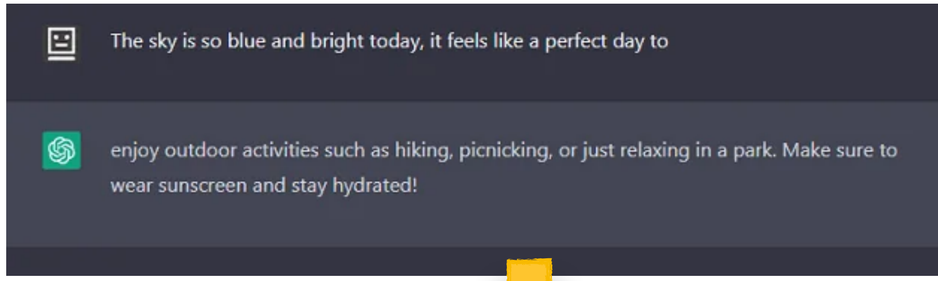
Shoggoth; slide credit: Justin Cho

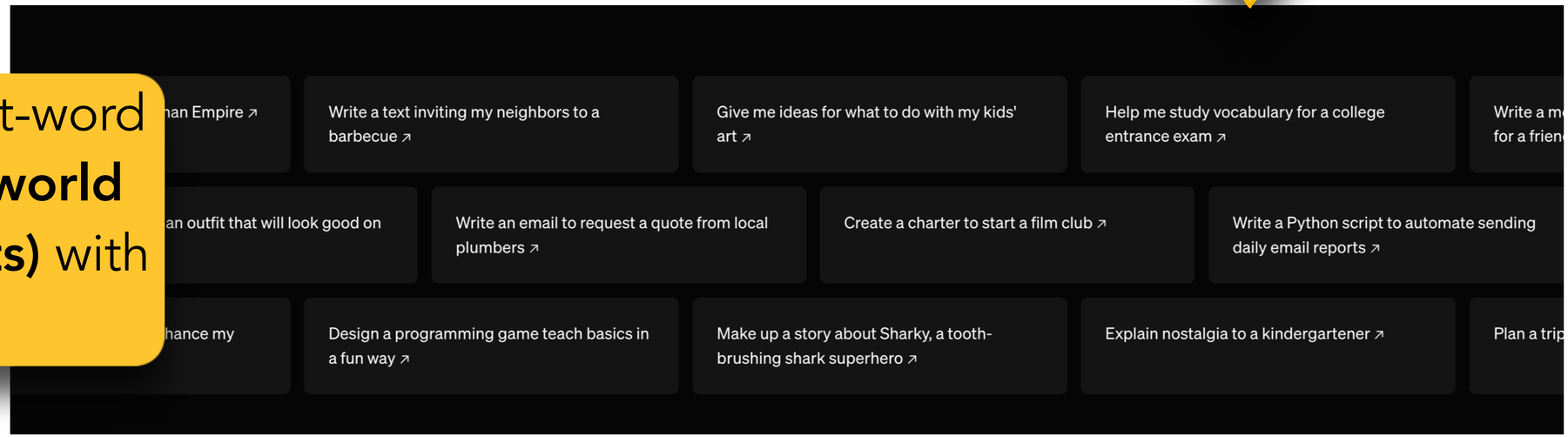
LLMs: Modern Training + Inference Recipe

- Training Recipe:
  - Stage 1: Pre-training on large corpus of text
    - This is called the base language model
    - Continued Pre-training for domain adaptation (optional)
  - Stage 2: Post-training
    - Instruction Tuning (Supervised Finetuning)
  - Stage 3: Post-training and Alignment
    - Reinforcement Learning with Human Feedback
    - Train a supervised classifier (reward model) on human demonstrations to provide feedback to LM
    - Supervised fine-tuning the LM with reinforcement learning to maximize rewards given by reward model
- Inference: Prompting with Instructions and Demonstrations (also called examples, shots)

Pre-training and Fine-tuning (Post-training)

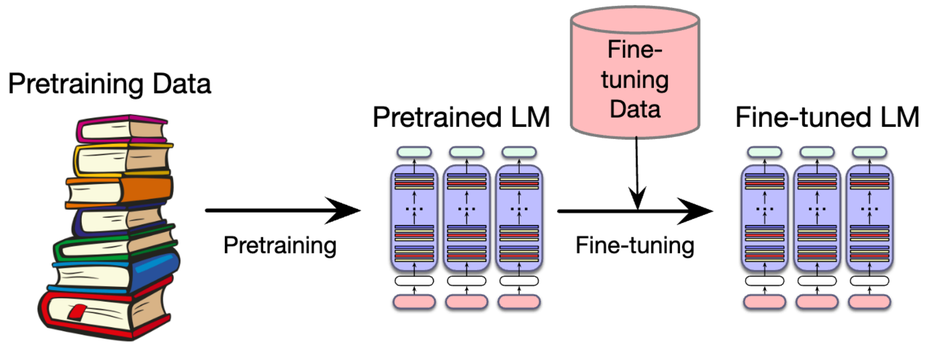
- Slightly different meaning than before
- Pre-training: Decoder-only models, standard next token prediction
- Fine-tuning: Supervised
  - **Instruction-Tuning**: Supervision is not necessarily via labels, but sequence pairs. Labels in standard NLP benchmarks can be converted into sequence pairs
  - **Preference-Tuning**: Collects human judgments / preferences as rewards
  - These steps are often called **post-training**





Post-training converts next-word completion models into **world models (agents, assistants)** with many capabilities!

Training Data for LLMs



- Pre- and post-training requires different kinds of data
- Pre-training needs a lot of raw text
  - Crawled from the Web: Natural Solution
  - But not all data crawled from the web is good for LM training
  - Needs to be filtered in various ways (deduplicated, removing non-natural language like code, sentences with offensive words from a blocklist).
  - Quality Filters! (Later lecture)
- Post-training / Fine-tuning
  - Instruction Tuning Data: Repurposed NLP / ML benchmarks
  - Preference Data for RLHF: Often human labels
  - Not easy to produce…

Pre-training Data

- Language models are trained on "raw text"
- To be highly capable (e.g., have linguistic and world knowledge), this text should span a **broad** range of domains, genres, languages, etc.
- A natural place (but not the only place) to look for such text is the **web**
  - Google search index is 100 petabytes; the actual web is likely even larger
  - P**rivate datasets** owned by big companies are even larger! WalMart generates 2.5 petabytes of data each hour!

- **Common Crawl** is a nonprofit organization that crawls the web and provides snapshots that are free to the public
  - Standard source of data to train many models such as T5, GPT-3, etc.
  - The April 2021 snapshot of Common Crawl has 320 TB
- The Colossal Clean Crawled Corpus (C4) is a larger was created to train the T5 model — 806 GB / 156 billion tokens

The Pile and DOLMA

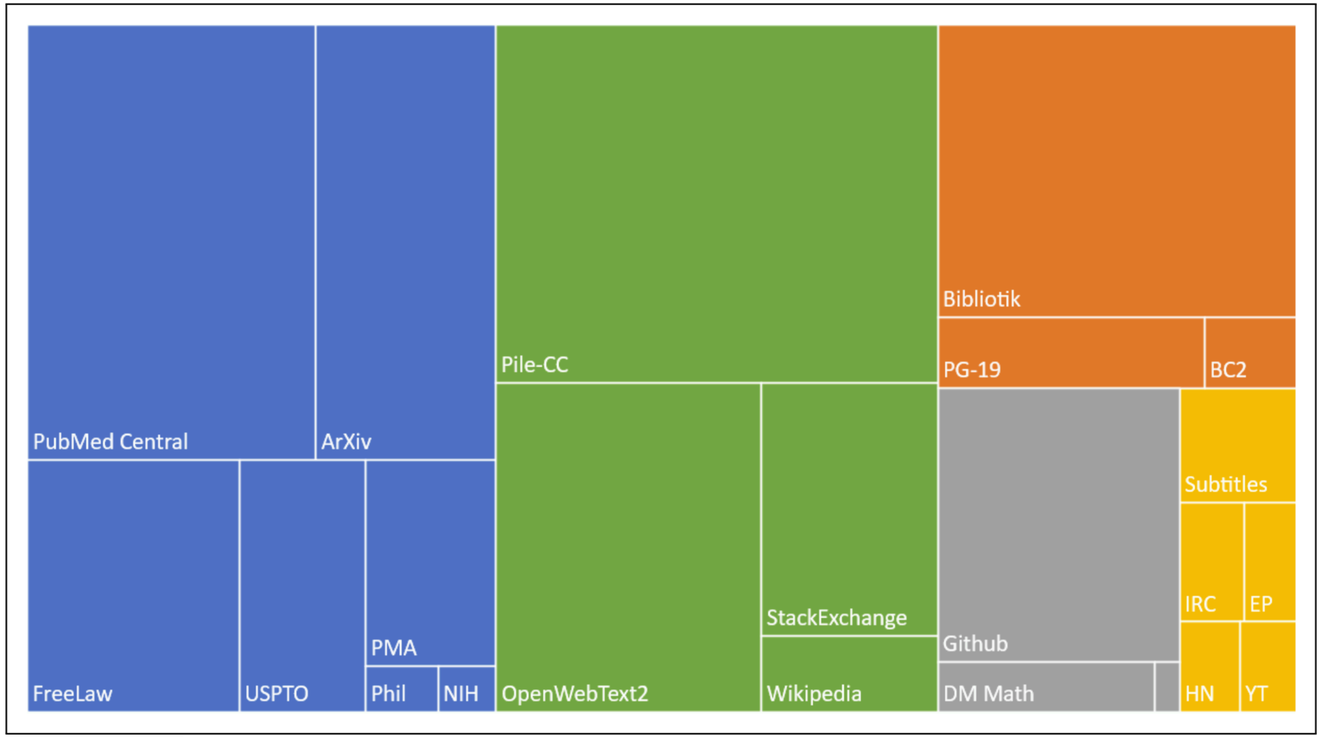


Figure 10.5 The Pile corpus, showing the size of different components, color coded as academic (articles from PubMed and ArXiv, patents from the USPTA; internet (webtext including a subset of the common crawl as well as Wikipedia), prose (a large corpus of books), dialogue (including movie subtitles and chat data), and misc.. Figure from Gao et al. (2020).

The Pile: 825 GB English text corpus containing a large amount of text scraped from the web, books and Wikipedia

Ai2

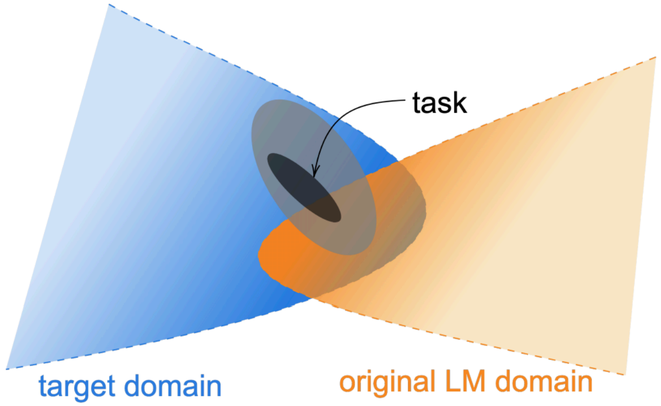
Dolma

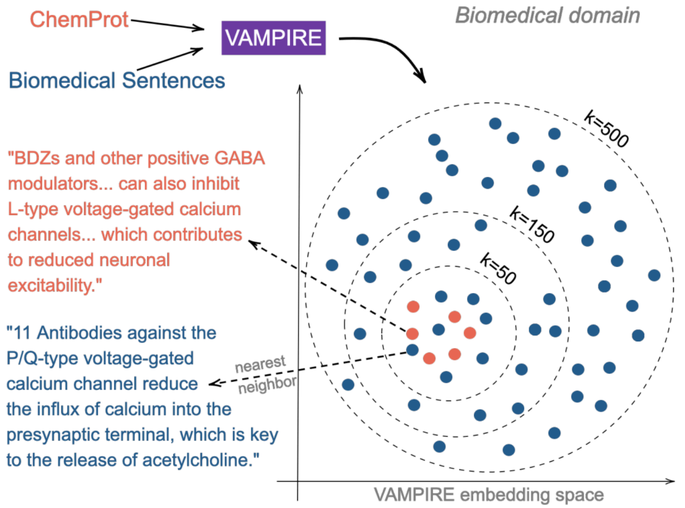
Source	Doc Type	UTF-8 bytes (GB)	Documents (millions)	Unicode words (billions)	Llama tokens (billions)
Common Crawl	web pages	9,812	3,734	1,928	2,479
GitHub	code	1,043	210	260	411
Reddit	social media	339	377	72	89
Semantic Scholar	papers	268	38.8	50	70
Project Gutenberg	books	20.4	0.056	4.0	6.0
Wikipedia, Wikibooks	encyclopedic	16.2	6.2	3.7	4.3
Total		**11,519**	**4,367**	**2,318**	**3,059**

Dolma is a larger open corpus of English, created with public tools, containing three trillion tokens, which similarly consists of web text, academic papers, code, books, encyclopedic materials, and social media (Soldaini et al., 2024)

Continued Pre-training

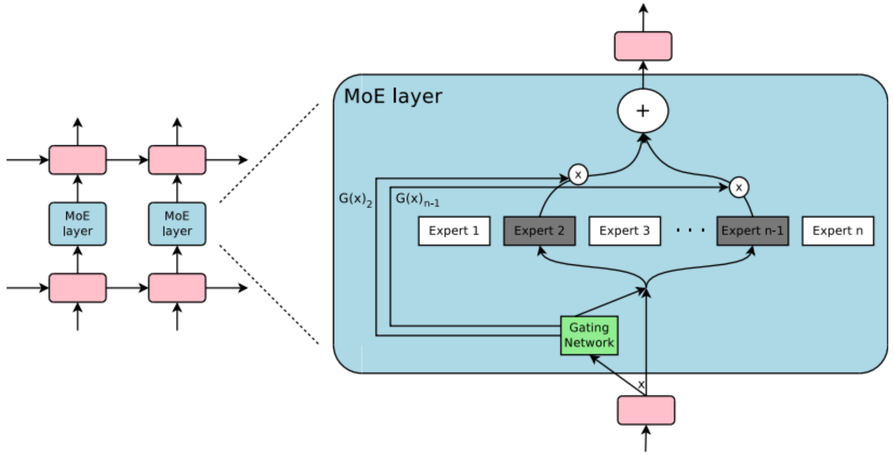
- LLMs are general domain. But we may need domain-specific LLMs…
  - For example, we might want a language model that's specialized to legal or medical text
- In such cases, we can simply continue training the model on relevant data from the new domain or language (Gururangan et al., 2020)
  - Next word prediction objective
- Works better when starting from a pretrained general-domain language model, as opposed to training from scratch
- It is even possible to do targeted data selection for obtaining domain-specific pretraining data (e.g. using k-means)



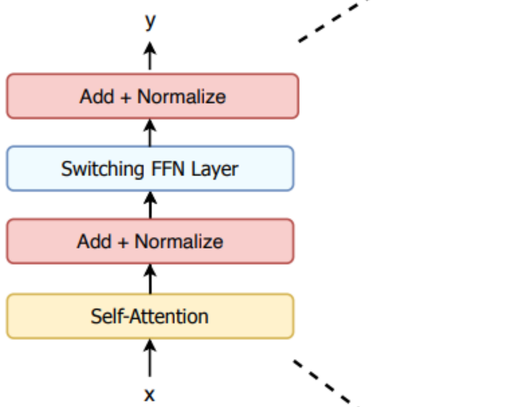


Gururangan, Marasović, **Swayamdipta** et al., 2020

LLMs as Mixtures of Experts



Source: Huggingface



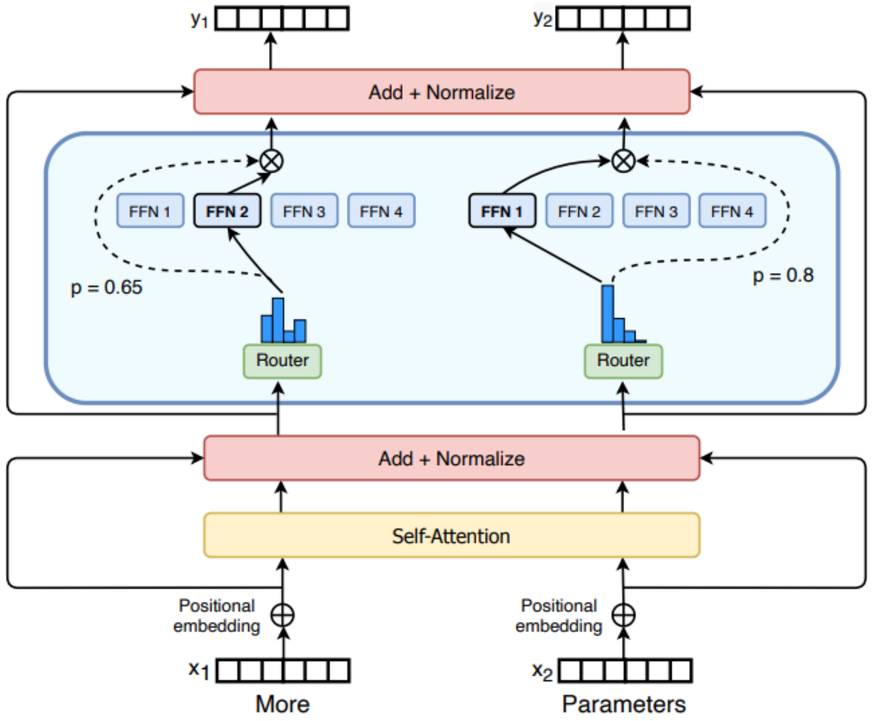
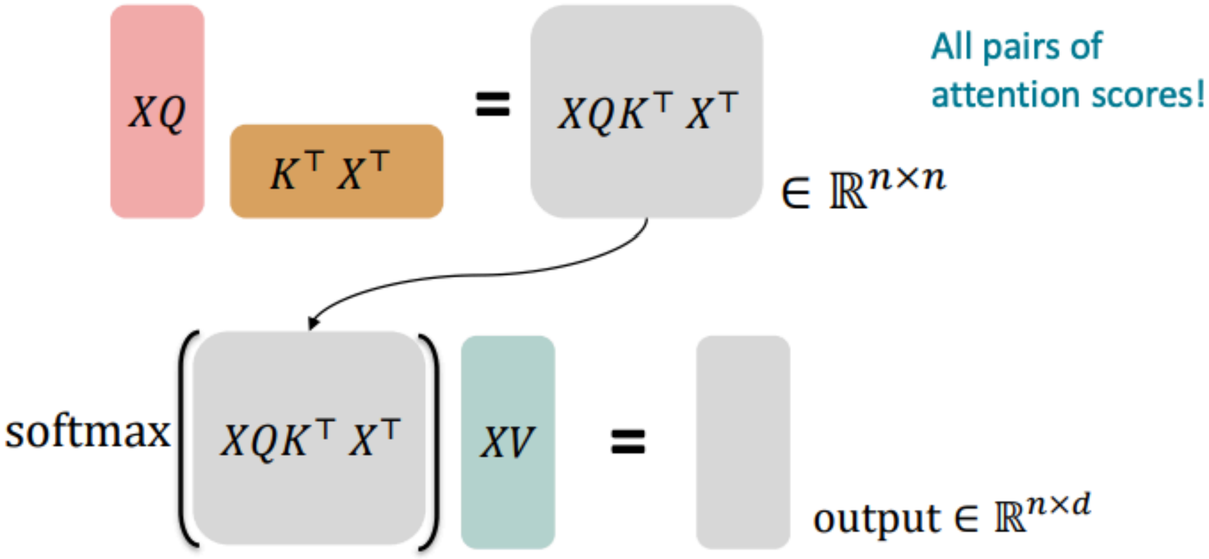


Figure 2: Illustration of a Switch Transformer encoder block. We replace the dense feed forward network (FFN) layer present in the Transformer with a sparse Switch FFN layer (light blue). The layer operates independently on the tokens in the sequence. We diagram two tokens (x_1 = "More" and x_2 = "Parameters" below) being routed (solid lines) across four FFN experts, where the router independently routes each token. The switch FFN layer returns the output of the selected FFN multiplied by the router gate value (dotted-line).

Switch Transformers; Fetus et al., 2021. (https://arxiv.org/abs/2101.03961)

LLM Inference: KV Cache

- Matrices / tensors for efficient computation cannot be applied during inference? Why?
  - At inference time, we iteratively generate the next tokens one at a time
  - Hence, we use key and value (and query) vectors
- KV Cache achieves inference-time speedup



- Instead of recomputing the key and value vectors for all the prior tokens $x_{<i}$, whenever we compute the key and value vectors we store them in memory in the KV cache, and then we can retrieve them from the cache as needed

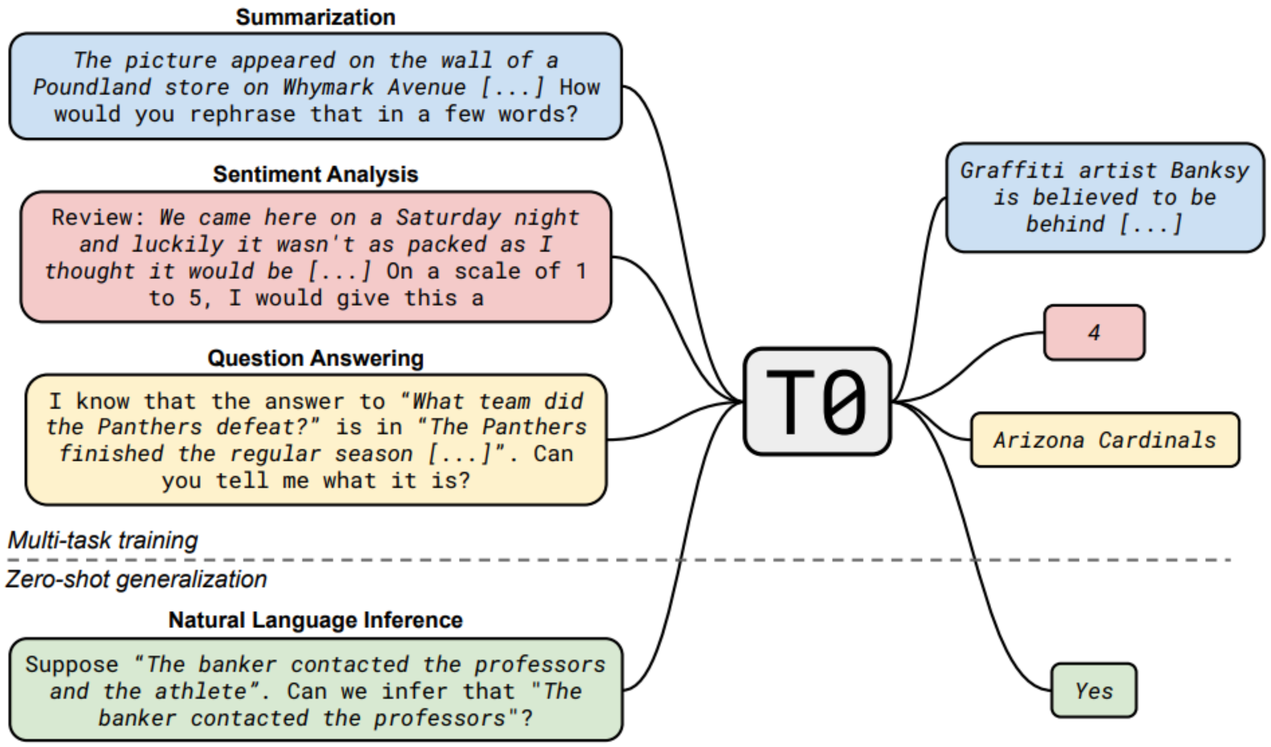
Lecture Outline

- Announcements
- Recap: Decoding Algorithms
- Evaluating Generated Language
- LLMs
  - Pretraining
  - Post-training with Supervised Finetuning:
    - Instruction Tuning
  - Interacting with LLMs: Prompting
  - Post-training with Alignment with Human Feedback:
    - Preference Tuning: RLHF [Next Class]

Supervised Fine-tuning LLMs: Instruction-Tuning

Instruction Tuning

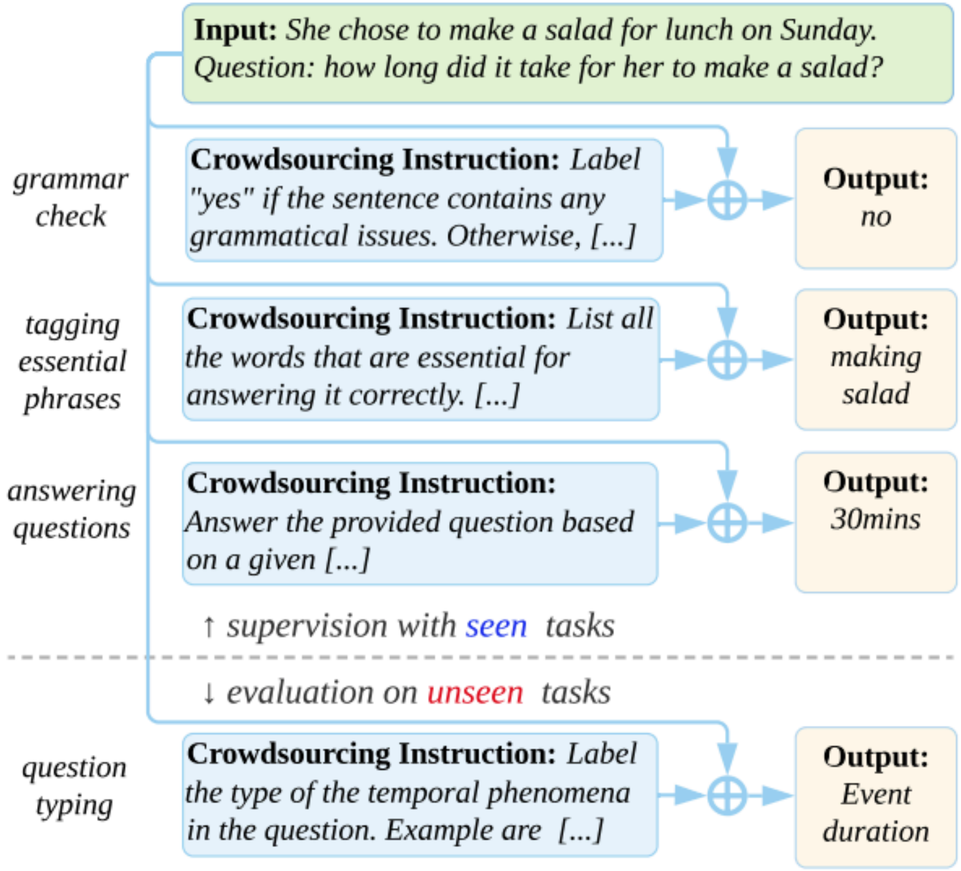
- Pretraining:
  - Train a model to continue a given context
- Instruction Tuning:
  - Train a model to follow varied instructions
  - Needed because the vast majority of pretraining is done on data which are not in the form of instructions
  - Fine-tuned (using the next-token-prediction objective) on a dataset of instructions together with correct responses



"Multitask Prompted Training Enables Zero-Shot Task Generalization" (Sahn et al., 2022)

Instruction Tuning and Task Generalization

- During training (supervised fine-tuning), the model learns to follow instructions of given tasks
- At test time, it generalizes to follow instructions on unseen tasks!



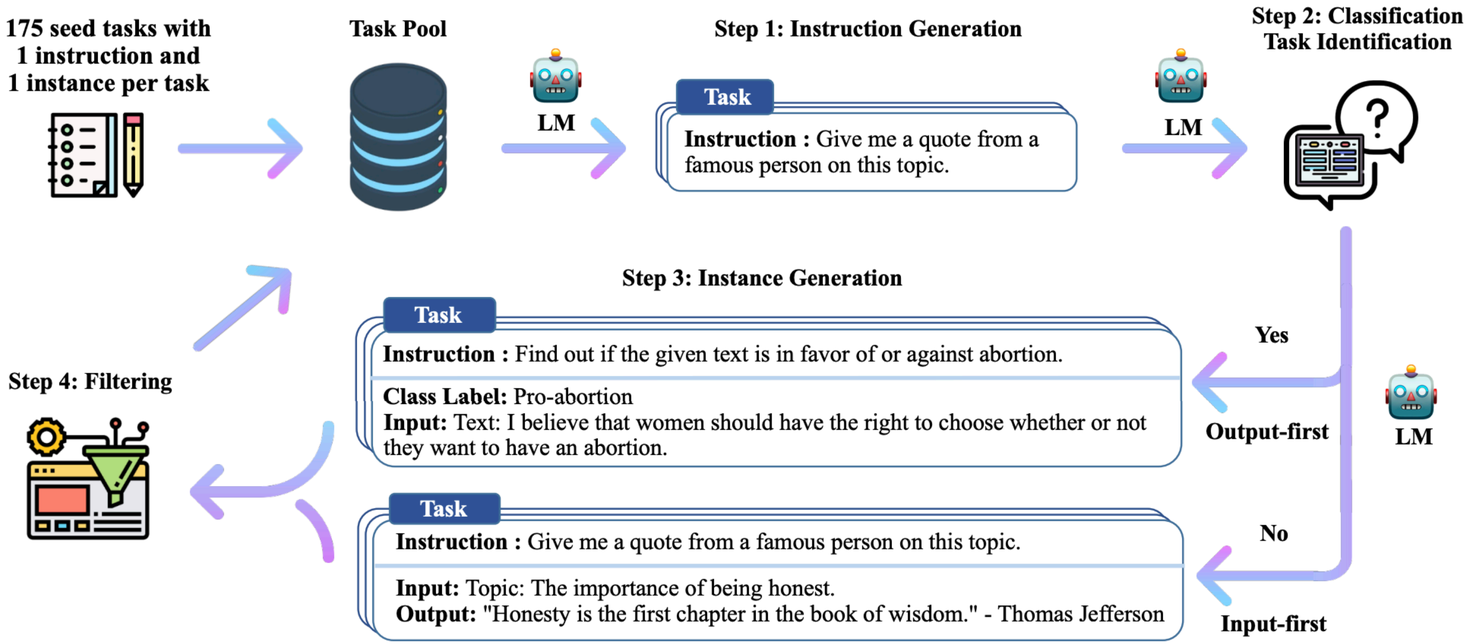
"Cross-Task Generalization via Natural Language Crowdsourcing Instructions" (Mishra et al., 2022)

Instruction Tuning Data

more data (instructions) → better model

Resource →	SUP-NATINST (this work)	NATINST (Mishra et al., 2022b)	CROSSFIT (Ye et al., 2021)	PROMPTSOURCE (Bach et al., 2022)	FLAN (Wei et al., 2022)	INSTRUCTGPT (Ouyang et al., 2022)
Has task instructions?	✓	✓	✗	✓	✓	✓
Has negative examples?	✓	✓	✗	✗	✗	✗
Has non-English tasks?	✓	✗	✗	✗	✓	✓
Is public?	✓	✓	✓	✓	✓	✗
Number of tasks	1616	61	269	176	62	–
Number of instructions	1616	61	–	2052	620	14378
Number of annotated tasks types	76	6	13	13*	12	10
Avg. task definition length (words)	56.6	134.4	–	24.8	8.2	–

"Super-NaturalInstructions: Generalization via Declarative Instructions on 1600+ NLP Tasks" (Wang et al., 2022) https://arxiv.org/abs/2212.10560



Diverse data (instructions) → better model

"Self-Instruct: Aligning Language Models with Self-Generated Instructions" (Wang et al., 2023)

Instruction Tuning Data

- Instruction tuning datasets are often created by repurposing standard NLP datasets for tasks like question answering or machine translation
- Often synthesized!
  - Prompting existing LLMs
- More variety in the instruction templates → better models!

		Model Details				Data Collection & Training Details			
Release	Collection	Model	Base	Size	Public?	Prompt Types	Tasks in Flan	# Exs	Methods
2020 05	UnifiedQA	UnifiedQA	RoBerta	110-340M	P	ZS	46 / 46	750k	
2021 04	CrossFit	BART-CrossFit	BART	140M	NP	FS	115 / 159	71.M	
2021 04	Natural Inst v1.0	Gen. BART	BART	140M	NP	ZS / FS	61 / 61	620k	+ Detailed k-shot Prompts
2021 09	Flan 2021	Flan-LaMDA	LaMDA	137B	NP	ZS / FS	62 / 62	4.4M	+ Template Variety
2021 10	P3	T0, T0+, T0++	T5-LM	3-11B	P	ZS	62 / 62	12M	+ Template Variety + Input Inversion
2021 10	MetaICL	MetaICL	GPT-2	770M	P	FS	100 / 142	3.5M	+ Input Inversion + Noisy Channel Opt
2021 11	ExMix	ExT5	T5	220M-11B	NP	ZS	72 / 107	500k	+ With Pretraining
2022 04	Super-Natural Inst.	*Tk*-Instruct	T5-LM, mT5	11-13B	P	ZS / FS	1556 / 1613	5M	+ Detailed k-shot Prompts + Multilingual
2022 10	GLM	GLM-130B	GLM	130B	P	FS	65 / 77	12M	+ With Pretraining + Bilingual (en, zh-cn)
2022 11	xP3	BLOOMz, mT0	BLOOM, mT5	13-176B	P	ZS	53 / 71	81M	+ Massively Multilingual
2022 12	Unnatural Inst.†	T5-LM-Unnat. Inst.	T5-LM	11B	NP	ZS	~20 / 117	64k	+ Synthetic Data
2022 12	Self-Instruct†	GPT-3 Self Inst.	GPT-3	175B	NP	ZS	Unknown	82k	+ Synthetic Data + Knowledge Distillation
2022 12	OPT-IML Bench†	OPT-IML	OPT	30-175B	P	ZS + FS, CoT	~2067 / 2207	18M	+ Template Variety + Input Inversion + Multilingual
2022 10	Flan 2022 (ours)	Flan-T5, Flan-PaLM	T5-LM, PaLM	10M-540B	P NP	ZS + FS, CoT	1836	15M	+ Template Variety + Input Inversion + Multilingual

"The Flan Collection: Designing Data and Methods for Effective Instruction Tuning" (Longpre et al., 2023)

Instruction Tuning: Masking Instructions

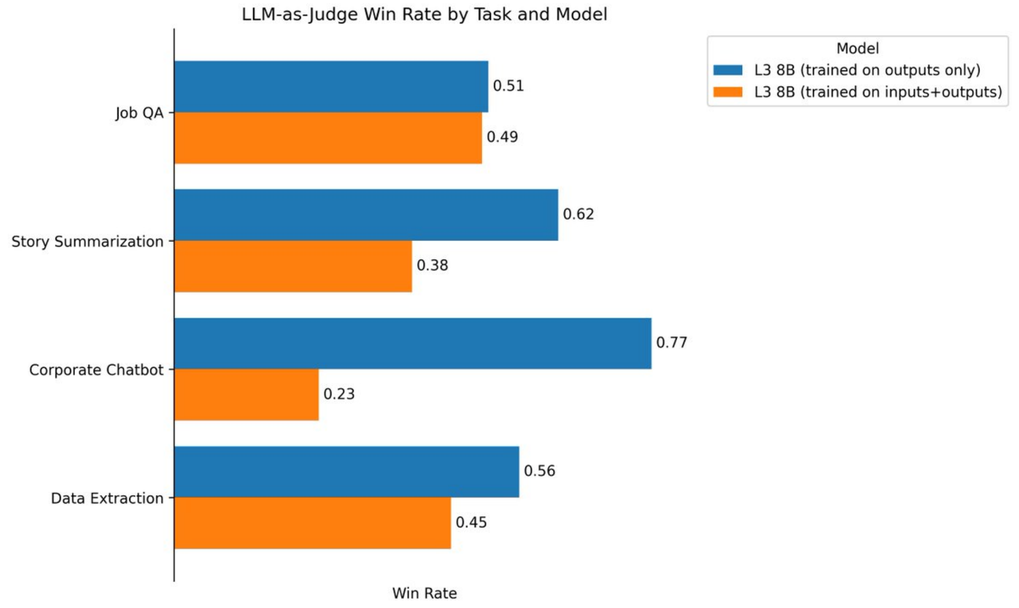
- We're still using decoder-only models
- The instruction itself is masked, so the model does not generate instructions.

In the loss function, mask out the tokens corresponding to prompt

```
Below is an instruction that describes a task. Write a
response that appropriately completes the request.

### Instruction:
Rewrite the following sentence using passive voice.

### Input:
The team achieved great results.

### Response:
Great results were achieved by the team.
```

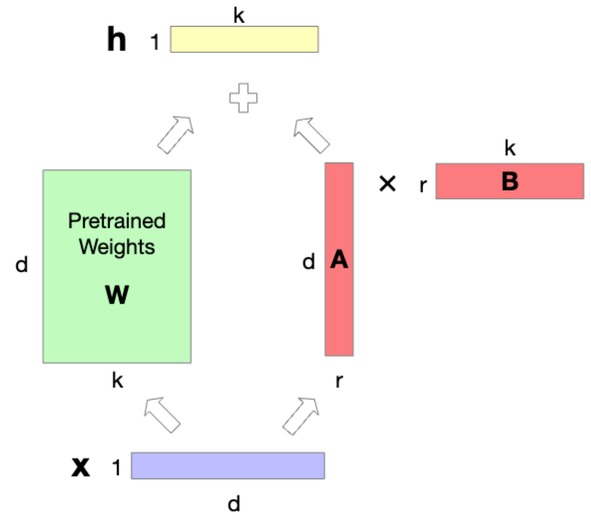
Calculate the loss only over the tokens corresponding to the response text

An illustration of input masking where the highlighted text is still fed to the LLM, but it is not used when calculating the loss during training.



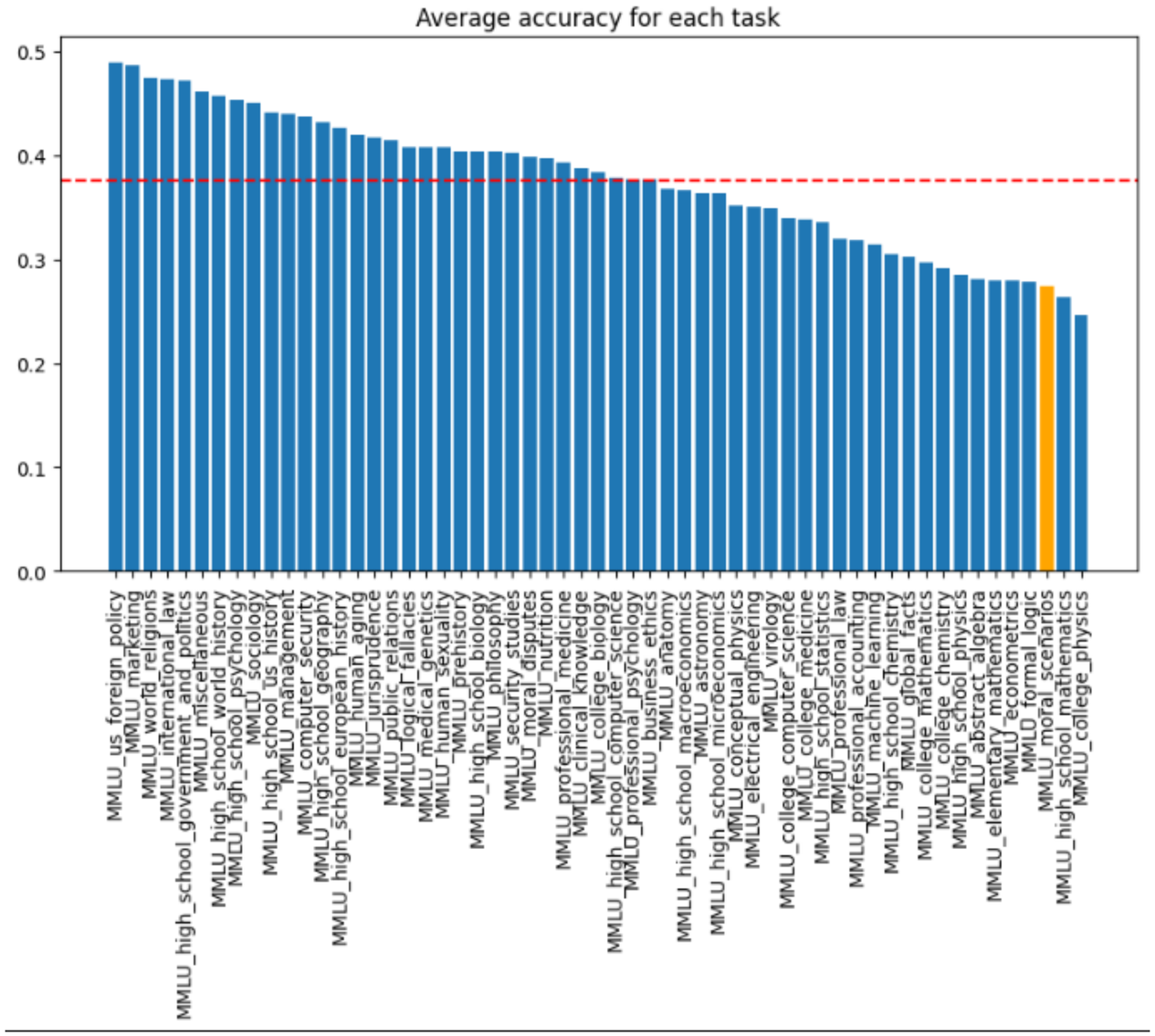
Parameter-Efficient Fine-tuning

- Fine-tuning can be very difficult and expensive with LLMs
  - enormous numbers of parameters to train;
  - each pass of batch gradient descent has to backpropagate through many many huge layers.
- Alternative: allow a model to be finetuned without changing all the parameters.
  - parameter-efficient fine tuning or PEFT,
  - efficiently select a subset of parameters to update when finetuning and keep the rest frozen
- Example: LoRA or Low Rank Adaptation
  - Instead of updating weight matrices in attention layers during finetuning, update a low-rank approximation of the matrix W: matrices A and B which are far smaller
    - We replace $W + \Delta W$ with $W + BA$
    - Gradients don't have to be calculated for most parameters.
  - Weight updates can be simply added in to the pretrained weights, since BA is the same size as W)



Evaluation of LLMs

- Almost exclusively on downstream tasks, as opposed to intrinsic metrics
  - Intrinsic metrics, e.g. perplexity
- Few popular multitask benchmarks
  - GLUE - Language Understanding Tasks
  - SuperGLUE - Language Understanding Tasks
  - HellaSwag - Commonsense Reasoning
  - Truthful QA - Fact Verification
  - MMLU - Massive Multitask Language Understanding, 15908 knowledge and reasoning questions in 57 areas including medicine, mathematics, computer science, law, and others
  - GSM - 8K Grade School Math
  - BigBench - subsumes some of these benchmarks



Lecture Outline

- Announcements
- Recap: Decoding Algorithms
- Evaluating Generated Language
- LLMs
  - Pretraining
  - Post-training with Supervised Finetuning:
    - Instruction Tuning
  - Interacting with LLMs: Prompting
  - Post-training with Alignment with Human Feedback:
    - Preference Tuning: RLHF [Next Class]

Interacting with LLMs: Prompting

Interfacing with Large Language Models

- Once trained, language models can be very powerful
  - The power only increases with scale
  - Most tasks in NLP can be formatted as sequence completion tasks
  - How to interface with a language model to extract relevant information?
- **Prompting (or In-Context / Few-Shot Learning):** the ability to do many tasks with no gradient updates and no / a few examples, by simply:
  - Specifying the right sequence prediction problem
  - You can get interesting zero-shot behavior if you're creative enough with how you specify your task!

Basic Prompt Templates

Summarization	{input} ; `tldr;`
Translation	{input} ; `translate to French:`
Sentiment	{input}; `Overall, it was`
Fine-Grained-Sentiment	{input}; `What aspects were important in this review?`

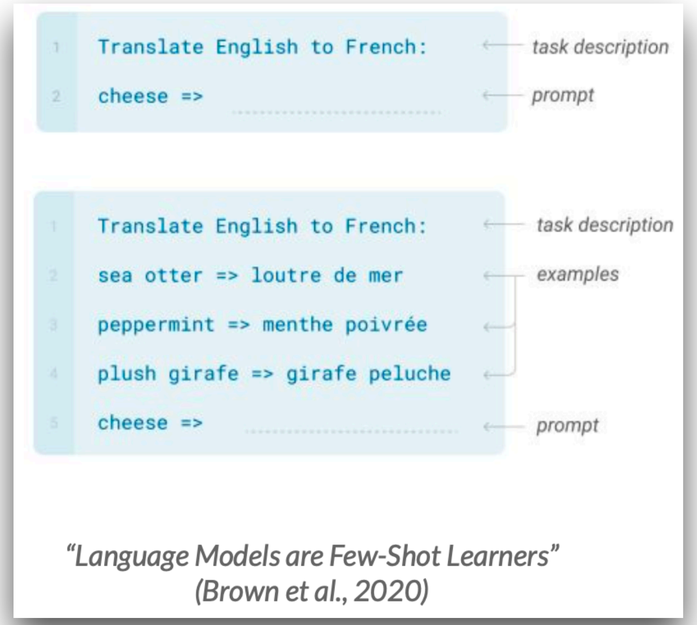
Prompting vs. Instruction Tuning

Translate this to Spanish: Goodbye.

GPT

Adiós.

Input **Model** **Output**



"Language Models are Few-Shot Learners"
(Brown et al., 2020)

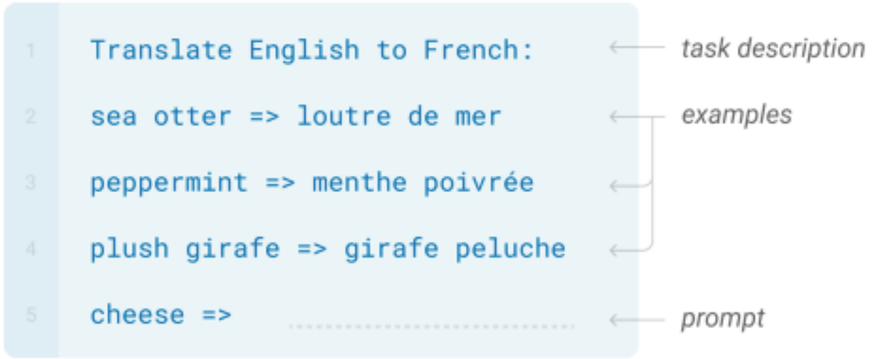
To improve the **output** quality...
We can either change the **input text** or the **model weights**

Prompting **Instruction Tuning**

Prompting

- Interface to a language model: prompts in natural language
- Very large language models seem to perform some kind of "learning" without gradient steps simply from examples you provide within their contexts
- Sometimes called in-context learning
  - Misnomer: no learning (parameter update) actually happens during prompting
- Can be zero shot (without examples) or few-shot (with a few examples)
  - Typically <10

Zero-Shot



Few-Shot

"Language Models are Few-Shot Learners" (Brown et al., 2020)

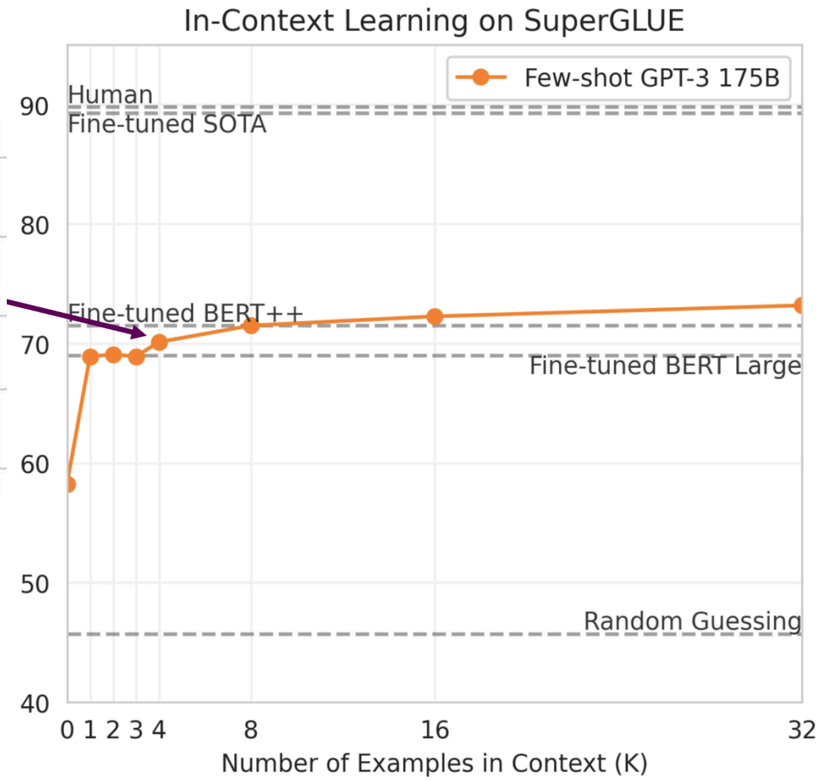
Prompting: Success

- Much more flexible than older formulation of pre-training encoder-only models and fine-tuning to specific classification tasks (the BERT paradigm)
- Now, pre-train one large model and prompt it to do a variety of tasks!
- Much much more generalizability!

Emergent few-shot learning

Few-shot

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



[Brown et al., 2020]

22

Why does prompting work so well?

- Induction heads
- Discovered by looking at mini language models with only 1-2 attention heads
- If the model sees the pattern AB … A in an input sequence, it predicts that B will follow, instantiating the pattern completion rule AB… A→ B
- Perhaps a generalized fuzzy version of this pattern completion rule, implementing a rule like A*B* … A→ B*, where A* ≈ A and B* ≈ B (by ≈ we mean some form of semantically similarity), might be responsible for in-context learning

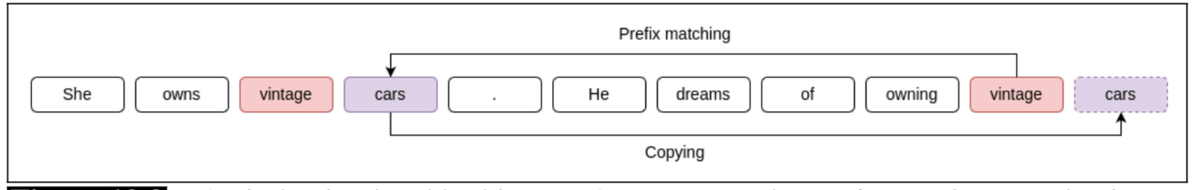
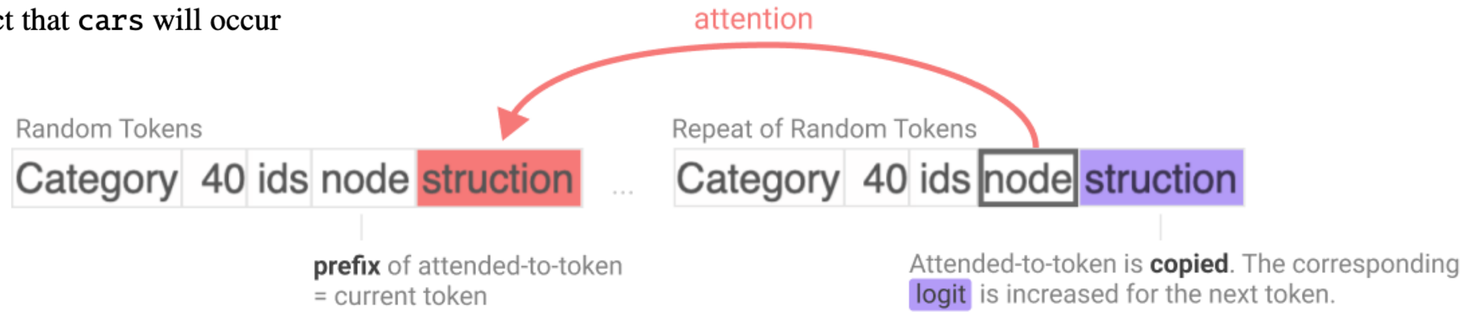


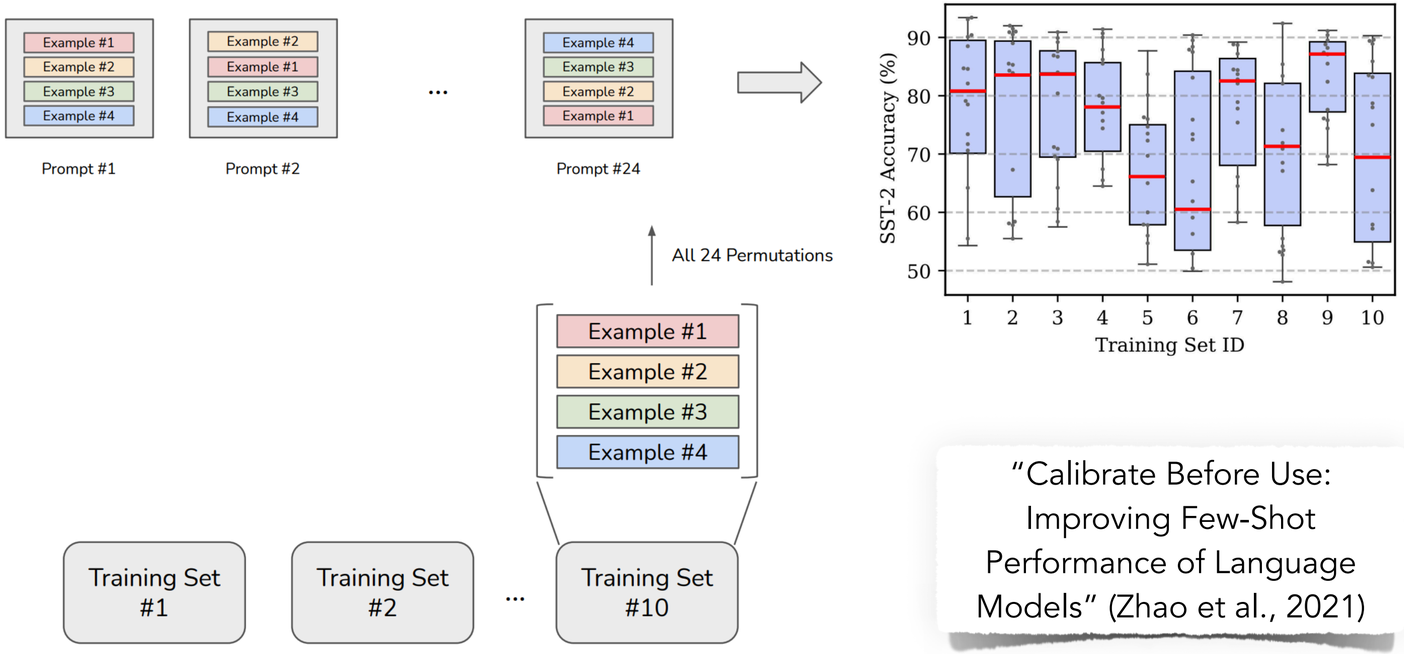
Figure 12.3 An induction head looking at `vintage` uses the *prefix matching* mechanism to find a prior instance of `vintage`, and the *copying* mechanism to predict that `cars` will occur again. Figure from Crosbie and Shutova (2022).



"In-context Learning and Induction Heads" (Olsson et al., 2022)

Prompting Limitations: Prompt Design

- Task performance is sensitive to prompt design
- Formatting
- Ordering of demonstrations
- Wording of the prompt



"Calibrate Before Use: Improving Few-Shot Performance of Language Models" (Zhao et al., 2021)

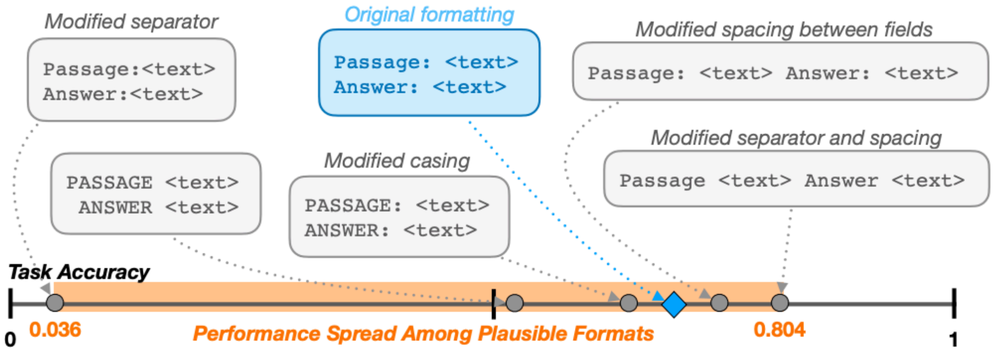
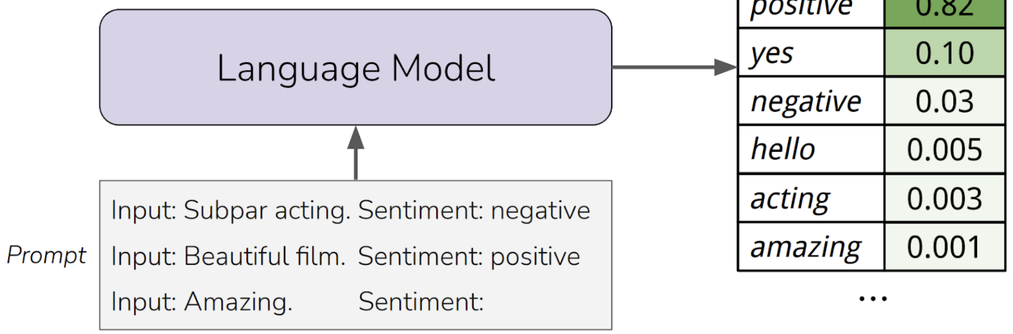


Figure 1: Slight modifications in prompt format templating may lead to significantly different model performance for a given task. Each <text> represents a different variable-length placeholder to be replaced with actual data samples. Example shown corresponds to 1-shot LLaMA-2-7B performances for task280 from SuperNaturalInstructions (Wang et al., 2022). This StereoSet-inspired task Nadeem et al., 2021) requires the model to, given a short passage, classify it into one of four types of stereotype or anti-stereotype (gender, profession, race, and religion).

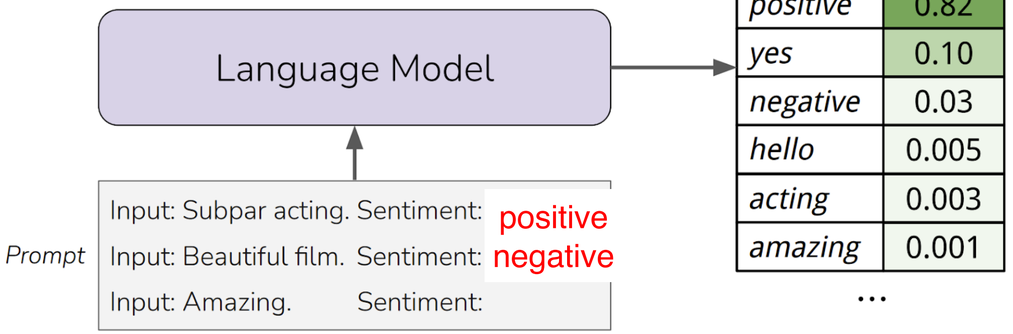
Sclar et al., ICLR 2024

Prompting Limitations: Robustness

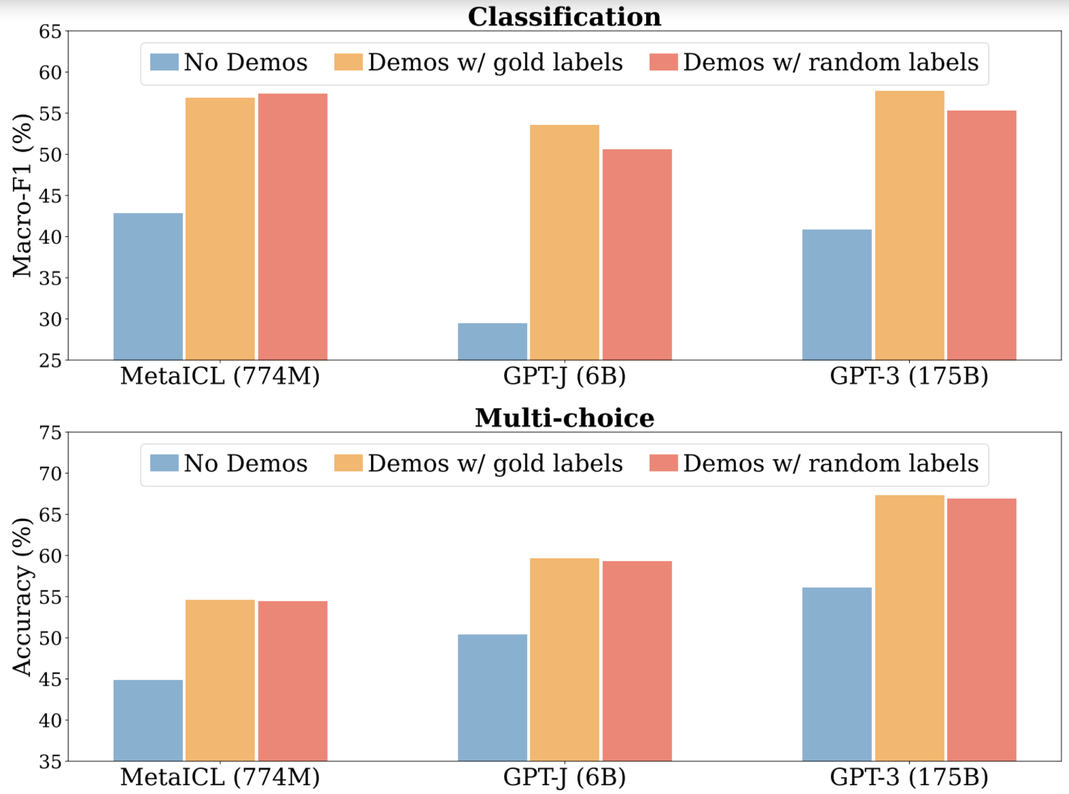
Correct labels



Random (perhaps wrong) labels



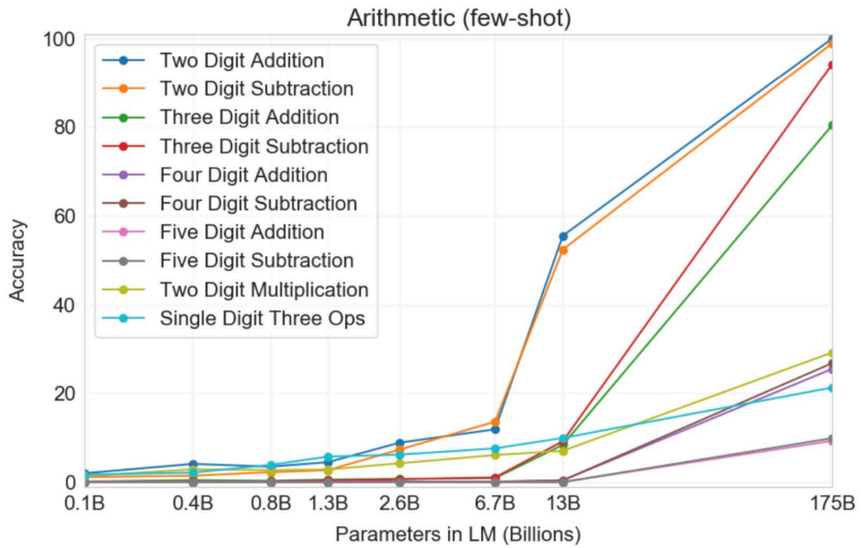
Demonstrations that have incorrect answers can still improve a system!

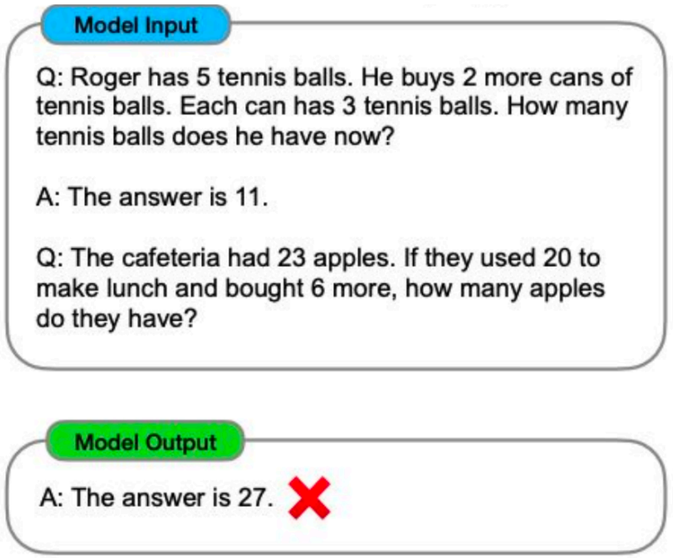


"Rethinking the Role of Demonstrations: What Makes In-Context Learning Work?" (Min et al., 2022)

Prompting Limitations: Math and Reasoning

- Some tasks seem too hard for even large LMs to learn through prompting alone. Especially tasks involving richer, multi-step reasoning. (Humans struggle at these tasks too!)

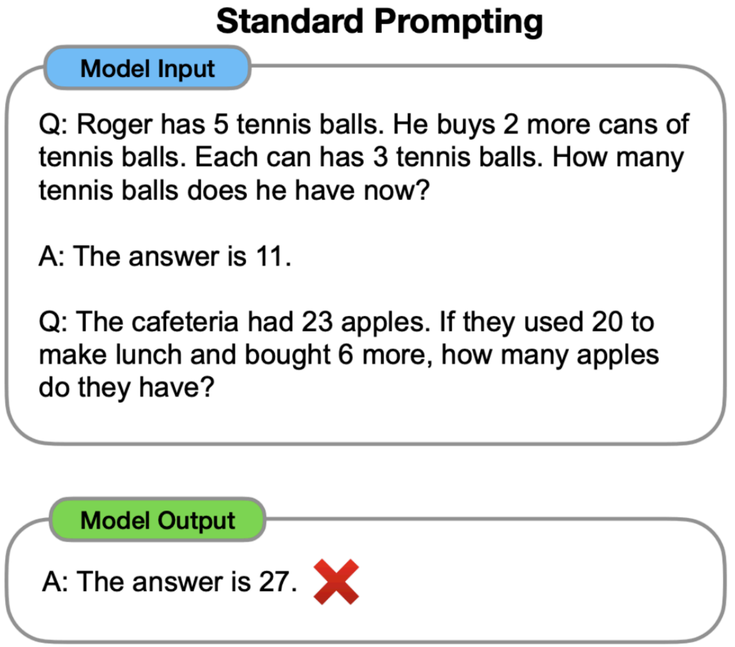





```
19583 + 29534 = 49117
98394 + 49384 = 147778
29382 + 12347 = 41729
93847 + 39299 = ?
```

Chain-of-Thought Prompting

- Since the model is trained on lots and lots of language data, perhaps relying on its capabilities to generate language can make it more accurate!



Zero-Shot Chain-of-Thought Prompting

- The model may not even need examples of reasoning, it may be able to "reason" on its own if provided the right trigger context

Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. 5 + 6 = 11. The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The cafeteria had 23 apples originally. They used 20 to make lunch. So they had 23 - 20 = 3. They bought 6 more apples, so they have 3 + 6 = 9. The answer is 9. ✓

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A: ***Let's think step by step.*** *There are 16 balls in total. Half of the balls are golf balls. That means there are 8 golf balls. Half of the golf balls are blue. That means there are 4 blue golf balls.* ✓

Chain-of-Thoughts Performance

	MultiArith	GSM8K
Zero-Shot	**17.7**	**10.4**
Few-Shot (2 samples)	33.7	15.6
Few-Shot (8 samples)	33.8	15.6
Zero-Shot-CoT	**78.7**	**40.7**
Few-Shot-CoT (2 samples)	84.8	41.3
Few-Shot-CoT (4 samples : First) (*1)	89.2	-
Few-Shot-CoT (4 samples : Second) (*1)	90.5	-
Few-Shot-CoT (8 samples)	93.0	48.7

Greatly outperforms zero-shot →

Manual CoT still better →

Kojima et al., 2022

Zero-shot CoT Trigger Prompt	Accuracy
Let's work this out in a step by step way to be sure we have the right answer.	**82.0**
Let's think step by step. (*1)	78.7
First, (*2)	77.3
Let's think about this logically.	74.5
Let's solve this problem by splitting it into steps. (*3)	72.2
Let's be realistic and think step by step.	70.8
Let's think like a detective step by step.	70.3
Let's think	57.5
Before we dive into the answer,	55.7
The answer is after the proof.	45.7
(Zero-shot)	17.7

There seems to be some wiggle room in the exact prompt to be used for achieving the best performance!

Prompt Engineering and Auto Prompts

WIKIPEDIA
The Free Encyclopedia

Prompt engineering

5 languages

Article Talk More

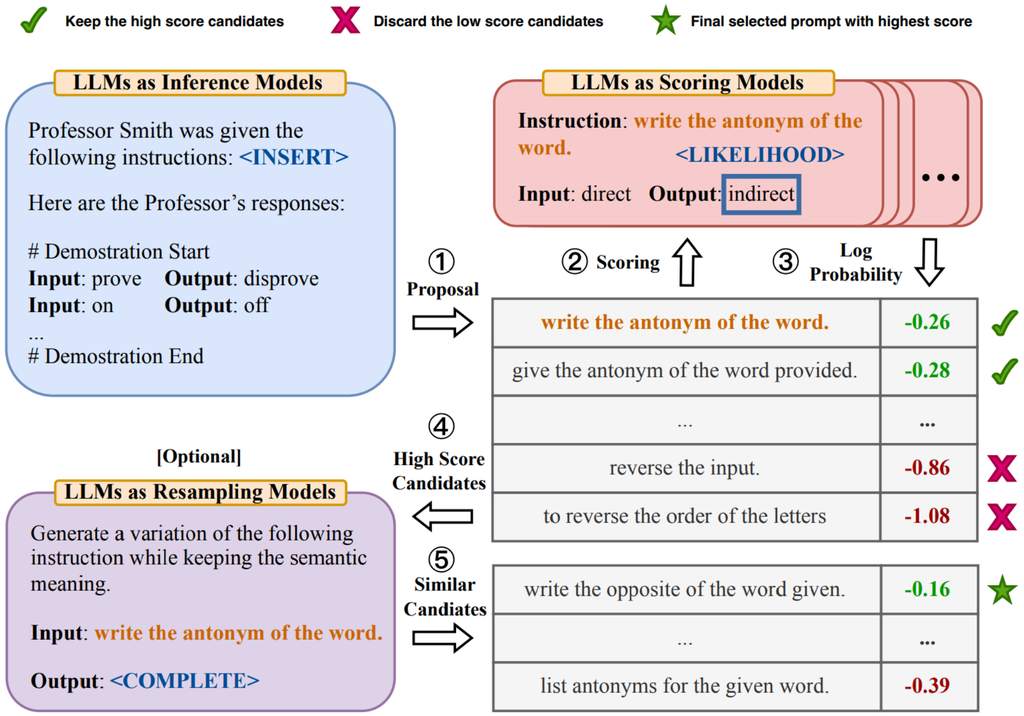
From Wikipedia, the free encyclopedia

Prompt engineering is a concept in artificial intelligence, particularly natural language processing (NLP). In prompt engineering, the description of the task is

Prompt Engineer and Librarian

APPLY FOR THIS JOB

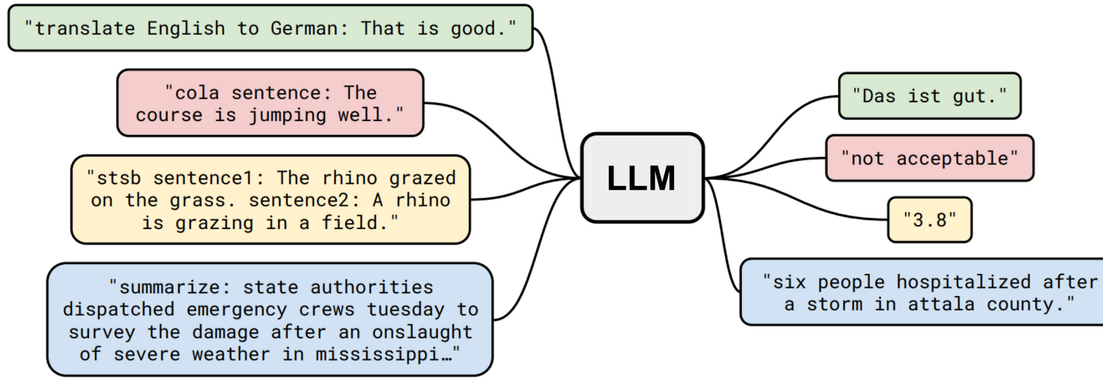
SAN FRANCISCO, CA / PRODUCT / FULL-TIME / HYBRID



Job: keep trying new prompts for better performance, usually via tedious trial-and-error efforts

Automatic Prompt Engineer (APE). LLMs Are Human-Level Prompt Engineers. Zhou et al., ICLR 2023

Prompting LLMs: Parting Thoughts



- Prompting is an interface into language models
- Works best with instruction-tuned language models

- **How Many Demonstrations?** small number of randomly selected labeled examples used as demonstrations can be sufficient
- **How to Select Demonstrations?** Demonstrations are generally created by formatting examples drawn from a labeled training set
  - using demonstrations that are similar to the current input seems to improve performance
  - dynamically retrieve demonstrations for each input, based on their similarity to the current example