Long-context Issues & Introduction to Language Model Alignment

Yu Meng
University of Virginia
yumeng5@virginia.edu

Nov 01, 2024

Reminder

Assignment 4 is due next Monday **(11/04) 11:59pm**!

Overview of Course Contents

- Week 1: Logistics & Overview
- Week 2: N-gram Language Models
- Week 3: Word Senses, Semantics & Classic Word Representations
- Week 4: Word Embeddings
- Week 5: Sequence Modeling and Neural Language Models
- Week 6-7: Language Modeling with Transformers (Pretraining + Fine-tuning)
- Week 8: Large Language Models (LLMs) & In-context Learning
- Week 9-10: Reasoning, Knowledge, and Retrieval-Augmented Generation (RAG)
- Week 11: LLM Alignment
- Week 12: Language Agents
- Week 13: Recap + Future of NLP
- Week 15 (after Thanksgiving): Project Presentations

(Recap) Dense Retrieval

- Motivation: sparse retrieval (e.g., TF-IDF) relies on the exact overlap of words between the query and document without considering semantic similarity
- Solution: use a language model to obtain (dense) distributed representations of query and document
- The retriever language model is typically a small text encoder model (e.g., BERT)
  - Retrieval is a natural language understanding task
  - Encoder-only models are more efficient than LLMs for this purpose
- Both query and document representations are computed by text encoders

(Recap) Dense Retrieval: Cross-encoder

- Process query-document pairs together
- Relevance score produced directly by the model output
- (+) Capture intricate interactions between the query and the document
- (-) Not scalable to large retrieval corpus
- Good for small document sets

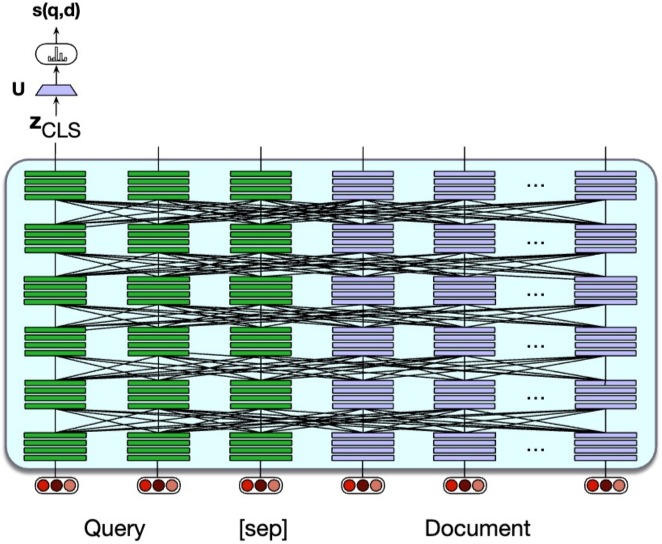


Figure source: https://web.stanford.edu/~jurafsky/slp3/14.pdf

Join at
slido.com
#1564 905

(Recap) Dense Retrieval: Bi-encoder

- Independently encode the query and the document using two separate (but often identical) encoder models
- Use cosine similarity between the query and document vectors as relevance score
- (+) Document vectors can be precomputed
- (-) Cannot capture query-document interactions
- Common choice for large-scale retrieval

$$\cos(\boldsymbol{q}, \boldsymbol{d}) = \frac{\boldsymbol{q} \cdot \boldsymbol{d}}{|\boldsymbol{q}||\boldsymbol{d}|}$$

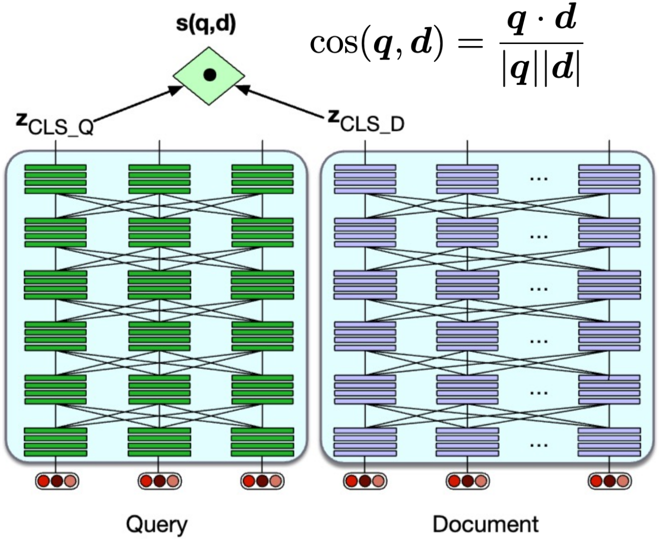


Figure source: https://web.stanford.edu/~jurafsky/slp3/14.pdf

(Recap) Evaluation of IR Systems

- Assume that each document returned by the IR system is either **relevant** to our purposes or **not relevant**
- Given a query, assume the system returns a set of ranked documents $\boldsymbol{T}$
  - A subset $\boldsymbol{R}$ of these are relevant (The remaining $\boldsymbol{N} = \boldsymbol{T} - \boldsymbol{R}$ is irrelevant)
  - There are $\boldsymbol{U}$ documents in the entire retrieval collection that are relevant to this query
- **Precision:** the fraction of the returned documents that are relevant

$$\text{Precision} = \frac{|R|}{|T|}$$

- **Recall:** the fraction of all relevant documents that are returned

$$\text{Recall} = \frac{|R|}{|U|}$$

(Recap) Precision & Recall @ k

- We hope to build a retrieval system that ranks the relevant documents higher
- Use precision & recall @ k (among the top-k items in the ranked list) to reflect this
- Recall @ k is non-decreasing wrt k

Rank	Judgment	Precision_{Rank}	Recall_{Rank}
1	R	1.0	.11
2	N	.50	.11
3	R	.66	.22
4	N	.50	.22
5	R	.60	.33
6	R	.66	.44
7	N	.57	.44
8	R	.63	.55
9	N	.55	.55
10	N	.50	.55

Assume there are 9 total relevant documents in the retrieval corpus

Figure source: https://web.stanford.edu/~jurafsky/slp3/14.pdf

(Recap) Average Precision

Average precision (AP): mean of the precision values at the points in the ranked list where a relevant document is retrieved

Indicator function of whether the document is relevant

$$\text{AP} = \frac{1}{|R|} \sum_{k=1}^{|T|} (\text{Precision@}k \times \underline{\mathbb{1}(d_k \text{ is relevant})})$$

Rank	Judgment	Precision_{Rank}	Recall_{Rank}
1	R	$\boxed{1.0}$	.11
2	N	.50	.11
3	R	$\boxed{.66}$	.22
4	N	.50	.22
5	R	$\boxed{.60}$	.33
6	R	$\boxed{.66}$	.44
7	N	.57	.44
8	R	$\boxed{.63}$	.55
9	N	.55	.55
10	N	.50	.55

(Recap) RAG vs. Direct Prompting

- Prompting relies on LM's parametric knowledge to directly answer the question:

$$P(w|\texttt{Q: Who wrote the book ``The Origin of Species"? A:})$$ prompt

- RAG prepends the set of retrieved passages to the question

Schematic of a RAG Prompt

```
retrieved passage 1

retrieved passage 2

...

retrieved passage n
```

Returned by the retriever

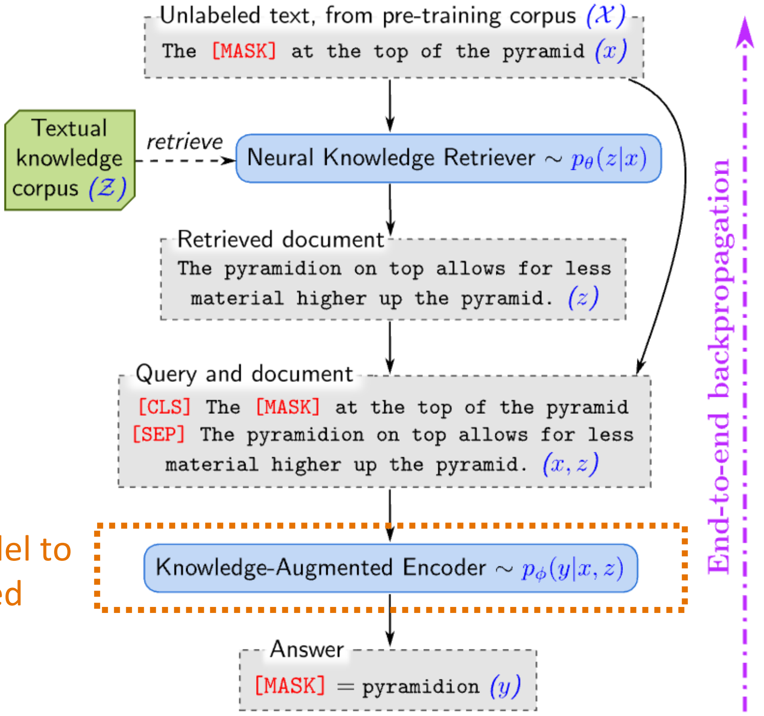
```
Based on these texts, answer this question: Q: Who wrote
the book ``The Origin of Species"? A:
```

Figure source: https://web.stanford.edu/~jurafsky/slp3/14.pdf

(Recap) RAG in Pretraining

- Retrieval-Augmented Language Model pre-training (REALM)
- The first paper that studies incorporating RAG into encoder pretraining (BERT style)
- Main model is a "knowledge-augmented encoder"
- Pretrain with masked language modeling (MLM) loss conditioned on retrieved content

BERT-style model to be pretrained



Paper: https://arxiv.org/pdf/2002.08909

UNIVERSITY of VIRGINIA

Join at
slido.com
#1564 905

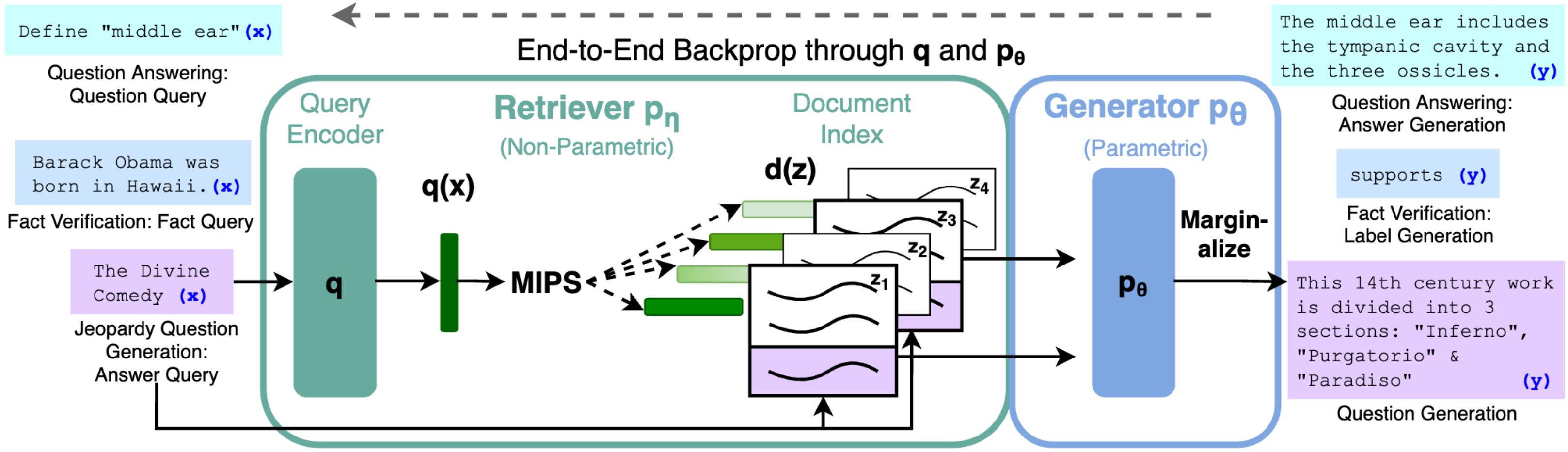
(Recap) RAG: A Latent Variable Model

The retrieved documents are treated as latent variables (z) for generation

$$p(\boldsymbol{y}|\boldsymbol{x}) = \sum_{\boldsymbol{z}\in\mathcal{D}} p(\boldsymbol{z}|\boldsymbol{x})p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{z})$$

Retrieve document (z) based on query (x)

Generate answer (y) based on retrieved docs (z) and query (x)



(Recap) RAG-Sequence Model

- Use the same retrieved document to generate the complete sequence
- Treat the retrieved document as a single latent variable
- Marginalize to get the generation probability $p(\boldsymbol{y}|\boldsymbol{x})$ via a top-K approximation

$$p_{\text{RAG-sequence}}(\boldsymbol{y}|\boldsymbol{x}) \approx \underbrace{\sum_{\boldsymbol{z} \in \text{top-K}(p(\cdot|\boldsymbol{x}))}}_{\text{Top-K approximation (only consider the top-K retrieved docs)}} p_\eta(\boldsymbol{z}|\boldsymbol{x}) p_\theta(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{z}) = \sum_{\boldsymbol{z} \in \text{top-K}(p(\cdot|\boldsymbol{x}))} p_\eta(\boldsymbol{z}|\boldsymbol{x}) \prod_{i=1}^{N} p_\theta(y_i|\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{y}_{<i})$$

Top-K approximation
(only consider the top-K retrieved docs)

The same retrieved doc (z) is used to
generate all tokens in the sequence

(Recap) RAG-Token Model

- Can use different retrieved documents to generate different tokens in a sequence
- Marginalization is performed for each generated token (rather than at sequence level)

$$p_{\text{RAG-token}}(\boldsymbol{y}|\boldsymbol{x}) = \prod_{i=1}^{N} p_\theta(y_i|\boldsymbol{x}, \boldsymbol{y}_{<i}) \approx \prod_{i=1}^{N} \sum_{\boldsymbol{z} \in \text{top-K}(p(\cdot|\boldsymbol{x}, \boldsymbol{y}_{<i}))} p_\eta(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{y}_{<i}) p_\theta(y_i|\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{y}_{<i})$$

Different retrieved doc (z) can be used to generate different tokens in the sequence

RAG-Sequence & RAG-Token Results

Evaluation results on open-domain QA tasks:

- Natural Questions (NQ)
- TriviaQA (TQA)
- WebQuestions (WQ)
- CuratedTrec (CT)

	Model	NQ	TQA	WQ	CT
Closed Book	T5-11B [52]	34.5	- /50.1	37.4	-
	T5-11B+SSM[52]	36.6	- /60.5	44.7	-
Open Book	REALM [20]	40.4	- / -	40.7	46.8
	DPR [26]	41.5	**57.9**/ -	41.1	50.6
	RAG-Token	44.1	55.2/66.1	**45.5**	50.0
	RAG-Seq.	**44.5**	56.8/**68.0**	45.2	**52.2**

Further Reading on RAG

- Generalization through Memorization: Nearest Neighbor Language Models [Khandelwal et al., 2019]
- Active Retrieval Augmented Generation [Jiang et al., 2023]
- Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection [Asai et al., 2023]
- InstructRAG: Instructing Retrieval-Augmented Generation via Self-Synthesized Rationales [Wei et al., 2024]

Agenda

- Long-context Issues
- Introduction to LLM Alignment

RAG & Long Context Issues in LLMs

- RAG significantly increases the input sequence length to LLMs (“**long context**”) by prepending multiple retrieved passages
- **Inefficiency**: the complexity of self-attention is quadratic wrt number of tokens
- **Irrelevant information**: LLMs might get distracted by irrelevant retrieval content
- **Lost in the middle**: LLMs tend to focus more on the beginning and end of the input sequence, but missing important information located in the middle of a long context
- **Performance saturation**: LLMs do not always effectively using the extra context (more retrieved documents)

Join at
slido.com
#1564 905

Lost in the Middle

- Main finding: LLM performance (with RAG) can degrade significantly when changing the position of relevant information
- Performance is often highest when relevant information occurs at the beginning or end of the input context
- Significantly degrades when models must access relevant information in the middle of long contexts
- Analogous to the serial-position effect: a person tends to recall the first and last items in a series best, and the middle items worst

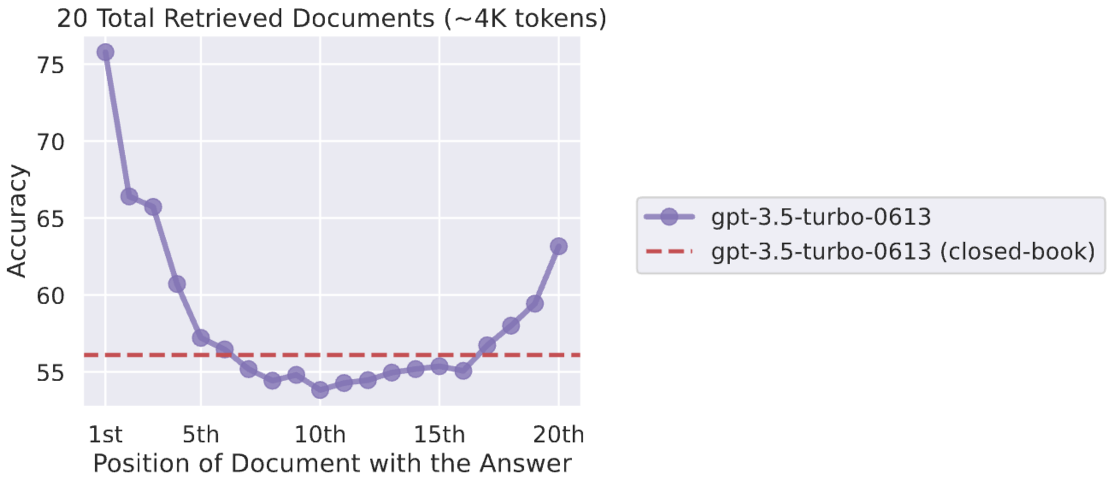
Lost in the Middle: How Language Models Use Long Contexts

Nelson F. Liu[1*] **Kevin Lin**[2] **John Hewitt**[1] **Ashwin Paranjape**[3]
Michele Bevilacqua[3] **Fabio Petroni**[3] **Percy Liang**[1]
[1]Stanford University [2]University of California, Berkeley [3]Samaya AI

Paper: https://arxiv.org/pdf/2307.03172

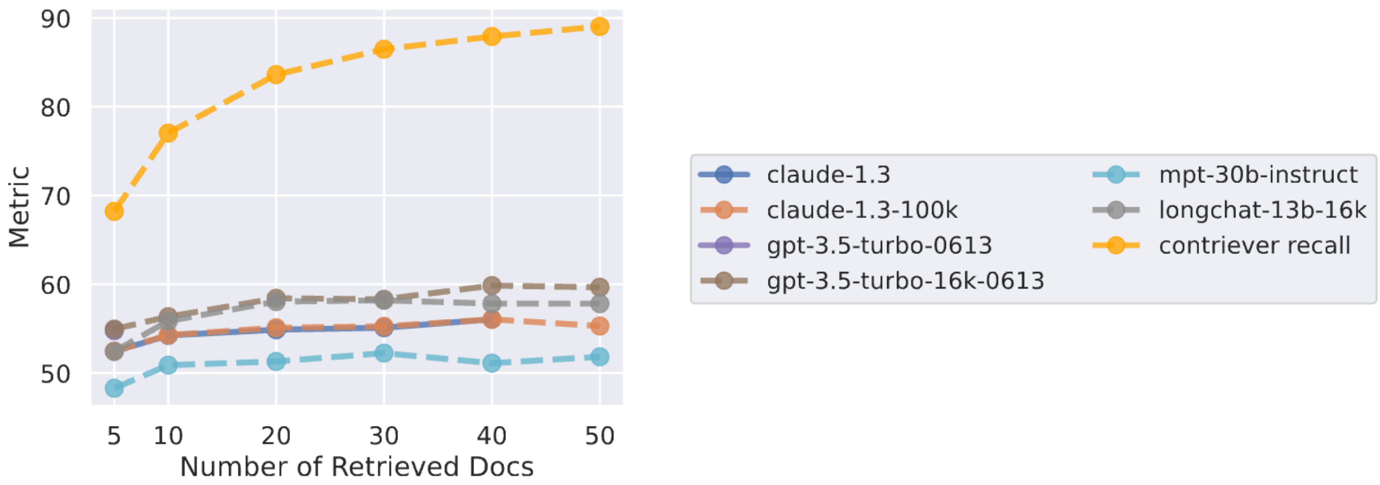
Primacy & Recency Bias

- Exactly one of the documents contains the answer, with other “distractor” documents
- Vary the position of the gold document
- U-shaped performance curve: LLMs are better at using relevant information that occurs at the very beginning (**primacy bias**) or end of its input context (**recency bias**)



Performance Saturation Under More Context

- Retriever recall always improves with more retrieved docs
- LLM performance saturates long before retriever performance saturates (using more than 20 retrieved documents only marginally improves LLM performance)



Practical Considerations of RAG

- Retrieve fewer documents when appropriate
- Avoid inputting irrelevant information to LLMs if possible
- Re-rank retrieved documents to push relevant information closer to the start of the input context
- Adjust & refine retrieval queries based on intermediate results or feedback
- Optimize document chunking: break entire documents into smaller, semantically coherent chunks to ensure only the most relevant parts are input to the LLM

Further Reading on Long-context LLMs

- Efficient Streaming Language Models with Attention Sinks [Xiao et al., 2023]
- LongNet: Scaling Transformers to 1,000,000,000 Tokens [Ding et al., 2023]
- Adapting Language Models to Compress Contexts [Chevalier et al., 2023]
- LLM Maybe LongLM: Self-Extend LLM Context Window Without Tuning [Jin et al., 2024]

Agenda

- Long-context Issues
- Introduction to LLM Alignment

Join at
slido.com
#1564 905

The Evolution of GPT Models: GPT-1

GPT-1: decoder-only Transformer pretraining

Improving Language Understanding by Generative Pre-Training

Alec Radford OpenAI alec@openai.com

Karthik Narasimhan OpenAI karthikn@openai.com

Tim Salimans OpenAI tim@openai.com

Ilya Sutskever OpenAI ilyasu@openai.com

GPT-1

2018

The Evolution of GPT Models: GPT-2

GPT-2: language model pretraining is multi-task learning

Language Models are Unsupervised Multitask Learners

Alec Radford [*1] **Jeffrey Wu** [*1] **Rewon Child** [1] **David Luan** [1] **Dario Amodei** [**1] **Ilya Sutskever** [**1]

GPT-1 GPT-2

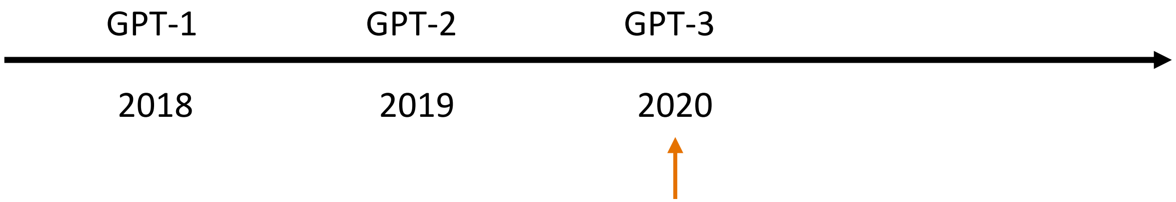
2018 2019

Join at
slido.com
#1564 905

The Evolution of GPT Models: GPT-3

GPT-3: scaling up & in-context learning

Language Models are Few-Shot Learners



UNIVERSITY of VIRGINIA

Join at
slido.com
#1564 905

The Evolution of GPT Models: ChatGPT

ChatGPT: language model alignment

Training language models to follow instructions with human feedback

GPT-1	GPT-2	GPT-3	ChatGPT (GPT-3.5)
2018	2019	2020	2022

UNIVERSITY of VIRGINIA

Join at
slido.com
#1564 905

Overview: Language Model Alignment

- Ensure language models behaviors are aligned with human values and intent
- “HHH” criteria (Askell et al. 2021):
  - **Helpful**: Efficiently perform the task requested by the user
  - **Honest**: Give accurate information & express uncertainty
  - **Harmless**: Avoid offensive/discriminatory/biased outputs

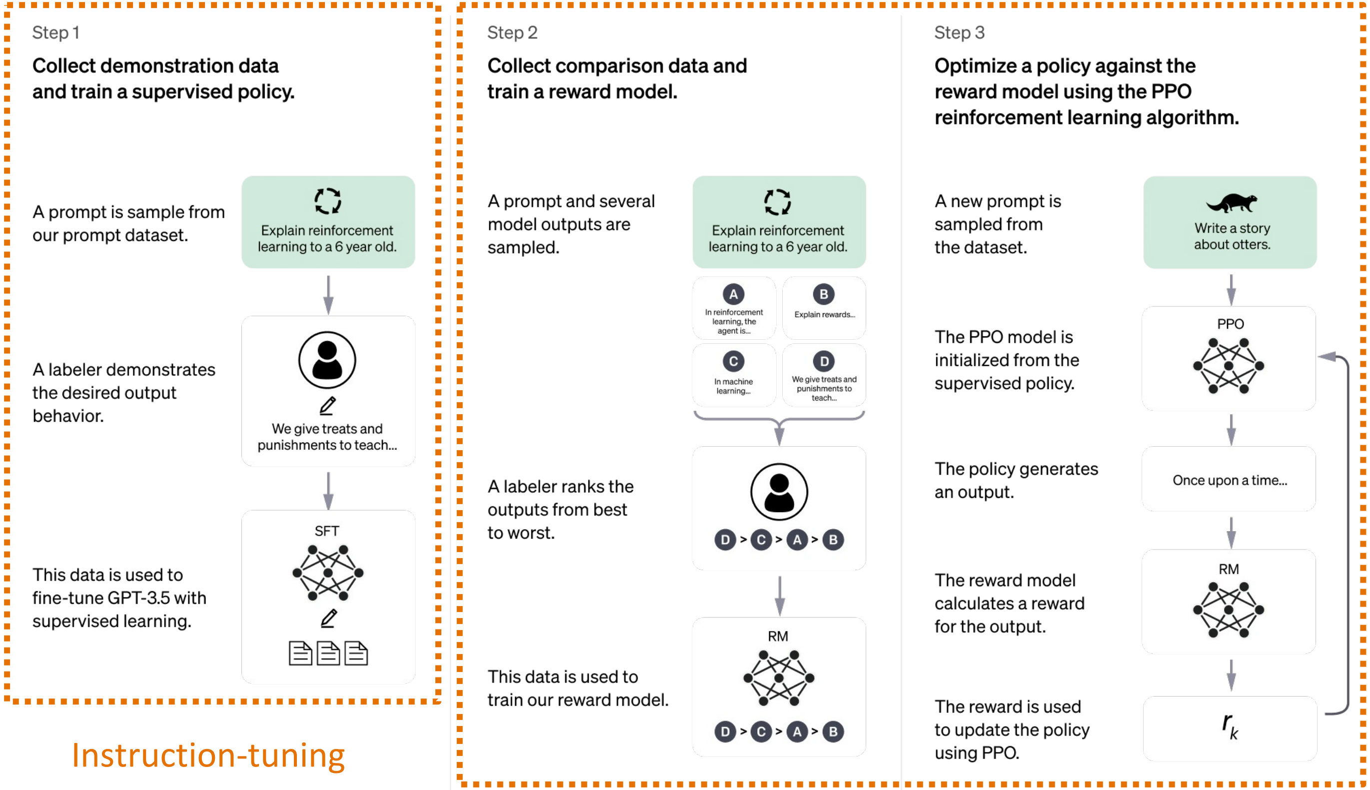
Paper: https://arxiv.org/pdf/2112.00861

Language Model Alignment: Post-training

- Pretrained language models are **not** aligned
- Objective mismatch
  - Pretraining is to predict the next word in a sentence
  - Does not involve understanding human intent/values
- Training data bias
  - Text from the internet can contain biased, harmful, or misleading information
  - LMs don’t distinguish between good and bad behavior in training data
- (Over-)generalization issues
  - LMs’ generalization can lead to outputs that are inappropriate in specific contexts
  - Might not align with intended ethics/honesty standard

Join at
slido.com
#1564 905

Language Model Alignment Techniques



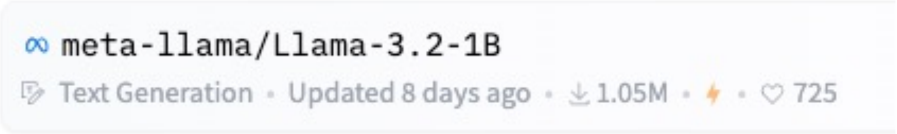
Instruction-tuning

Reinforcement Learning from Human Feedback (RLHF)

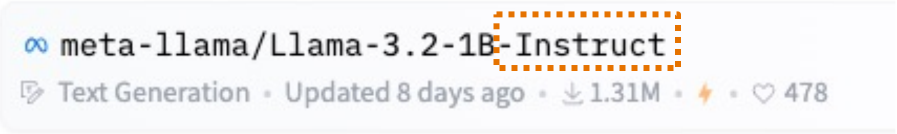
Figure source: https://openai.com/index/chatgpt/

Overview: Instruction-tuning

- Train an LM using a diverse set of tasks
  - Each task is framed as an **instruction** followed by an example of the desired output
  - The goal is to teach the model to follow specific instructions (human intent) effectively
- The resulting model can perform a variety of tasks **zero-shot** (w/o requiring in-context demonstrations)
- The instructions can also be in chat format – tuning an LM into a chatbot



Pretrained (base) model



Instruction-tuned (post-trained) model

Models: https://huggingface.co/meta-llama

Overview: RLHF

- Human feedback collection
  - Generate multiple responses using the model given the same prompt
  - Human evaluators rank responses of the model based on helpfulness/honesty/safety…
- Reward model training
  - A reward model is trained on human feedback data to predict the quality of responses
  - Higher reward = more preferred by human evaluators
- Policy optimization
  - Use reinforcement learning algorithms to further train the LM to maximize the reward predicted by the reward model
  - Encourage the model to produce outputs that align better with human preferences

Summary: Retrieval

- Non-parametric knowledge: (external) information not stored in the model's parameters but can be accessed through retrieval
- Sparse retrieval: based on traditional IR techniques where the representations of documents and queries are sparse vectors (e.g., TF-IDF)
- Dense retrieval: encode documents and queries into dense vectors (embeddings) using encoder LMs (e.g., BERT)
- Evaluation retrieval: Precision/recall @ k, average precision

Summary: Retrieval-Augmented Generation

- RAG can be seen as a latent variable approach (retrieved documents are latent variables in answer generation)
- RAG-sequence uses the same set of retrieved documents to generate the entire sequence
- RAG-token can use different retrieved documents to generate different tokens in the sequence

Summary: Long-context Issues

- RAG increases the LLM input sequence lengths (“long context”) by prepending multiple retrieved passages
- Lost in the middle: performance is higher when relevant information occurs at the beginning or end of the input context, but worse when at middle
- RAG performance does not necessarily improve when more documents are retrieved

Thank You!

Yu Meng
University of Virginia
yumeng5@virginia.edu