Part I: Language Foundation Models for Text Analysis

Automated Mining of Structured Knowledge from Text in the Era of Large Language Models

Yunyi Zhang, Ming Zhong, Siru Ouyang, Yizhu Jiao, Sizhe Zhou, Linyi Ding, Jiawei Han

Computer Science, University of Illinois Urbana-Champaign

KDD 2024 Tutorial, Aug 25, 2024

Tutorial Website:

Large Language Models: Overview

- Pretrained language models: Language models that are trained with deep neural language models (usually Transformer models) via **self-supervised** objectives on **large-scale general-domain corpora**
- Many language models are pretrained and then fine-tuned:
  - Fine-tuning: Adapt the pretrained language models (PLMs) to downstream tasks using task-specific data
- The power of PLMs: Encode generic linguistic features and knowledge learned through large-scale pretraining, which can be effectively transferred to the target applications
- Large language models (LLMs) are PLMs of billions of parameters with astonishing generalization ability to various applications!

Outline

- Pretrained Language Models: Categorization by Architecture 
  - Encoder-Only (Bidirectional) PLM
  - Encoder-Decoder (Sequence-to-Sequence) PLM
  - Decoder-Only (Unidirectional) PLM
- Training Paradigm
- Using and Augmenting LLMs

Categorization of Pretrained Language Models

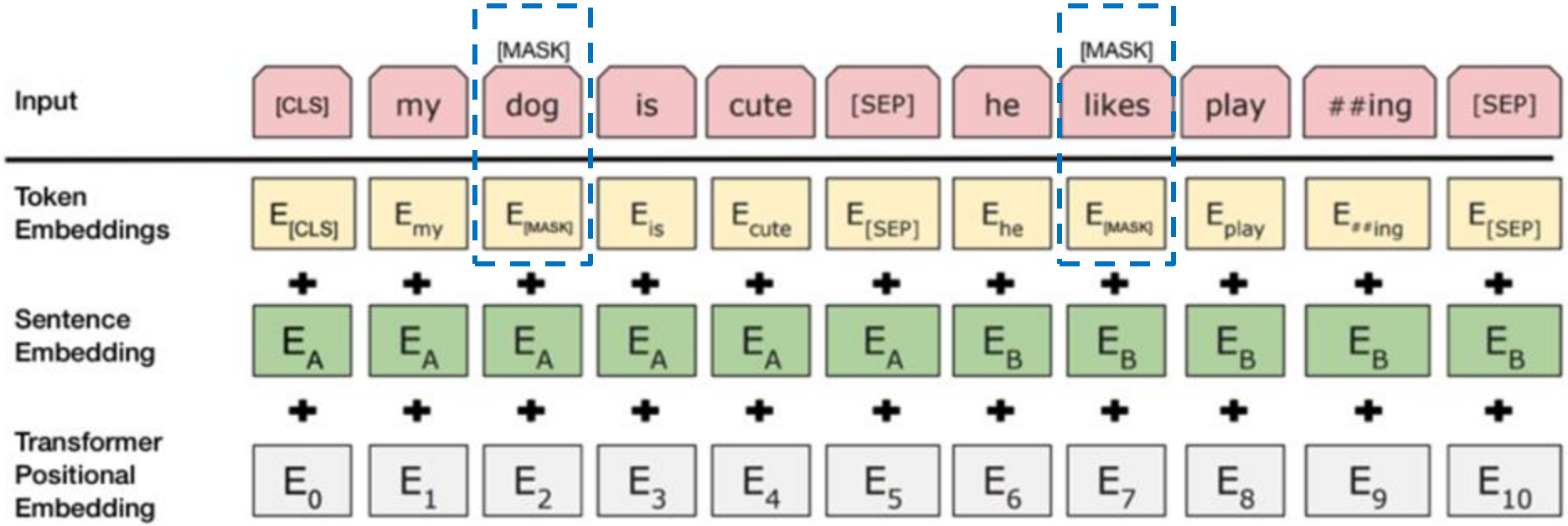
- There are multiple ways to categorize PLMs
  - By pretraining objectives: Standard language modeling, masked language modeling, permuted language modeling…
  - By pretraining settings: Multilingual, knowledge-enriched, domain-specific…
- In this presentation, we categorize PLMs **by architecture** which correlates with the task type PLMs are used for:
  - **Decoder-Only (Unidirectional) PLM**: Predict the next token based on previous tokens, usually used for **language generation tasks** (e.g., GPT, LLaMA)
  - **Encoder-Only (Bidirectional) PLM**: Predict masked/corrupted tokens based on all other (uncorrupted) tokens, usually used for **language understanding/classification tasks** (e.g., BERT, XLNet, ELECTRA)
  - **Encoder-Decoder (Sequence-to-Sequence) PLM**: Generate output sequences given masked/corrupted input sequences, can be used for both **language understanding and generation tasks** (e.g., T5, BART)

Outline

- Pretrained Language Models: Categorization by Architecture
  - Encoder-Only (Bidirectional) PLM 
  - Encoder-Decoder (Sequence-to-Sequence) PLM
  - Decoder-Only (Unidirectional) PLM
- Training Paradigm
- Using and Augmenting LLMs

BERT: Masked Language Modeling

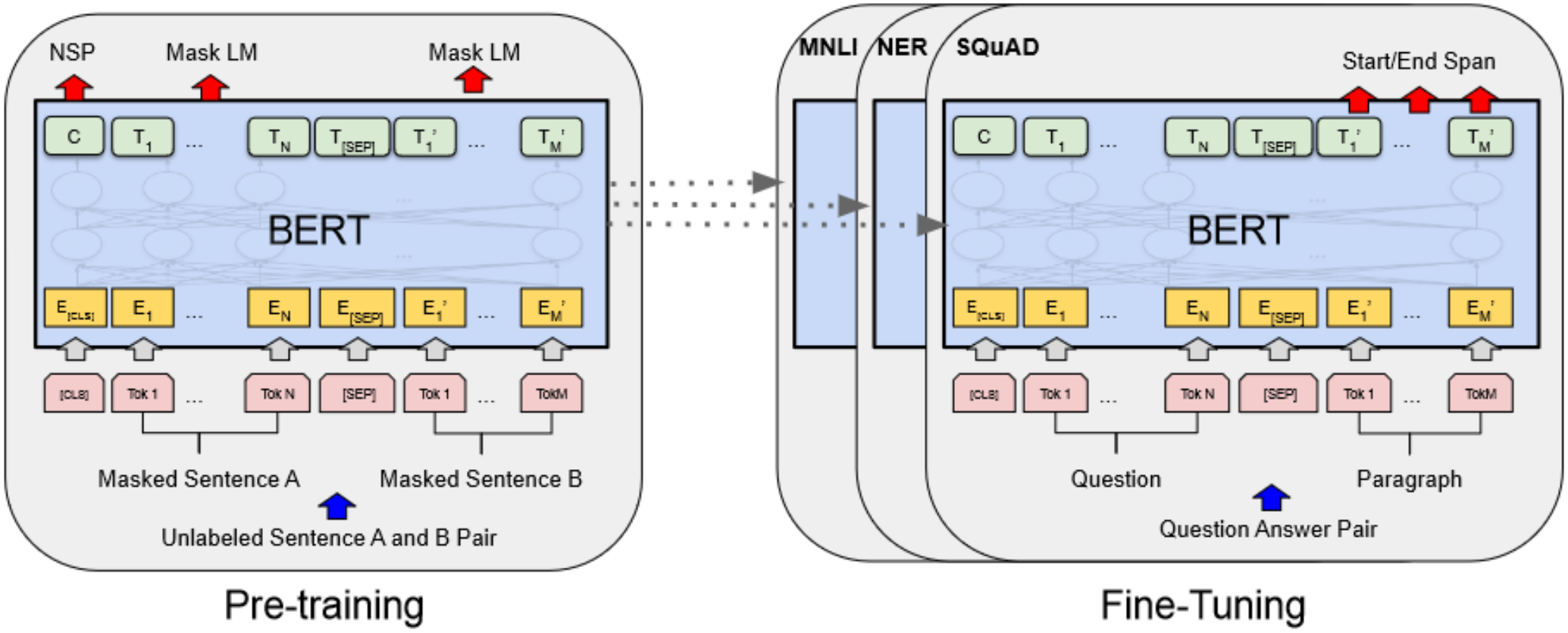
- The Transformer employs an attention mechanism that learns contextual relations between words (and sub-words) in a text sequence
- Masked LM: With 15% words randomly masked, the model learns bidirectional contextual information to predict the masked words



Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." *NAACL* (2019).

BERT: Next Sentence Prediction

- BERT receives pairs of sentences as input and learns to predict if the second sentence in the pair is the subsequent sentence in the original document



Variants of BERT

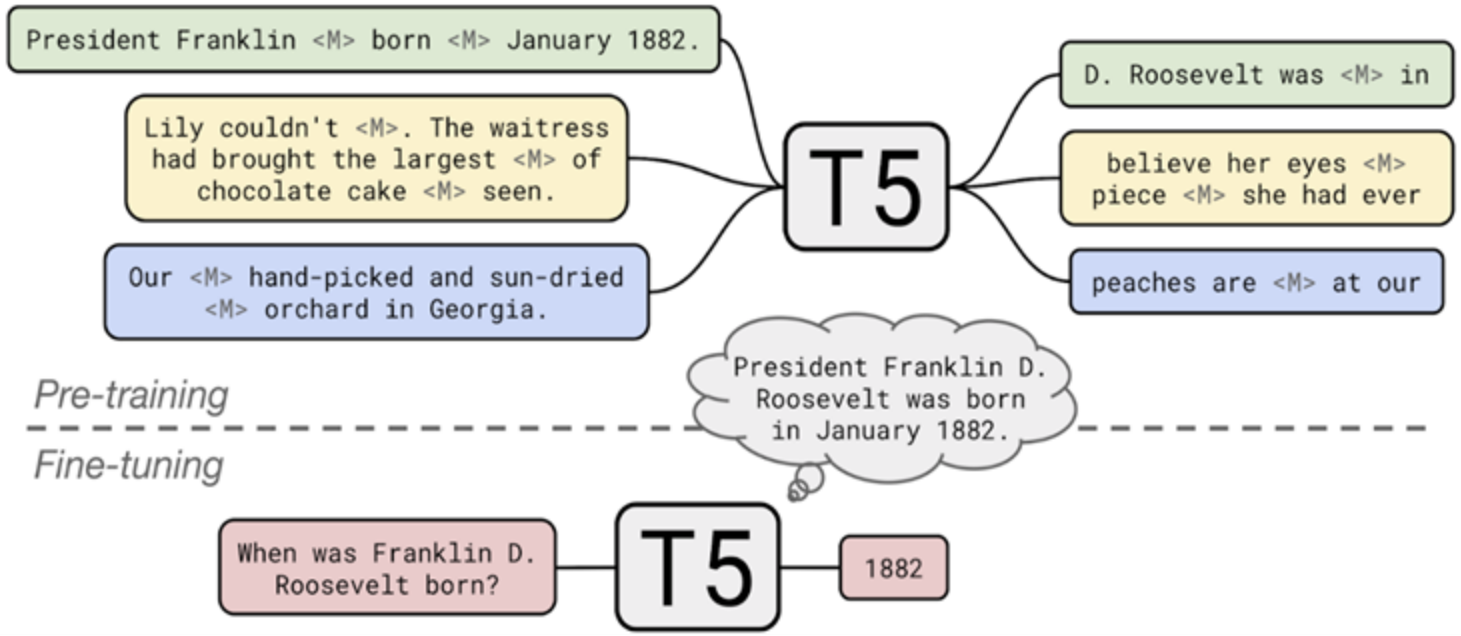
- RoBERTa (Liu et al. 2019): Pretrain BERT on more data for longer, without next sentence prediction
- XLNet (Yang et al. 2019): Permutation language modeling with two-stream self-attention
- ALBERT (Lan et al. 2020): Shared Transformer parameters across layers for parameter efficiency
- ELECTRA (Clark et al. 2020): Replaced token detection by corrupting text sequences with an auxiliary MLM
- DeBERTa (He et al. 2021): Disentangled attention for contents and positions; absolute position incorporated before decoding
- COCO-LM (Meng et al. 2021): Token replacement correction and sequence contrastive learning

Outline

- Pretrained Language Models: Categorization by Architecture
  - Encoder-Only (Bidirectional) PLM
  - Encoder-Decoder (Sequence-to-Sequence) PLM 
  - Decoder-Only (Unidirectional) PLM
- Training Paradigm
- Using and Augmenting LLMs

T5

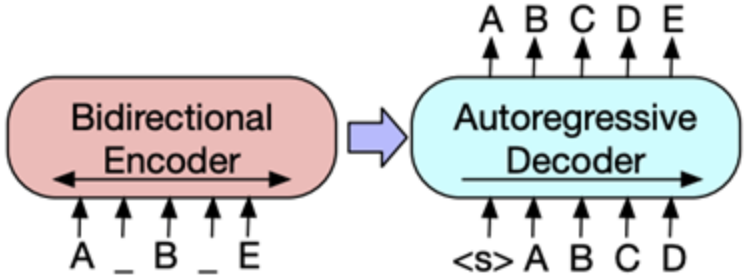
- T5: **T**ext-**t**o-**T**ext **T**ransfer **T**ransformer
- Pretraining: Mask out spans of texts; generate the original spans
- Fine-Tuning: Convert every task into a sequence-to-sequence generation problem



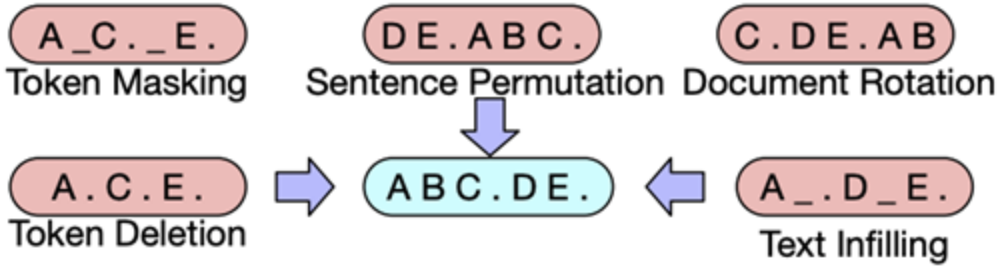
Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., ... & Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. JMLR.

BART

- BART: Denoising autoencoder for pretraining sequence-to-sequence models
- Pretraining: Apply a series of noising schemes (e.g., masks, deletions, permutations…) to input sequences and train the model to recover the original sequences
- Fine-Tuning:
  - For classification tasks: Feed the same input into the encoder and decoder, and use the final decoder token for classification
  - For generation tasks: The encoder takes the input sequence, and the decoder generates outputs autoregressively



BART architecture



BART pretraining objectives

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., ... & Zettlemoyer, L. (2020). BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. ACL.

Outline

- Pretrained Language Models: Categorization by Architecture
  - Encoder-Only (Bidirectional) PLM
  - Encoder-Decoder (Sequence-to-Sequence) PLM
  - Decoder-Only (Unidirectional) PLM 
- Training Paradigm
- Using and Augmenting LLMs

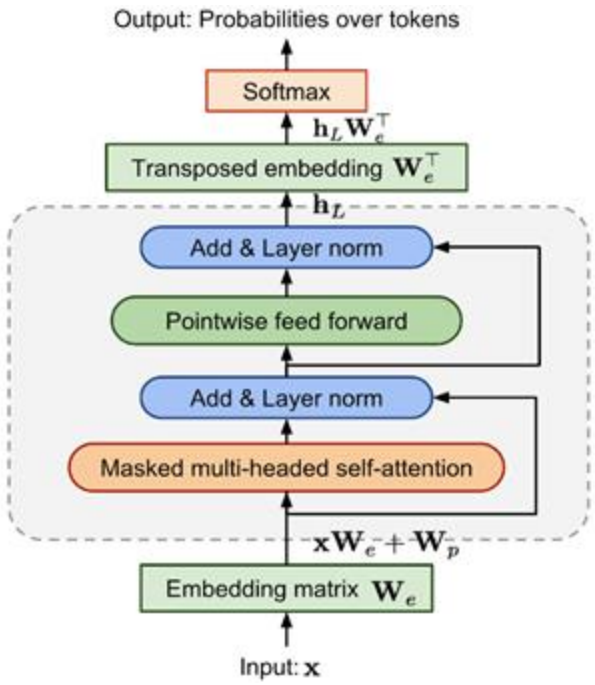
GPT-Style Pretraining: Introduction

- Generative Pretraining (GPTs [1-3]):
- Leverage unidirectional context (usually left-to-right) for next token prediction (i.e., language modeling)

k previous tokens as context

$$\mathcal{L}_{\text{LM}} = -\sum_i \log p(x_i \mid x_{i-k}, \ldots, x_{i-1})$$

- The Transformer uses **unidirectional** attention masks (i.e every token can only attend to previous tokens)



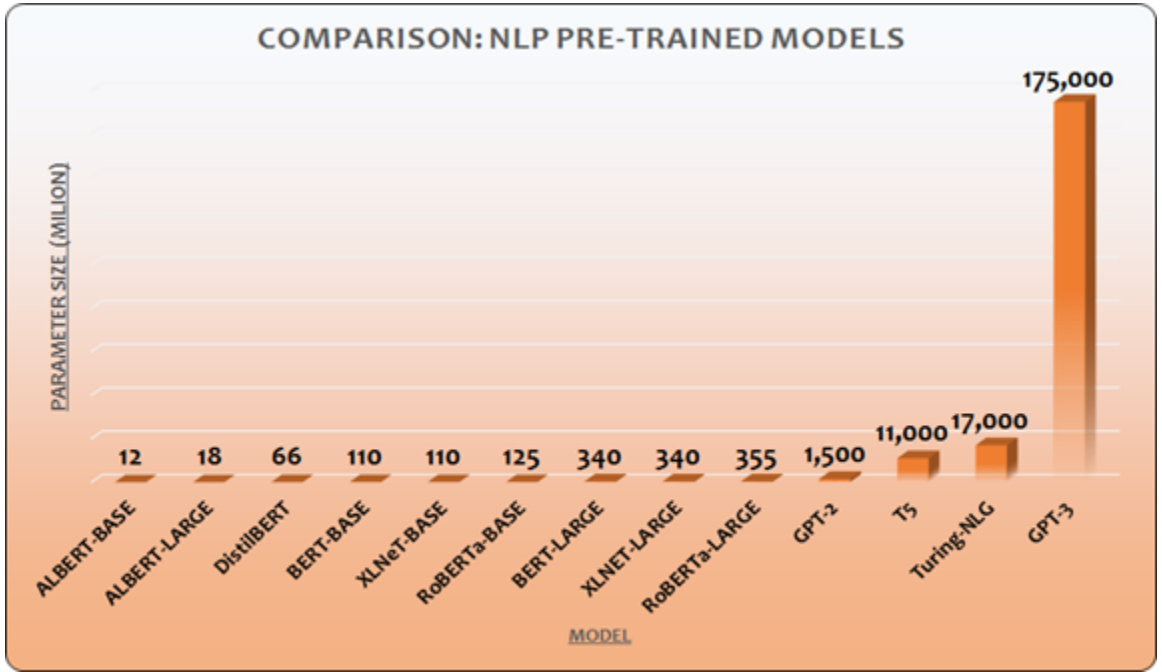
[1] Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving language understanding by generative pre-training. OpenAI blog

[2] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multitask learners. OpenAI blog, 1(8), 9.

[3] Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ... & Amodei, D. (2020). Language models are few-shot learners. NeurIPS.

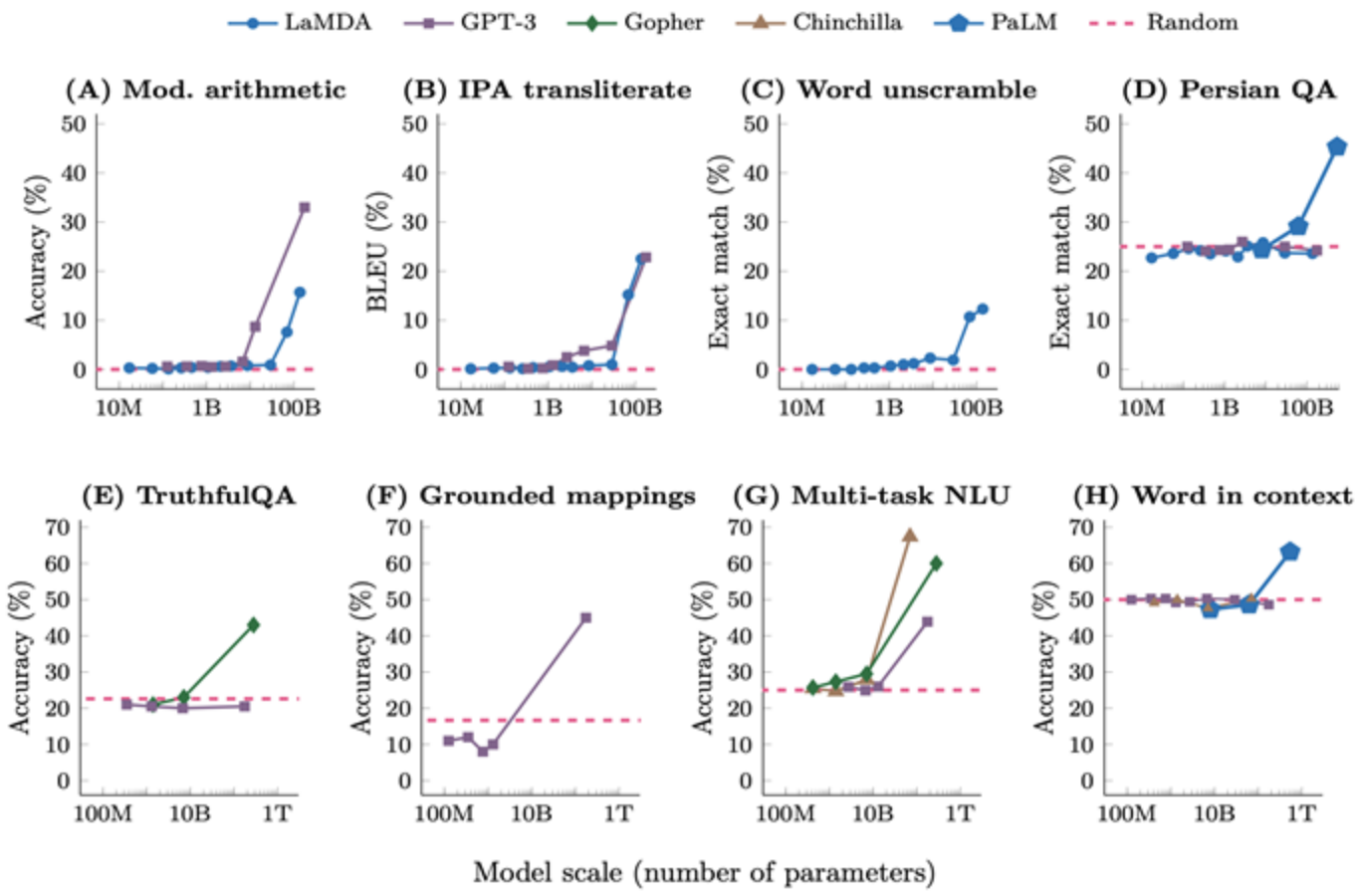
GPT-Style Pretraining: Text Generation

- Unidirectional LMs are commonly used for autoregressive **text generation tasks** (e.g., summarization, translation, …)
- A lot of downstream tasks can be converted into text generation tasks (e.g., letting the model generate the sequence label)!
- They can be very, very large (GPT-3 has 175 billion parameters; GPT-4 may have much more!) and have very strong text generation abilities



Why Large Language Models (LLMs)?

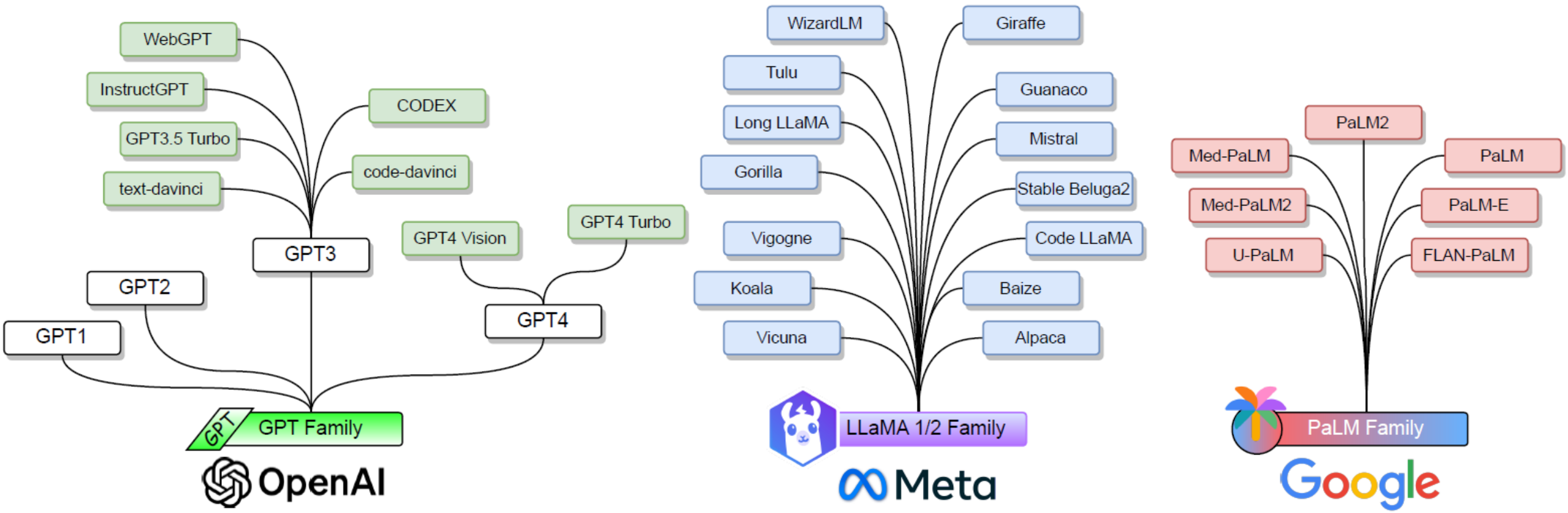
- Scaling up language models induces **emergent abilities**
- "Emergent": not present in smaller models but in larger models



Emergent ability for few-shot prompting: LMs have random performance until a certain scale, after which performance significantly increases well-above random

Wei, J., et al. (2022). Emergent Abilities of Large Language Models. TMLR.

Popular LLM Families



Courtesy of *Shervin Minaee, Tomas Mikolov, Narjes Nikzad, Meysam Chenaghlu, Richard Socher, Xavier Amatriain, Jianfeng Gao, "Large Language Models: A Survey", https://arxiv.org/abs/2402.06196*

LLMs: Pretrained vs. Instruction-Tuned Models

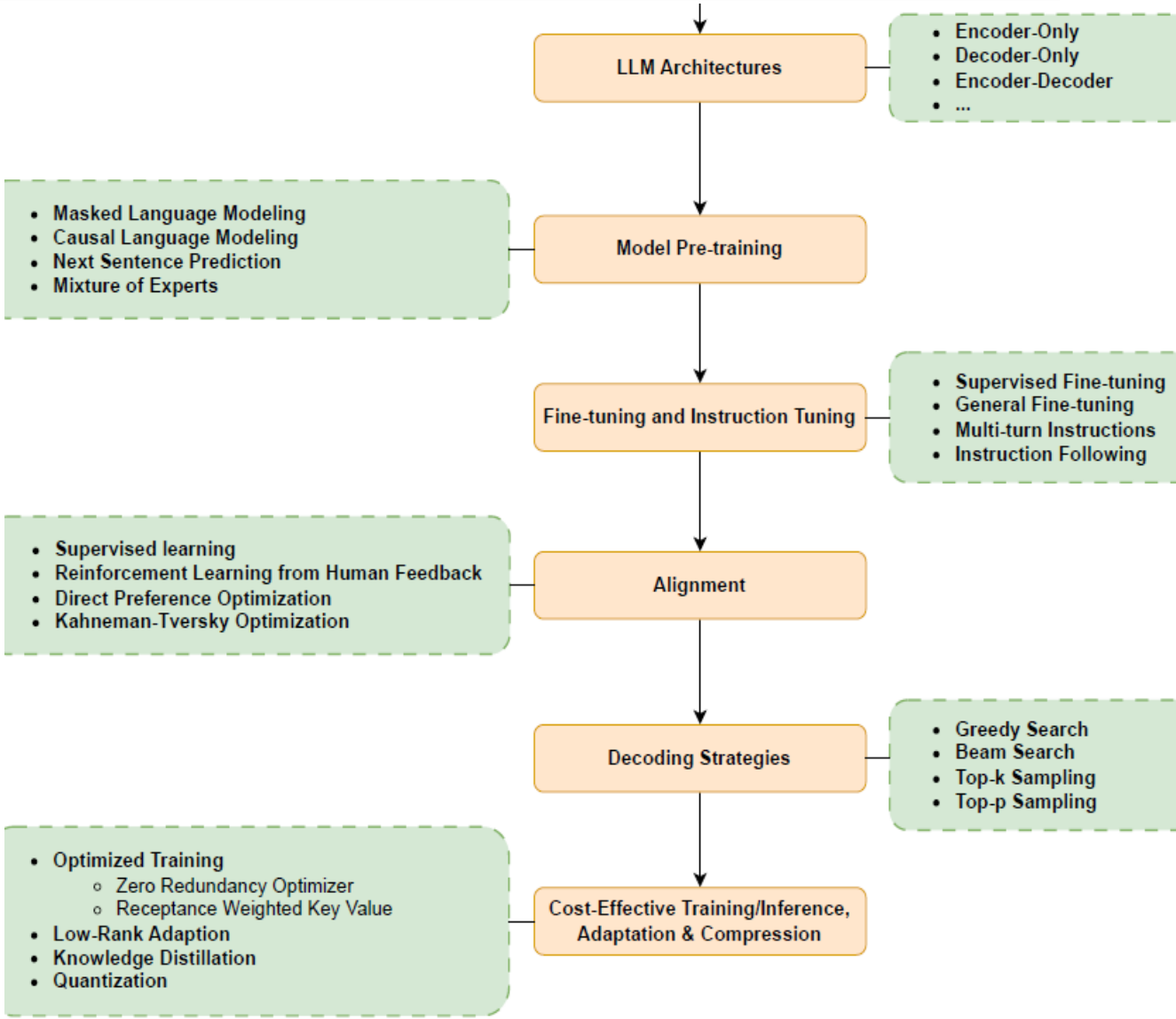
- Pretrained models
  - PaLM (Chowdhery et al. 2022): 8B/62B/540B
  - OPT (Zhang et al. 2022): up to 175B
  - LLaMA (Touvron et al. 2023a): 7B/13B/33B/65B
- Instruction-tuned models
  - Bard (Google 2023)
  - LLaMA 2 (Touvron et al. 2023b): 7B/13B/34B/70B
  - Stanford Alpaca (Taori et al.): tuned based on LLaMA
  - LLaMA 3 (2024): 8B/70B/405B
- More LLMs can be found on the Chatbot Arena leaderboard

Outline

- Pretrained Language Models: Categorization by Architecture
- Training Paradigm 
  - Supervised Fine-Tuning (SFT)
  - Parameter-Efficient Fine-Tuning (PEFT)
  - Reinforcement Learning from Human Feedback (RLHF)
- Using and Augmenting LLMs

Building LLMs: Language Model Training Processes

- Three stages of LM training: **Pretraining, Fine-tuning** and Human **Alignment**
- **Pretraining**: Initial pretraining on large-scale general domain corpora to learn generic linguistic features to grasp language basic
  - Models are exposed to vast amounts of (usually unlabeled) text data, learning to predict accurately in various contexts without direct supervision (i.e., in a self-supervised manner)
  - Approaches: Next token prediction (autoregressive language modeling), Masked language modeling, next sentence prediction
- Supervised **Fine-tuning** for specific applications or desired output styles
- Human **alignment** with human preference and ethical standards



Adapted from: S. Minaee, et al., , "Large Language Models: A Survey", arxiv: 2402.06196

Deployment of Pretrained Language Models

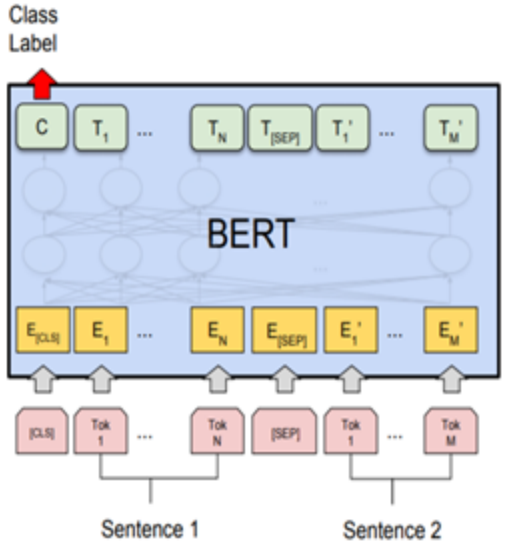
- Pretrained language models (PLMs) are usually trained on large-scale general domain corpora to learn generic linguistic features that can be transferred to downstream tasks
- Common usages of PLMs in downstream tasks
  - Fine-tuning: Update all parameters in the PLM encoder and task-specific layers (linear layer for standard fine-tuning or MLM layer for prompt-based fine-tuning) to fit downstream data
  - Prompt-based methods: Convert tasks to cloze-type token prediction problems; can be used for either fine-tuning or zero-shot inference
  - Parameter-efficient tuning: Only update a small portion of PLM parameters and keep other (majority) parameters unchanged

Outline

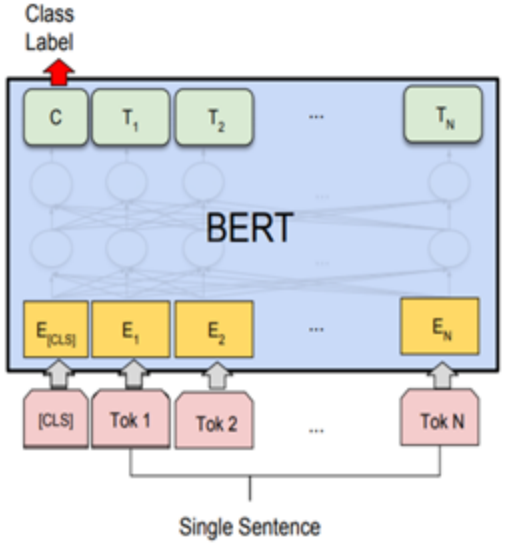
- Pretrained Language Models: Categorization by Architecture
- Training Paradigm
  - Supervised Fine-Tuning (SFT) 
  - Parameter-Efficient Fine-Tuning (PEFT)
  - Reinforcement Learning from Human Feedback (RLHF)
- Using and Augmenting LLMs

Supervised Fine-Tuning of PLMs

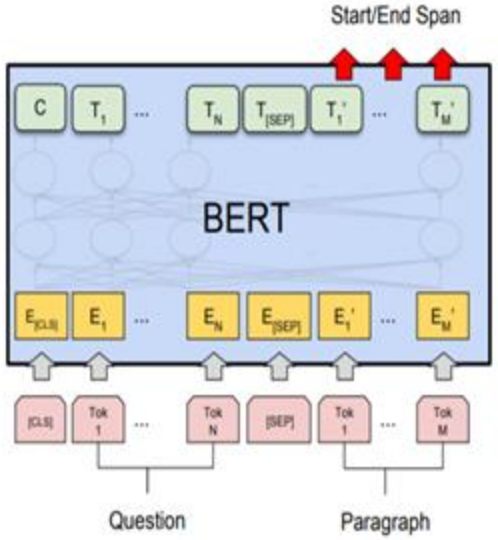
- Add task-specific layers (usually one or two linear layers) on top of the embeddings produced by the PLMs (sequence-level tasks use [CLS] token embeddings; token-level tasks use real token embeddings)
- Task-specific layers and the PLMs are jointly fine-tuned with task-specific training data



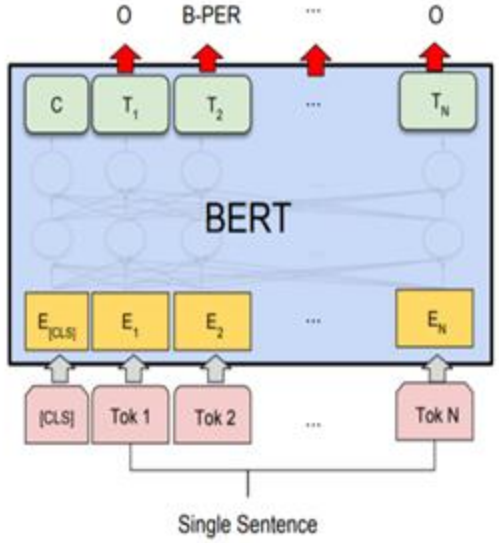
(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG



(b) Single Sentence Classification Tasks: SST-2, CoLA



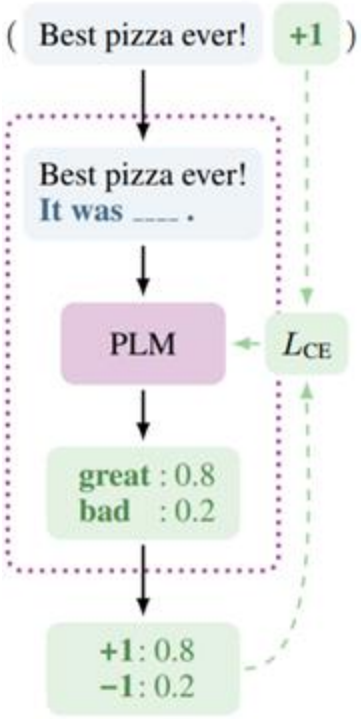
(c) Question Answering Tasks: SQuAD v1.1

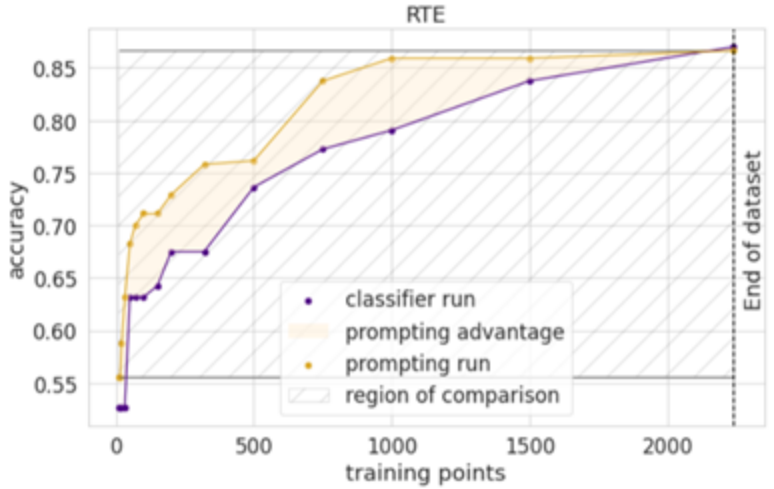


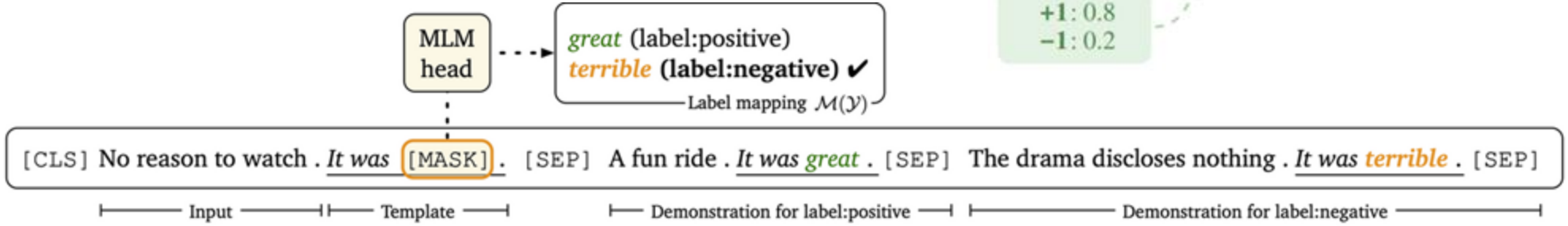
(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

Prompt-Based Fine-Tuning: Introduction

- Task descriptions are created to convert training examples to cloze questions
- Highly resemble the pretraining tasks (MLM) so that pretraining knowledge could be better leveraged
- Better than standard fine-tuning especially for few-shot settings







Schick, T., & Schütze, H. (2021). Exploiting cloze questions for few shot text classification and natural language inference. EACL.

Le Scao, T., & Rush, A. M. (2021). How many data points is a prompt worth? NAACL.

Prompt-Based Fine-Tuning: Results

- Further improve prompt-based few-shot fine-tuning:
  - Prompt templates and label words can be automatically generated
  - Demonstrations can be concatenated with target sequences to provide hints

	SST-2 (acc)	SST-5 (acc)	MR (acc)	CR (acc)	MPQA (acc)	Subj (acc)	TREC (acc)	CoLA (Matt.)
Majority†	*50.9*	*23.1*	*50.0*	*50.0*	*50.0*	*50.0*	*18.8*	*0.0*
Prompt-based zero-shot‡	83.6	35.0	80.8	79.5	67.6	51.4	32.0	2.0
"GPT-3" in-context learning	84.8 (1.3)	30.6 (0.9)	80.5 (1.7)	87.4 (0.8)	63.8 (2.1)	53.6 (1.0)	26.2 (2.4)	-1.5 (2.4)
Fine-tuning	81.4 (3.8)	43.9 (2.0)	76.9 (5.9)	75.8 (3.2)	72.0 (3.8)	90.8 (1.8)	88.8 (2.1)	**33.9** (14.3)
Prompt-based FT (man)	92.7 (0.9)	47.4 (2.5)	87.0 (1.2)	90.3 (1.0)	84.7 (2.2)	91.2 (1.1)	84.8 (5.1)	9.3 (7.3)
+ demonstrations	92.6 (0.5)	**50.6** (1.4)	86.6 (2.2)	90.2 (1.2)	**87.0** (1.1)	**92.3** (0.8)	87.5 (3.2)	18.7 (8.8)
Prompt-based FT (auto)	92.3 (1.0)	49.2 (1.6)	85.5 (2.8)	89.0 (1.4)	85.8 (1.9)	91.2 (1.1)	88.2 (2.0)	14.0 (14.1)
+ demonstrations	**93.0** (0.6)	49.5 (1.7)	**87.7** (1.4)	**91.0** (0.9)	86.5 (2.6)	91.4 (1.8)	**89.4** (1.7)	21.8 (15.9)
Fine-tuning (full)†	*95.0*	*58.7*	*90.8*	*89.4*	*87.8*	*97.0*	*97.4*	*62.6*

	MNLI (acc)	MNLI-mm (acc)	SNLI (acc)	QNLI (acc)	RTE (acc)	MRPC (F1)	QQP (F1)	STS-B (Pear.)
Majority†	*32.7*	*33.0*	*33.8*	*49.5*	*52.7*	*81.2*	*0.0*	-
Prompt-based zero-shot‡	50.8	51.7	49.5	50.8	51.3	61.9	49.7	-3.2
"GPT-3" in-context learning	52.0 (0.7)	53.4 (0.6)	47.1 (0.6)	53.8 (0.4)	60.4 (1.4)	45.7 (6.0)	36.1 (5.2)	14.3 (2.8)
Fine-tuning	45.8 (6.4)	47.8 (6.8)	48.4 (4.8)	60.2 (6.5)	54.4 (3.9)	76.6 (2.5)	60.7 (4.3)	53.5 (8.5)
Prompt-based FT (man)	68.3 (2.3)	70.5 (1.9)	77.2 (3.7)	64.5 (4.2)	69.1 (3.6)	74.5 (5.3)	65.5 (5.3)	71.0 (7.0)
+ demonstrations	**70.7** (1.3)	**72.0** (1.2)	**79.7** (1.5)	**69.2** (1.9)	68.7 (2.3)	77.8 (2.0)	**69.8** (1.8)	73.5 (5.1)
Prompt-based FT (auto)	68.3 (2.5)	70.1 (2.6)	77.1 (2.1)	68.3 (7.4)	**73.9** (2.2)	76.2 (2.3)	67.0 (3.0)	75.0 (3.3)
+ demonstrations	70.0 (3.6)	**72.0** (3.1)	77.5 (3.5)	68.5 (5.4)	71.1 (5.3)	**78.1** (3.4)	67.7 (5.8)	**76.4** (6.2)
Fine-tuning (full)†	*89.8*	*89.5*	*92.6*	*93.3*	*80.9*	*91.4*	*81.7*	*91.9*

Gao, T., Fisch, A., & Chen, D. (2021). Making pre-trained language models better few-shot learners. ACL

Outline

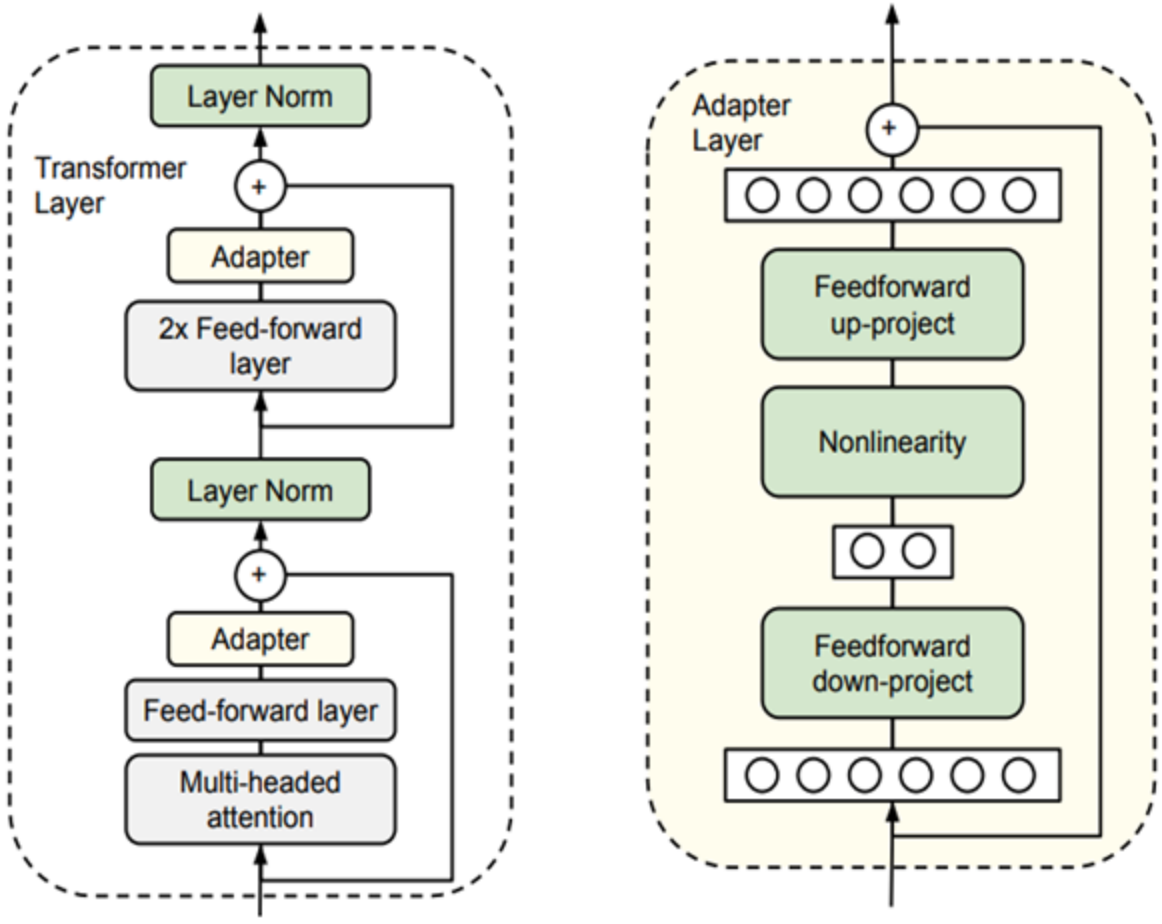
- Pretrained Language Models: Categorization by Architecture
- Training Paradigm
  - Supervised Fine-Tuning (SFT)
  - Parameter-Efficient Fine-Tuning (PEFT) 
  - Reinforcement Learning from Human Feedback (RLHF)
- Using and Augmenting LLMs

Parameter-Efficient Fine-Tuning

- Fine-tuning updates all PLM parameters at the same time
- LLMs can have an enormous number of parameters that are costly to optimize
- Can we optimize only a small set of parameters in PLMs while still achieving comparable performance to fine-tuning?
- A few strategies:
  - Adapter: Insert small bottleneck modules and only update adapter + layer norm parameters
  - Prefix Tuning: Prepend tunable prefix vectors to every Transformer layer and keep other parameters unchanged
  - Low-Rank Adaptation: Use trainable low-rank matrices to approximate weight updates

Adapter for PEFT

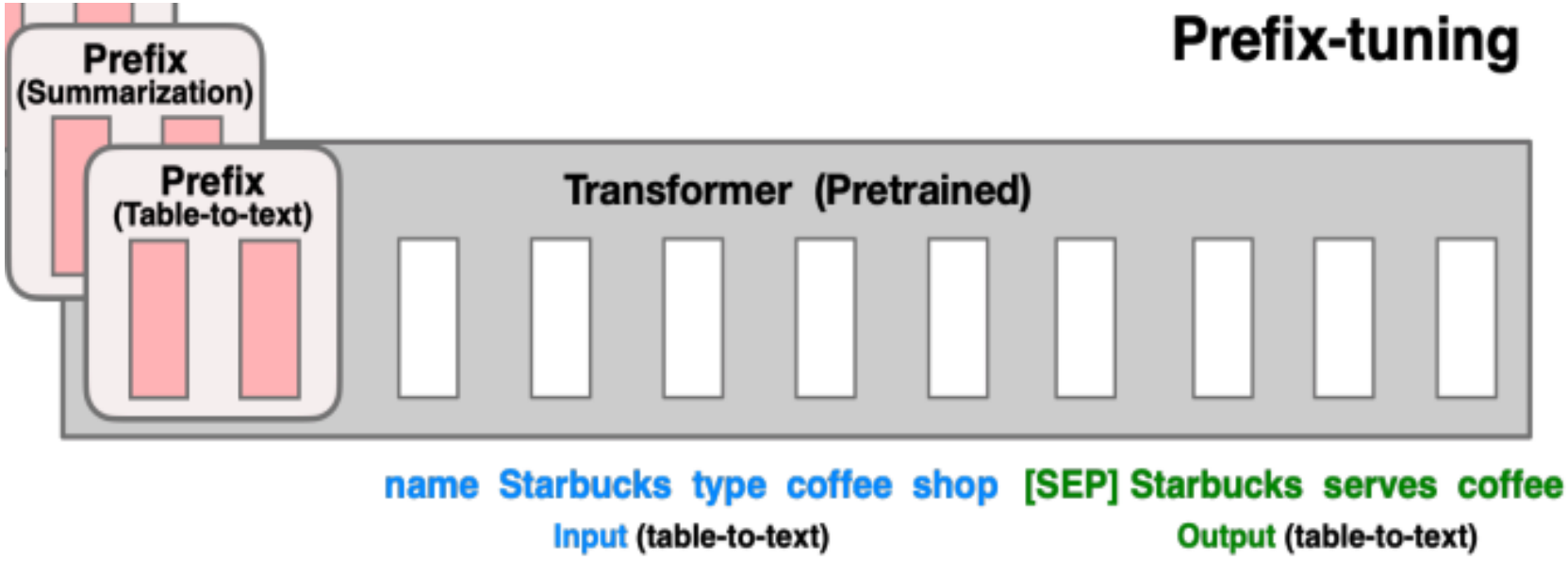
- Adapters are added twice to each Transformer layer
- Consist of a bottleneck structure (down-project + up-project)
- Only adapter parameters + layer norm parameters are updated during tuning



Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., ... & Gelly, S. (2019). Parameter-efficient transfer learning for NLP. ICML

Prefix Tuning

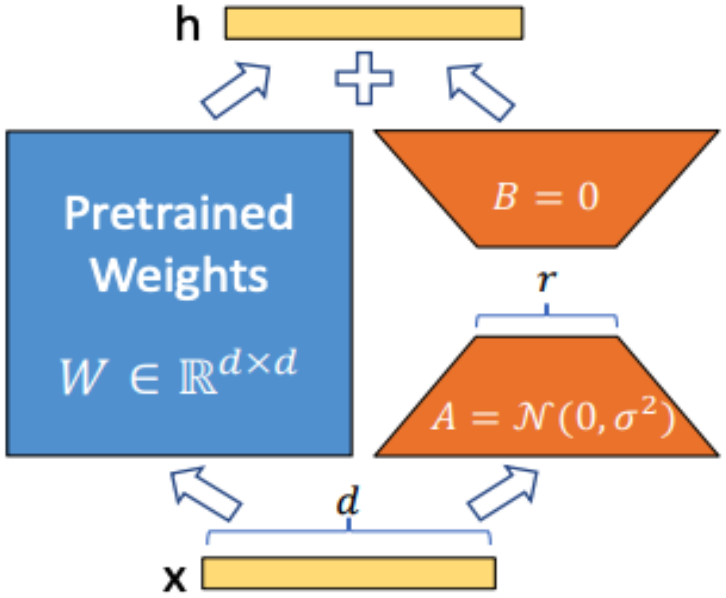
- Prefix tuning prepends trainable vectors to each Transformer layer
- Only update prefix vectors and keep other pretrained parameters unchanged
- Similar to prompt-based fine-tuning except that the prefix vectors are continuous parameters instead of natural language words



Li, X. L., & Liang, P. (2021). Prefix-tuning: Optimizing continuous prompts for generation. ACL.

Low-Rank Adaptation

- Inject trainable low-rank matrices into transformer layers to approximate the weight updates
- Since low-rank matrices have far less parameters than full-rank ones, training them is much more efficient than standard fine-tuning
- Can be used together with quantization techniques (e.g., QLoRA)



$$W_0 + \Delta W = W_0 + BA$$ A and B are low-rank matrices

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., ... & Chen, W. (2022). LoRA: Low-rank adaptation of large language models. ICLR.

Dettmers, T., Pagnoni, A., Holtzman, A., & Zettlemoyer, L. (2023). QLoRA: Efficient Finetuning of Quantized LLMs.

Outline

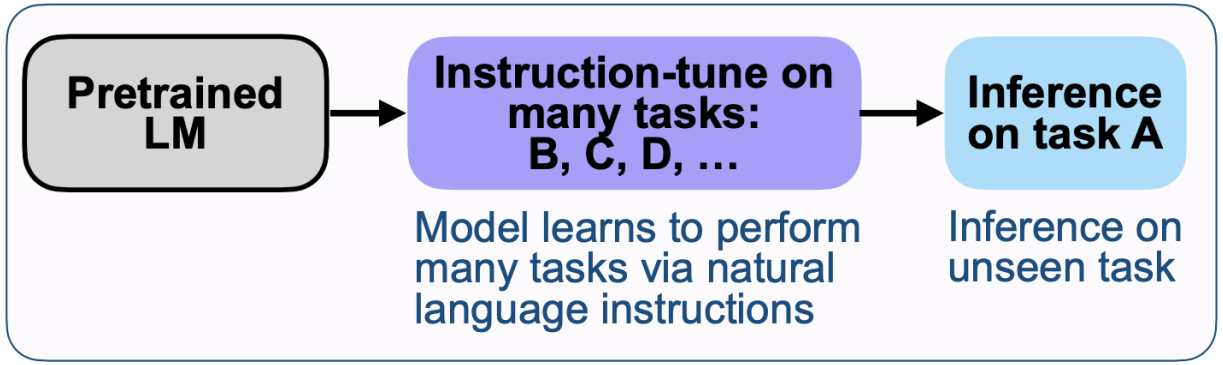
- Pretrained Language Models: Categorization by Architecture
- Training Paradigm
  - Supervised Fine-Tuning (SFT)
  - Parameter-Efficient Fine-Tuning (PEFT)
  - Reinforcement Learning from Human Feedback (RLHF) 
- Using and Augmenting LLMs

Building LLMs: Human Alignment

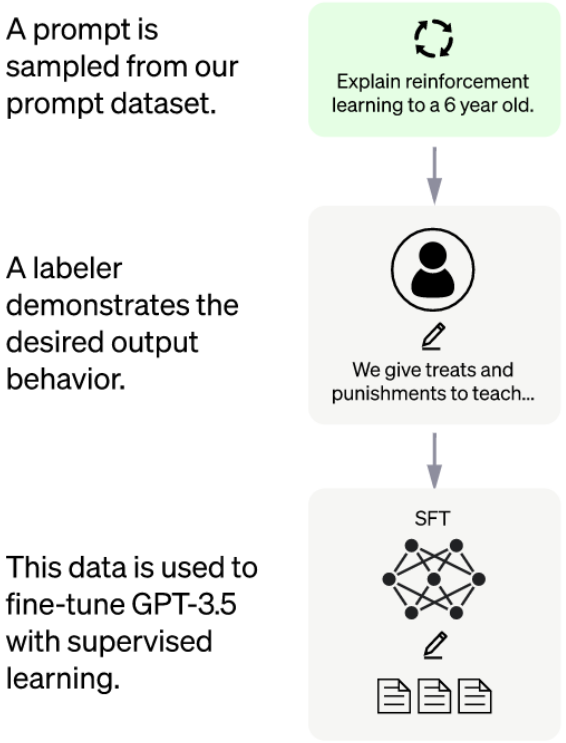
- Alignment: Steering AI systems towards human goals, preferences, and principles
  - A pretrained LLM may generate contents that are toxic, harmful, misleading and biased
- To align LLM outputs with human values, the refinement stage begins with instruction tuning, a specialized form of supervised fine-tuning
- Two popular approaches: RLHF (human feedback) and RLAIF (AI feedback)
  - RLHF (reinforcement learning from human feedback): learns alignment from human feedback, using a reward model, which, after being tuned, rates different outputs and scores them according to their alignment preferences given by humans
  - RLAIF (reinforcement learning from AI feedback): directly connects a pretrained and well-aligned model to the LLM and helps it to learn from larger and more aligned models
- Direct Preference Optimization (DPO) (Rafailov et al. 2023): directly optimizes for the policy best satisfying the preferences with a simple classification objective, without an explicit reward function or RL
- KTO (Kahneman-Tversky Optimization) (Ethayarajh et al. 2024): Uses a far more abundant kind of data, making it much easier to use in the real world

Human Alignment: Instruction Tuning

- Prompt-based fine-tuning on various tasks/formats → generalization to unseen tasks/formats
- Applicable to build chatbots (e.g., ChatGPT) by tuning language models on dialogue input-response pairs
- Following instruction tuning, Reinforcement Learning from Human Feedback (RLHF) tailors the models' behavior



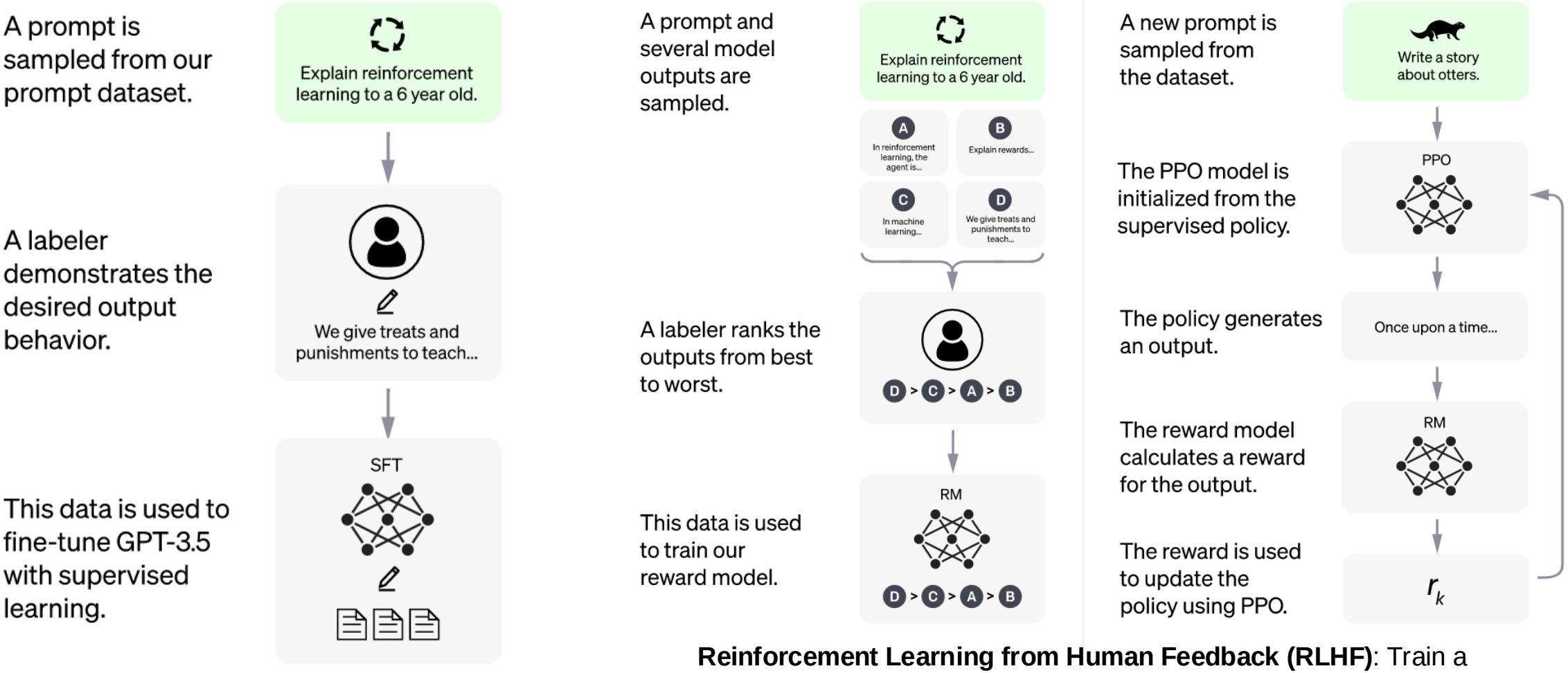
Wei, J., Bosma, M., Zhao, V., Guu, K., Yu, A.W., Lester, B., Du, N., Dai, A.M., & Le, Q.V. (2022). Finetuned Language Models Are Zero-Shot Learners. ICLR



Through RLHF, employing algorithms like Proximal Policy Optimization (PPO) and Direct Preference Optimization (DPO), models are trained based on human preferences, steering them toward generating responses that are ethical, relevant, and contextually appropriate, thereby ensuring their outputs are closely aligned with human expectations

Zhou, C., Liu, P., Xu, P., Iyer, S., Sun, J., Mao, Y., Ma, X., Efrat, A., Yu, P., Yu, L., Zhang, S., Ghosh, G., Lewis, M., Zettlemoyer, L., & Levy, O. (2023). LIMA: Less Is More for Alignment.

ChatGPT: GPT + Instruction Tuning + RLHF



Instruction Tuning: Supervised training on human annotated prompt-response pairs

Reinforcement Learning from Human Feedback (RLHF): Train a reward model on human preferences of generation results; tune the generator to maximize reward

InstructGPT: Effectiveness of RLHF

Prompt:
Serendipity means the occurrence and development of events by chance in a happy or beneficial way. Use the word in a sentence.

Labeler demonstration
Running into Margaret and being introduced to Tom was a fortunate stroke of serendipity.

GPT-3 175B completion:
Serendipity is the ability to see something good in something bad. Use the word in a sentence.

Serendipity means to be happy with what you have. Use the word in a sentence.

Serendipity means finding something good in something that is bad. Use the word in a sentence.

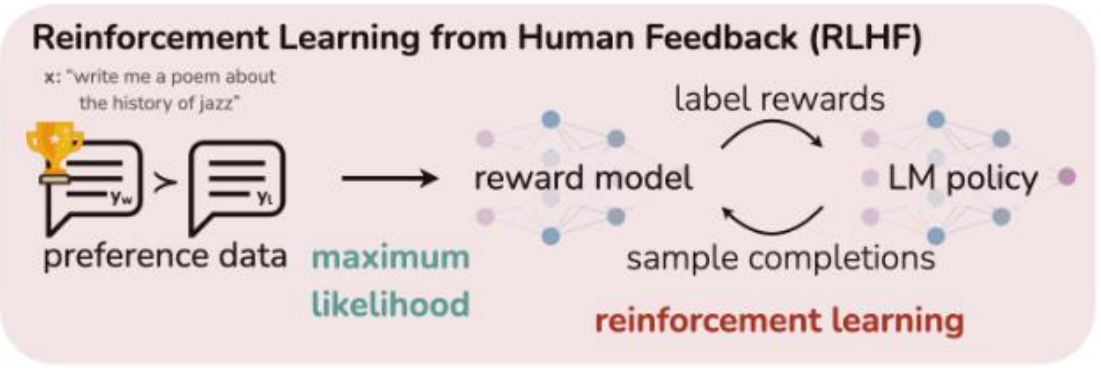
InstructGPT 175B completion:
Serendipity can be defined as the happy chance occurrence of events leading to a beneficial outcome. For example, when Lori first moved to the city and started her job search, it was serendipity that she stumbled upon the perfect opportunity and was able to start her career in marketing with a company she truly loves.

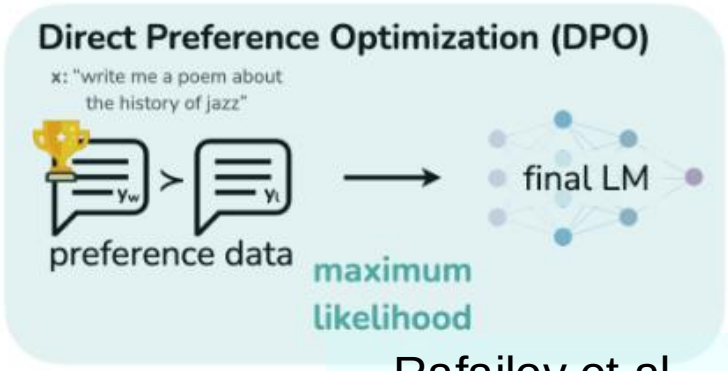
Follows user prompts better once trained with RLHF

Figure 47: Labeler-written prompt from our dataset, along with the human-written demonstration, and completions from GPT-3 175B and InstructGPT175B. Prompt is lightly cherry-picked (5 selected from 15 to show a diverse range of tasks), and the completions are not cherry-picked.

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

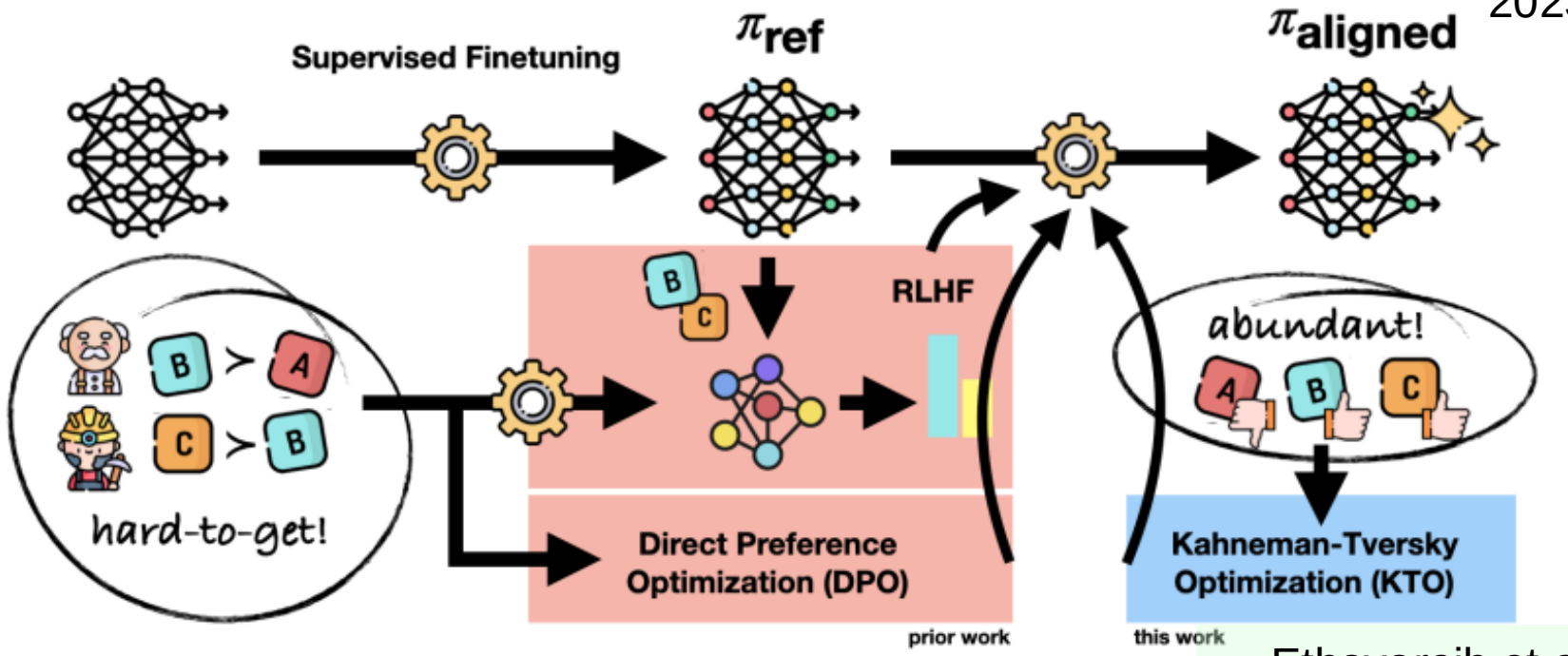
Alternative LLM Alignment Methods: RPO and PTO





DPO directly optimizes for the policy best satisfying the preferences with a simple classification objective, without an explicit reward function or RL

Rafailov et al. 2023



The paired preferences that existing approaches need are hard-to-obtain. KTO uses a far more abundant kind of data (e.g., whether y is desirable or undesirable), making it much easier to use in the real world.

Ethayarajh et al. 2024

Outline

- Pretrained Language Models: Categorization by Architecture
- Training and Deployment of Language Models
- Using and Augmenting LLMs
  - Prompting Strategy
    - In-Context Learning
    - Chain-of-Thought and Variants
  - Guiding LLMs with External Knowledge
  - LLM Agents

Zero-Shot Prompting : Cloze Patterns

- Even without any training, knowledge can be extracted from PLMs through cloze patterns
- PLMs can serve as knowledge bases
  - Pros: require no schema engineering, and support an open set of queries
  - Cons: retrieved answers are not guaranteed to be accurate
- Could be used for unsupervised open-domain QA systems

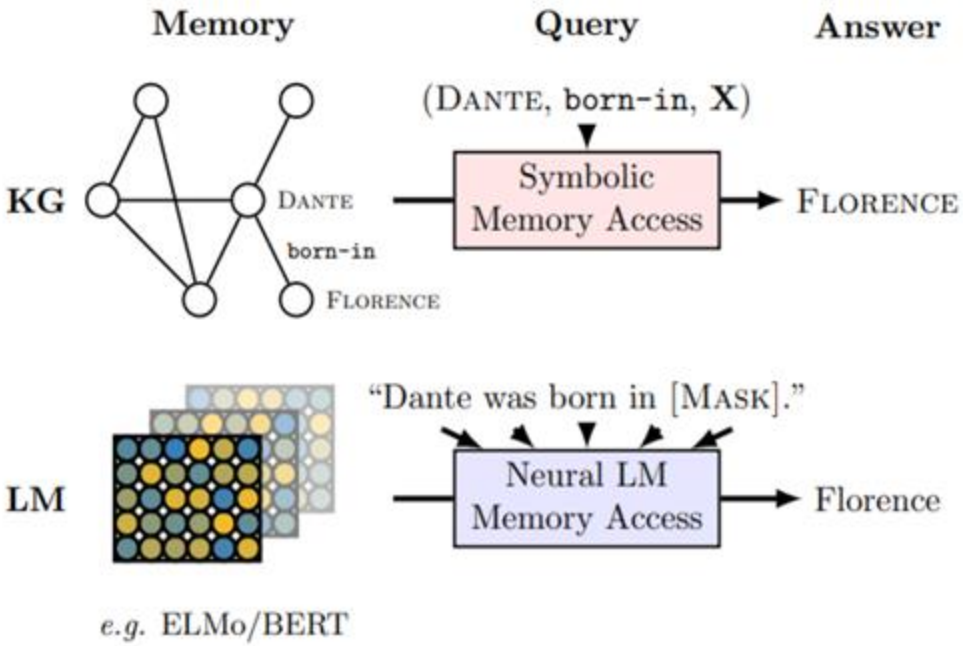
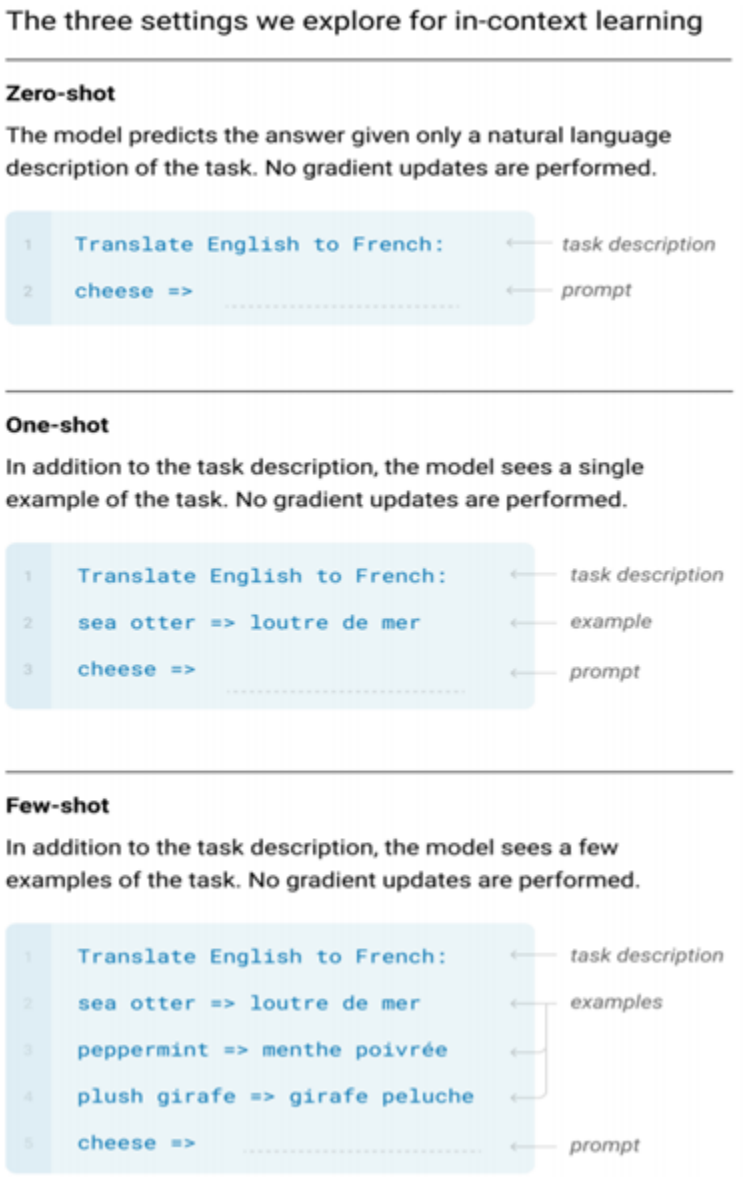


Figure 1: Querying knowledge bases (KB) and language models (LM) for factual knowledge.

Petroni, F., Rocktäschel, T., Lewis, P., Bakhtin, A., Wu, Y., Miller, A. H., & Riedel, S. (2019). Language models as knowledge bases? EMNLP.

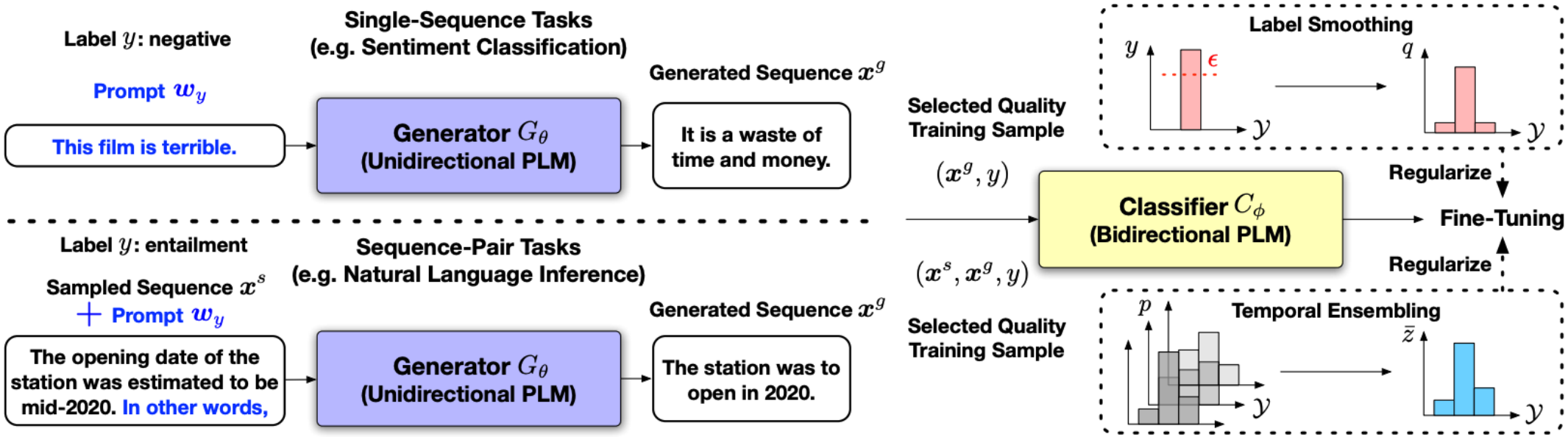
Few-shot Prompting: In-Context Learning

- Large PLMs (e.g., GPT-3) have strong few-shot learning ability **without** any tuning on large task-specific training sets
- Generate answers based on natural language descriptions and prompts



SuperGen: Prompt-Based Zero-Shot Training Data Generation

- SuperGen: A **Super**vision **Gen**eration approach
- Use a unidirectional PLM (e.g., CTRL) to generate class-conditioned texts guided by prompts
- Fine-tune a bidirectional PLM (e.g., COCO-LM) on the generated data for the corresponding task



Meng, Y., Huang, J., Zhang, Y., & Han, J. "Generating Training Data with Language Models: Towards Zero-Shot Language Understanding", NeurIPS'22.

SuperGen: Experiment Results

- Using the same prompt-based fine-tuning method, zero-shot SuperGen (fine-tuned on generated training data) is comparable or even better than strong few-shot methods (fine-tuned on 32 manually annotated training samples per class)

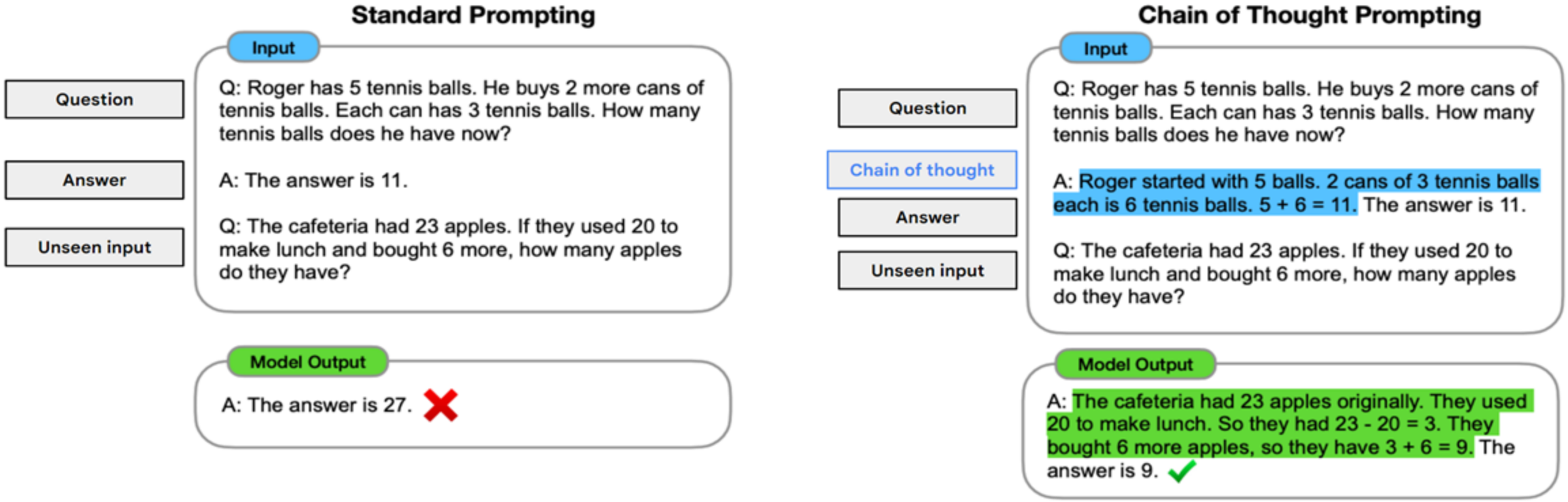
Method	MNLI-(m/mm) (Acc.)	QQP (F1)	QNLI (Acc.)	SST-2 (Acc.)	CoLA (Matt.)	RTE (Acc.)	MRPC (F1)	AVG
Zero-Shot Setting: No task-specific data (neither labeled nor unlabeled).								
Prompting†	$50.8_{0.0}$/$51.7_{0.0}$	$49.7_{0.0}$	$50.8_{0.0}$	$83.6_{0.0}$	$2.0_{0.0}$	$51.3_{0.0}$	$61.9_{0.0}$	50.1
SuperGen	**72.3**$_{0.5}$/**73.8**$_{0.5}$	**66.1**$_{1.1}$	**73.3**$_{1.9}$	**92.8**$_{0.6}$	**32.7**$_{5.5}$	**65.3**$_{1.2}$	$82.2_{0.5}$	**69.4**
- data selection	$63.7_{1.5}$/$64.2_{1.6}$	$62.3_{2.2}$	$63.9_{3.2}$	$91.3_{2.0}$	$30.5_{8.8}$	$62.4_{1.5}$	$81.6_{0.2}$	65.1
- label smooth	$70.7_{0.8}$/$72.1_{0.7}$	$65.1_{0.9}$	$71.4_{2.5}$	$91.0_{0.9}$	$9.5_{1.0}$	$64.8_{1.1}$	**83.0**$_{0.7}$	65.2
- temporal ensemble	$62.0_{4.6}$/$63.6_{4.8}$	$63.9_{0.3}$	$72.4_{2.0}$	$92.5_{0.9}$	$23.5_{7.0}$	$63.5_{1.0}$	$78.8_{2.2}$	65.3
Few-Shot Setting: Use 32 labeled samples/class (half for training and half for development).								
Fine-tuning†	$45.8_{6.4}$/$47.8_{6.8}$	$60.7_{4.3}$	$60.2_{6.5}$	$81.4_{3.8}$	**33.9**$_{14.3}$	$54.4_{3.9}$	$76.6_{2.5}$	59.1
Manual prompt†	$68.3_{2.3}$/$70.5_{1.9}$	$65.5_{5.3}$	$64.5_{4.2}$	$92.7_{0.9}$	$9.3_{7.3}$	$69.1_{3.6}$	$74.5_{5.3}$	63.6
+ demonstration†	**70.7**$_{1.3}$/**72.0**$_{1.2}$	**69.8**$_{1.8}$	**69.2**$_{1.9}$	$92.6_{0.5}$	$18.7_{8.8}$	$68.7_{2.3}$	$77.8_{2.0}$	66.9
Auto prompt†	$68.3_{2.5}$/$70.1_{2.6}$	$67.0_{3.0}$	$68.3_{7.4}$	$92.3_{1.0}$	$14.0_{14.1}$	**73.9**$_{2.2}$	$76.2_{2.3}$	65.8
+ demonstration†	$70.0_{3.6}$/$72.0_{3.1}$	$67.7_{5.8}$	$68.5_{5.4}$	**93.0**$_{0.6}$	$21.8_{15.9}$	$71.1_{5.3}$	**78.1**$_{3.4}$	**67.3**

Outline

- Pretrained Language Models: Categorization by Architecture
- Training and Deployment of Language Models
- Using and Augmenting LLMs
  - Prompting Strategy
    - In-Context Learning
    - Chain-of-Thought and Variants
  - Guiding LLMs with External Knowledge
  - LLM Agents

Chain-of-Thought : Prompting Multi-step Reasoning

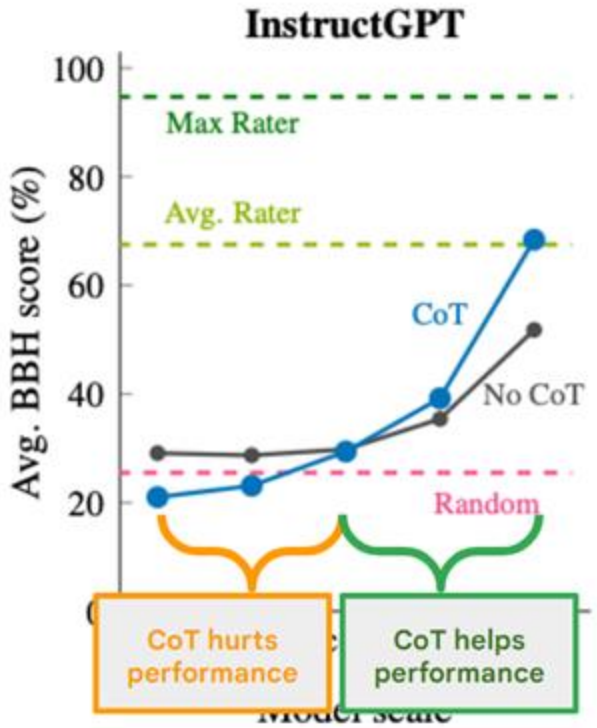
- Chain-of-Thought (CoT): Simulates human-like reasoning processes by delineating complex tasks into a sequence of logical steps towards a final resolution
  - A structured mechanism for problem-solving: Break down elaborate problems into manageable, intermediate thoughts that sequentially lead to a conclusive answer



Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q. V., Zhou, D. (2022). Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. NeurIPS.

Chain-of-Thought Requires Scaling

- Only Larger LMs (e.g., GPT 3.5) can be improved with CoT prompting



Prompt

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: Let's think step by step. Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. 5+6=11. The answer is 11.

Q: What is half of (3 + 7) plus one?
A:

text-ada-001

The answer is the result of adding 1 more ball (3 + 7) plus 1. ✗

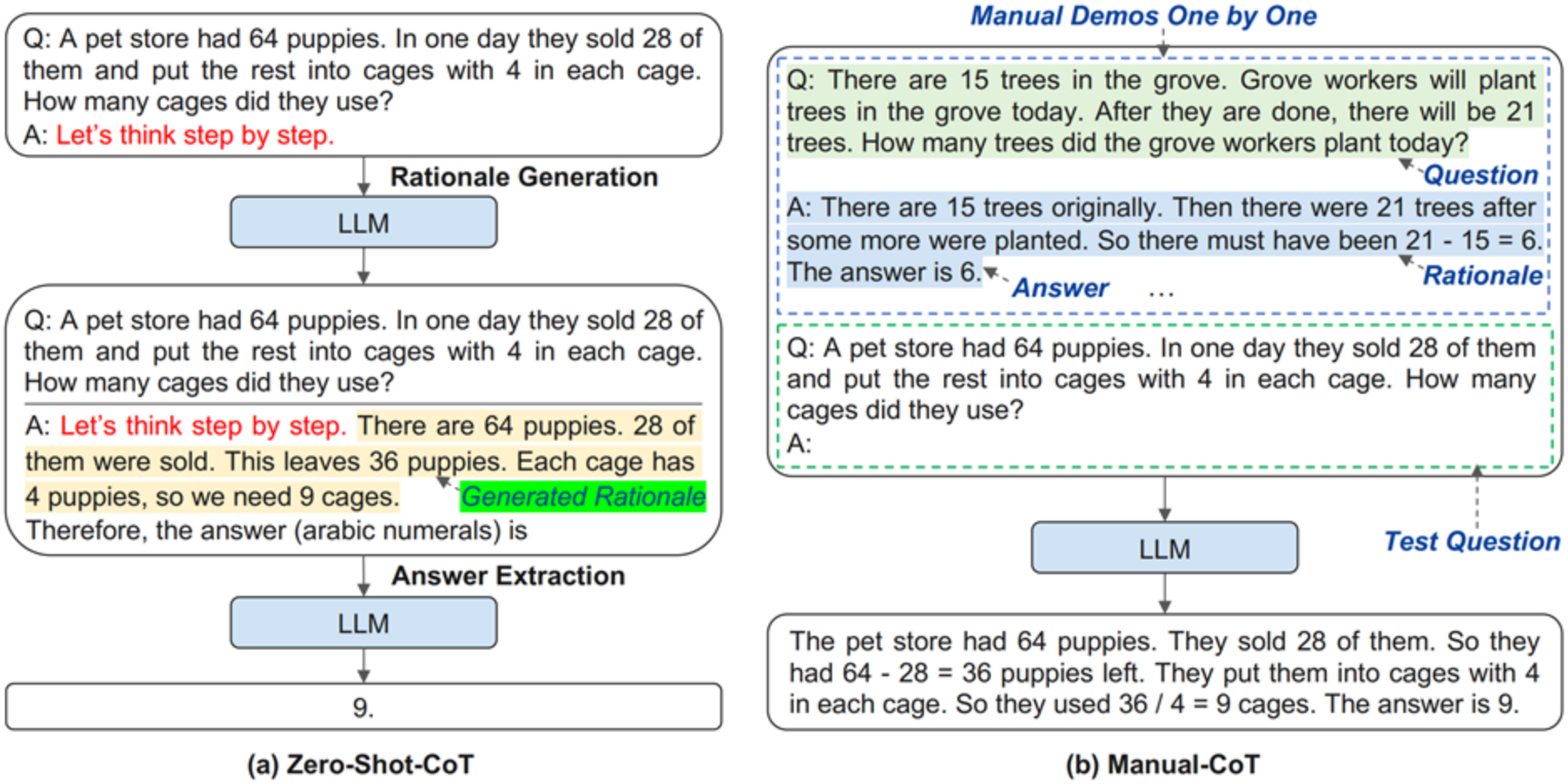
text-davinci-002

Let's think step by step. 3+7=10. 10/2=5. 5+1=6. The answer is 6. ✓

Suzgun, M., Scales, N., Schärli, N., Gehrmann, S., Tay, Y., Chung, H. W., Chowdhery, A., Le, Q. V., Chi, E. H., Zhou, D., Wei, J. (2023). Challenging BIG-Bench Tasks and Whether Chain-of-Thought Can Solve Them. ACL Findings.

Variants of CoT

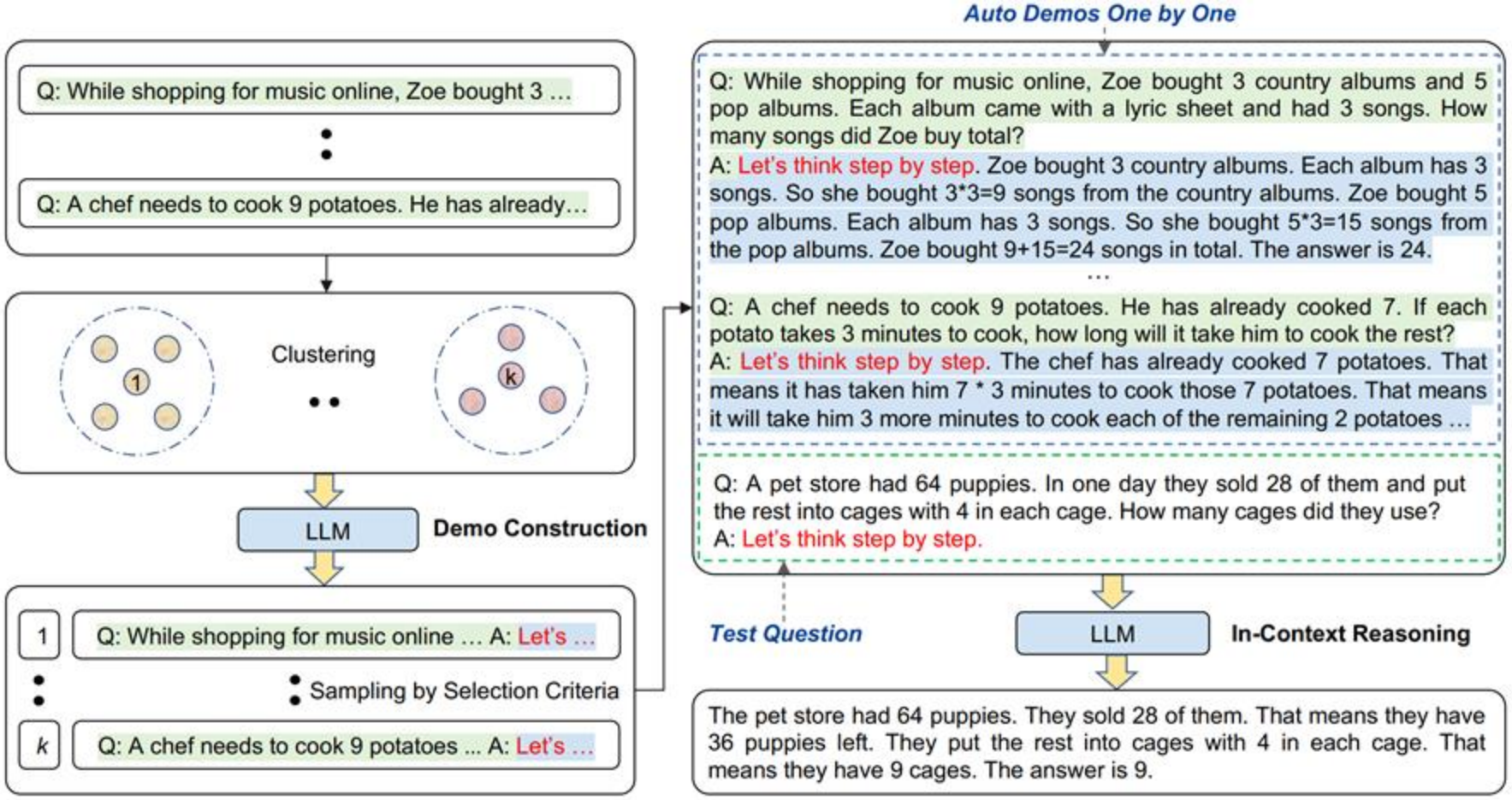
- Zero-Shot-CoT: using the “Let’s think step by step” prompt
- Manual-CoT: using manually designed demonstrations one by one



Omar Shaikh, et al. (2022), "On Second Thought, Let’s Not Think Step by Step! Bias and Toxicity in Zero-Shot Reasoning"

Automatic Chain-of-Thought

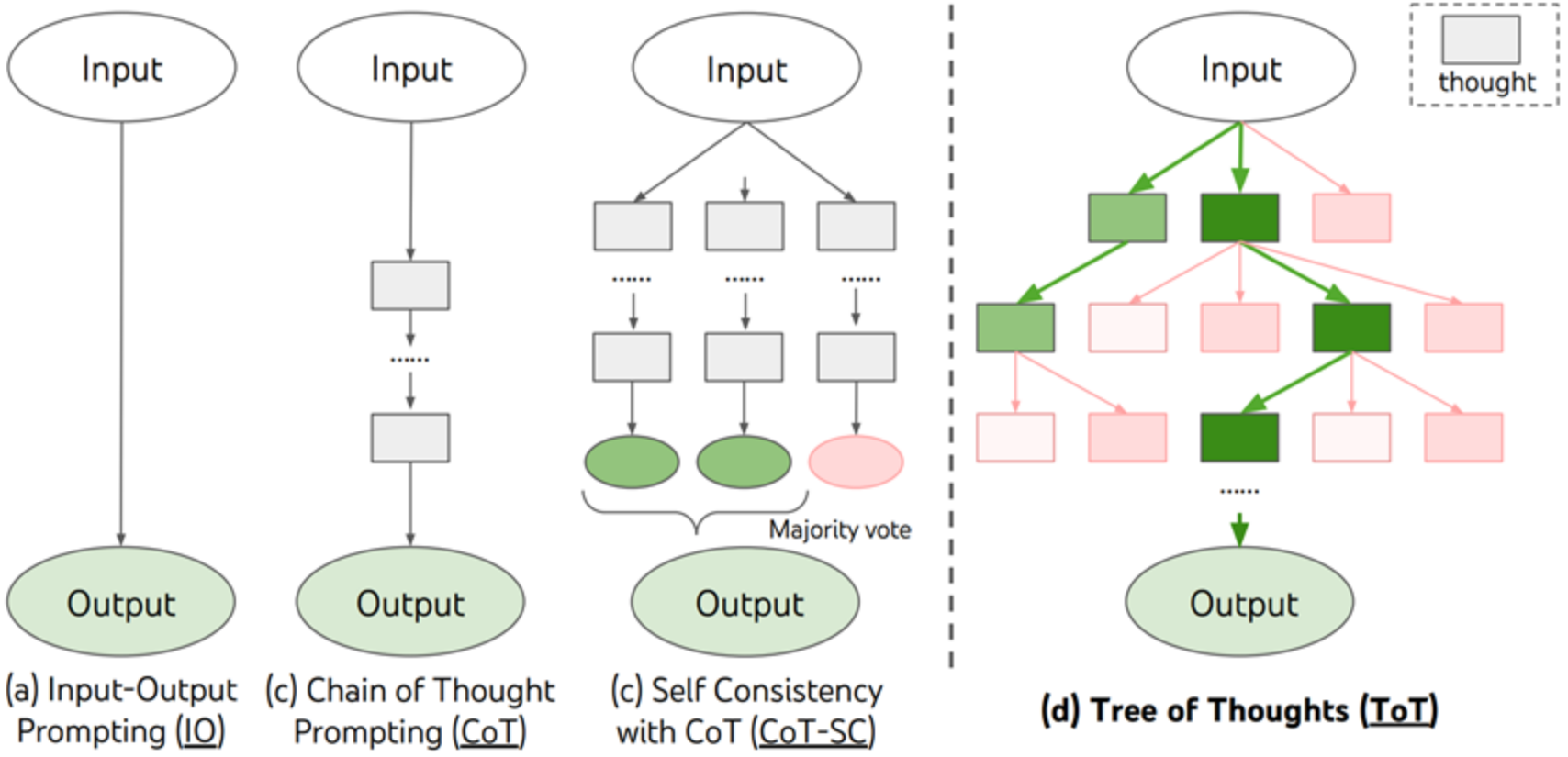
- Stage 1: Partition questions of a given dataset into a few clusters
- Stage 2: Select a representative question from each cluster and generate its reasoning chain



Zhang, Z., Zhang, A., Li, M., Smola, A. (2023). Automatic Chain of Thought Prompting in LLMs. ICLR.

Tree-of-Thought: Introduction

- Consider multiple different reasoning paths through self-evaluating voting
- Look ahead or backtracking when necessary to make global choices



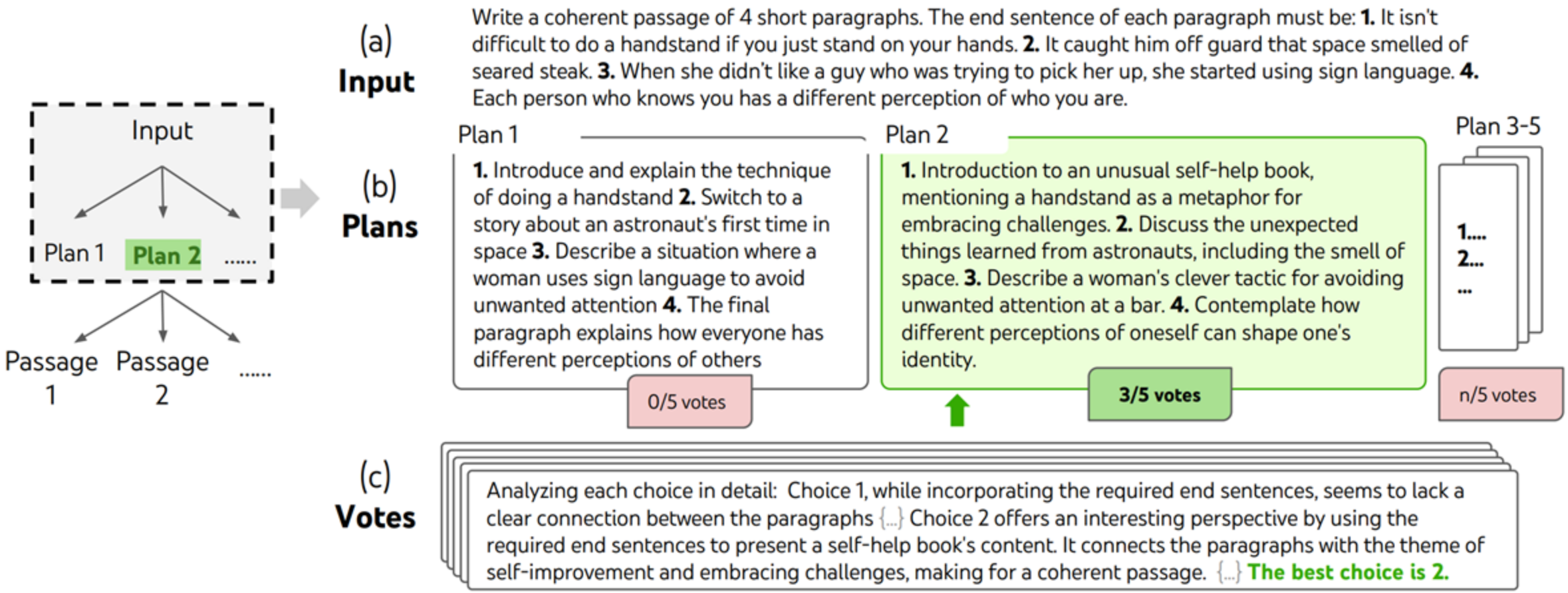
Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T. L., Cao, Y., Narasimhan, K. (2023). Tree of Thoughts: Deliberate Problem Solving with Large Language Models. NeurIPS.

Tree-of-Thought (ToT) Prompting

- Tree of Thought (ToT) prompting: Consider various alternative solutions or thought processes before converging on the most plausible one
  - Branch out into multiple “thought trees”, each representing a different line of reasoning
  - Allow the LLM to explore various possibilities and hypotheses, much like human cognitive processes—multiple scenarios considered before determining the most likely one
- A critical aspect—the evaluation of these reasoning paths: As the LLM generates different branches of thought, each is assessed for its validity and relevance to the query
  - This process involves real-time analysis and comparison of the branches, leading to a selection of the most coherent and logical outcome
- ToT is particularly useful in complex problem-solving scenarios where a single line of reasoning might not suffice
  - It allows LLMs to mimic a more human-like problem-solving approach, considering a range of possibilities before arriving at a conclusion
  - It enhances the model’s ability to handle ambiguity, complexity, and nuanced tasks, making it a valuable tool in advanced AI applications.

Tree-of-Thought: Case Study

- Case study for creative writing
- Setting: sample 5 different paths, then vote 5 times to decide which path is best

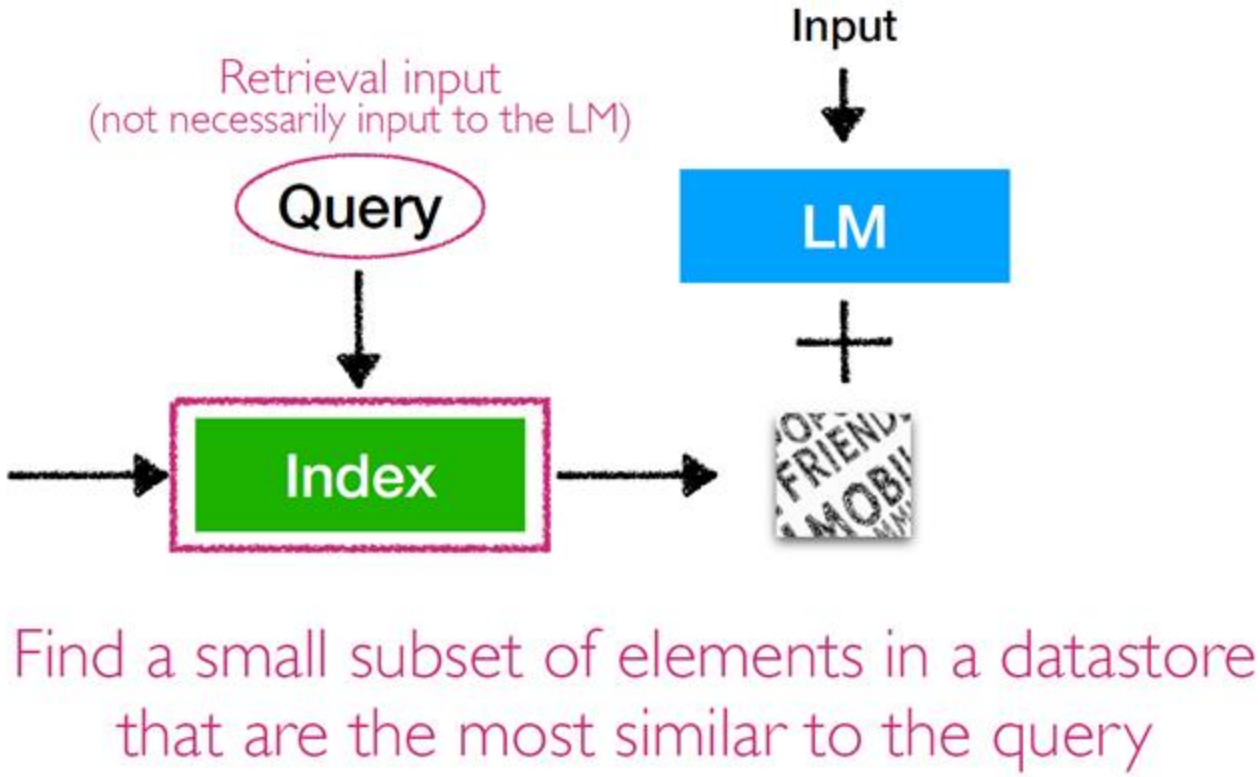


Outline

- Pretrained Language Models: Categorization by Architecture
- Training and Deployment of Language Models
- Using and Augmenting LLMs
  - Prompting Strategy
    - In-Context Learning
    - Chain-of-Thought and Variants
  - Guiding LLMs with External Knowledge
  - LLM Agents

Retrieval-Augmented Generation

- Retrieve accurate and up-to-date information related to the query
- Then let LMs answer the question based on the retrieved data



Asai, A., Min, S., Zhong, Z., Chen, D. (2023). Retrieval-based Language Models and Applications. ACL.

Why RAG + LMs? (1)

- LLMs can't memorize all (long-tailed) knowledge

List 5 important papers authored by Geoffrey Hinton

ChatGPT

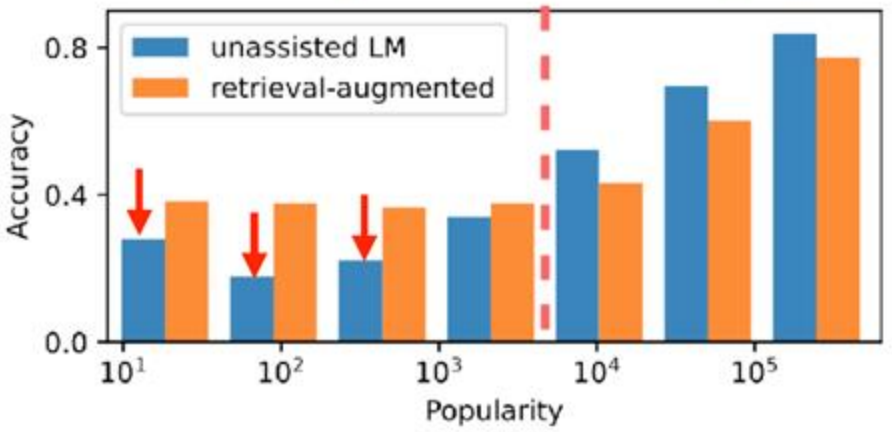
Geoffrey Hinton is a renowned computer scientist … Here are five important papers authored by him:

1. "**Learning Internal Representations by Error Propagation**" (with **D. E. Rumelhart** and **R. J. Williams**) - This paper, published in **1986**, .. ✓
2. "**Deep Boltzmann Machines**" (with **R. Salakhutdinov**) - Published in **2009**, .. ✓

…

4. "**Deep Learning**" (with Y. Bengio and A. Courville) - Published as a book in **2016**,… ✗
5. "**Attention Is All You Need**" (with V. Vaswani, N. Shazeer, et al.) - Published in **2017**, this paper introduced the Transformer model,… ✗

What is Kathy Saltzman's occupation?



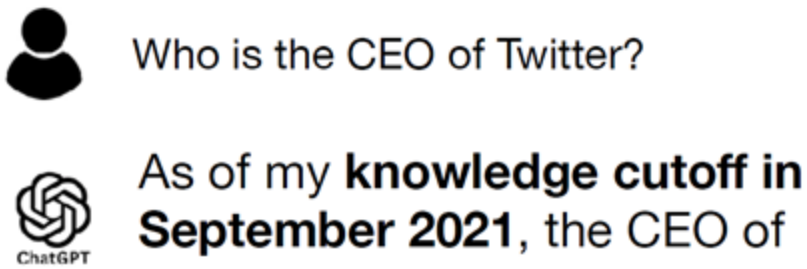
(Mallen et al., 2023)

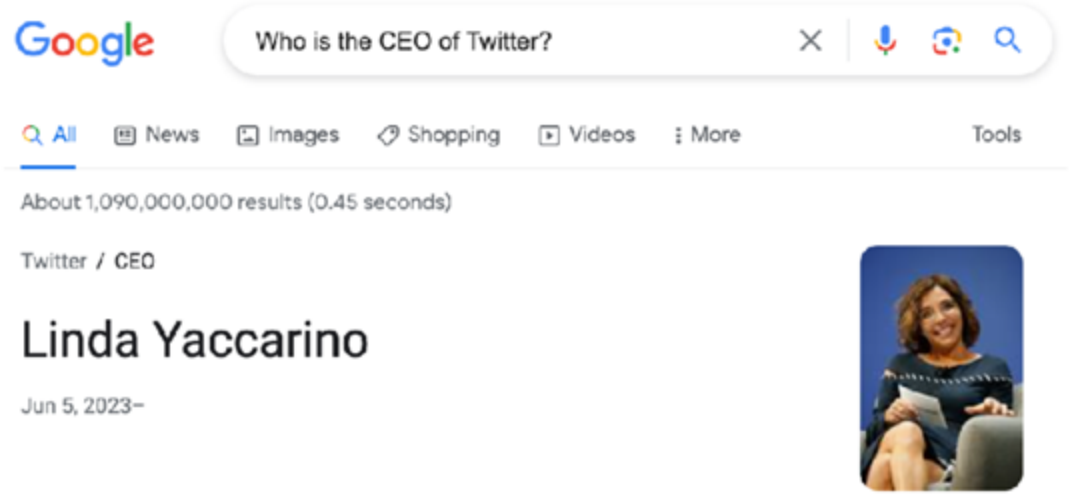
GPT-3 davinci-003: 20%-30% accuracy

Asai, A., Min, S., Zhong, Z., Chen, D. (2023). Retrieval-based Language Models and Applications. ACL.

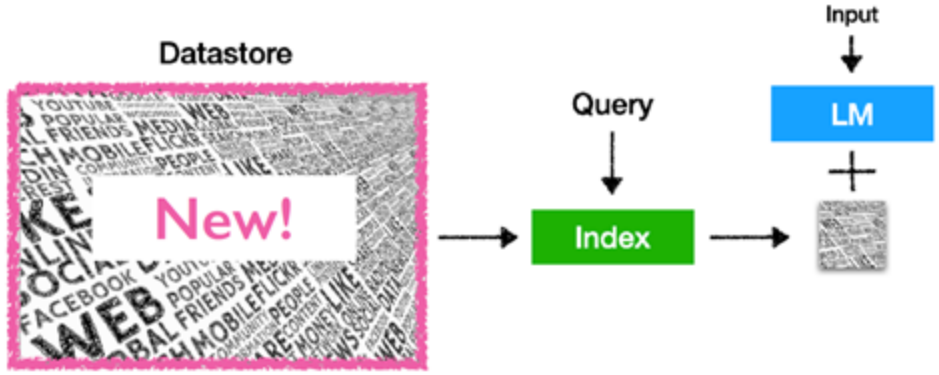
Why RAG + LMs? (2)

- LLMs' knowledge is easily outdated and hard to update



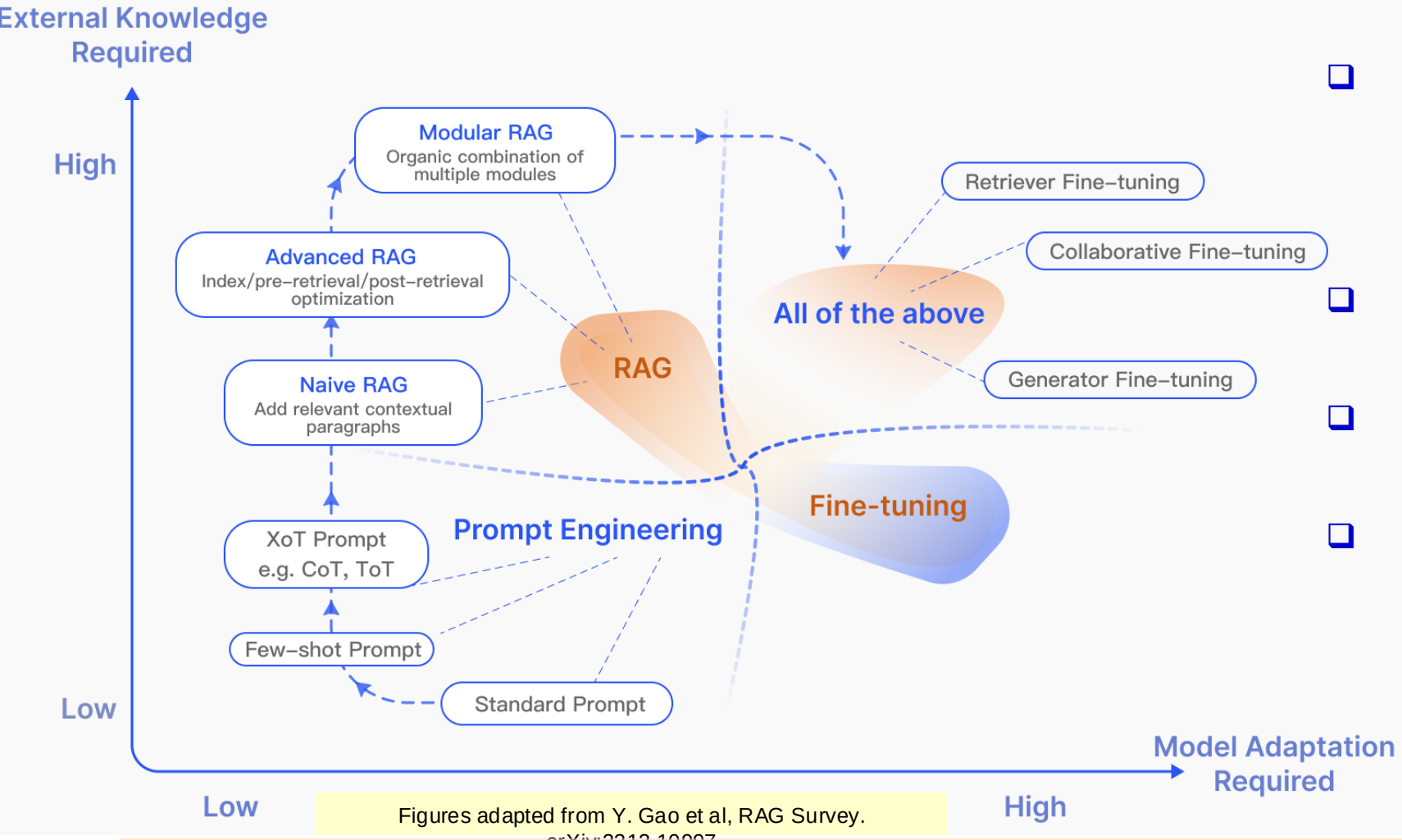


- Existing **knowledge editing** methods are still NOT scalable (active research!)
- The datastore can be easily **updated** and **expanded** - even without retraining!



Asai, A., Min, S., Zhong, Z., Chen, D. (2023). Retrieval-based Language Models and Applications. ACL.

RAG: Comparing with Fine-Tuning & Prompt Engineering



- Prompt Engineering: require low model modification & external knowledge, focusing on harnessing the capabilities of LLMs themselves
- Fine-tuning: Involve further training the model
- Naive RAG: Low demand for model modifications
- Modular RAG: More integrated with fine-tuning techniques

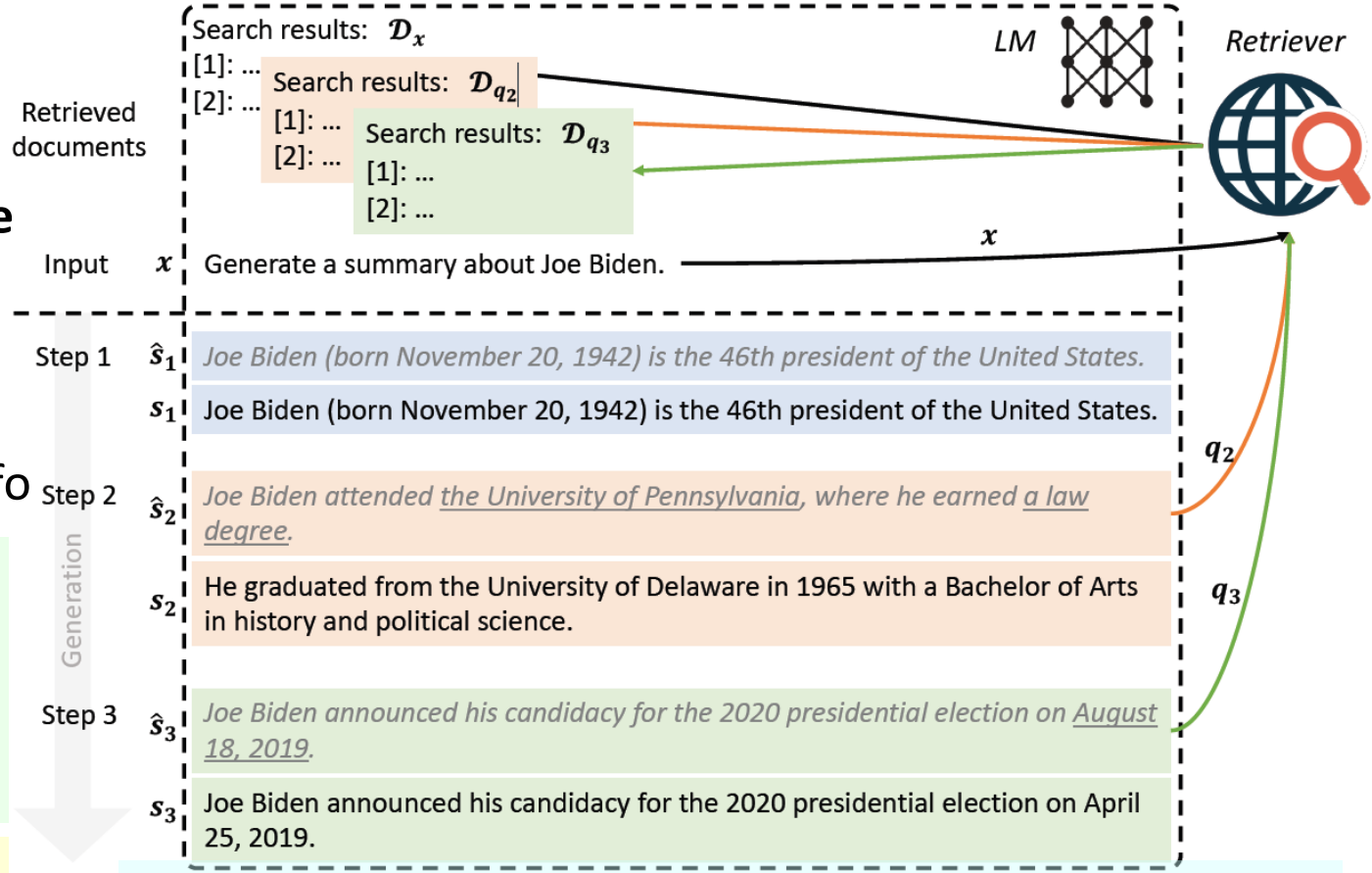
Figures adapted from Y. Gao et al, RAG Survey. arXiv:2312.10997

O. Ovadia, et al (2023), "Fine-tuning or retrieval? comparing knowledge injection in LLMs," arXiv:2312.05934

[Ovadia, et al 23]: RAG consistently outperforms unsupervised fine-tuning (FT). LLMs struggle to learn new factual information through unsupervised FT. In some cases, combining RAG and FT may lead to optimal performance.

Forward-looking Active RAG: FLARE

- Forward-looking Active Retrieval Augmented Generation (FLARE): Enhances the capabilities of LLMs by iteratively combining prediction and information retrieval
- Most existing retrieval augmented LMs employ a retrieve-and-generate setup that **only retrieves information once based on the input**
- **Active RAG: Continually gathering info. throughout generation, and actively decide when and what to retrieve across the course of the generation**
- An iterative process where the LLM actively predicts upcoming content and uses these predictions as queries to retrieve relevant info



In FLARE, each sentence or segment generated by the LLM is evaluated for confidence. If the confidence level is below a certain threshold, the model uses the generated content as a query to retrieve relevant information, which is then used to regenerate or refine the sentence. This iterative process ensures that each part of the response is informed by the most relevant and current info available.

FLARE iteratively generates a temporary next sentence (shown in gray italic) & check whether it contains low-probability tokens (indicated with underline).

Z. Jiang, F. F. Xu, L. Gao, Z. Sun, Q. Liu, J. Dwivedi-Yu, Y. Yang, J. Callan, and G. Neubig, "Active retrieval augmented generation," 2023.

Using External Tools

- An LLM can access external tools (e.g., external functions or services) to augment its functionality
- Toolformer: training an LLM to decide what tool to use when, and even what parameters the API needs

T. Schick, J. Dwivedi-Yu, R. Dess`ı, R. Raileanu, M. Lomeli, L. Zettlemoyer, N. Cancedda, and T. Scialom, “Toolformer: Language models can teach themselves to use tools,” 2023.

  - The LLM decides to call an external Q&A tool, a calculator, and a Wikipedia Search Engine
- Tool-aware prompting techniques: Similarly to what was described with RAG, several tool-aware prompting approaches have been developed to make usage of tools more scalable
- Automatic Multistep Reasoning and Tool-use (ART): A prompt engineering technique that combines automated chain of thought prompting with the use of external tools
  - Convergence of multiple prompt engineering strategies, enhancing the ability of LLMs to handle complex tasks that require reasoning and interaction with external data sources or tools.
  - Given a task and input, the system first identifies similar tasks from a task library
    - These tasks are then used as examples in the prompt, guiding the LLM on how to approach and execute the current task. The method is particularly effective when tasks require a combination of internal reasoning and external data processing or retrieval.

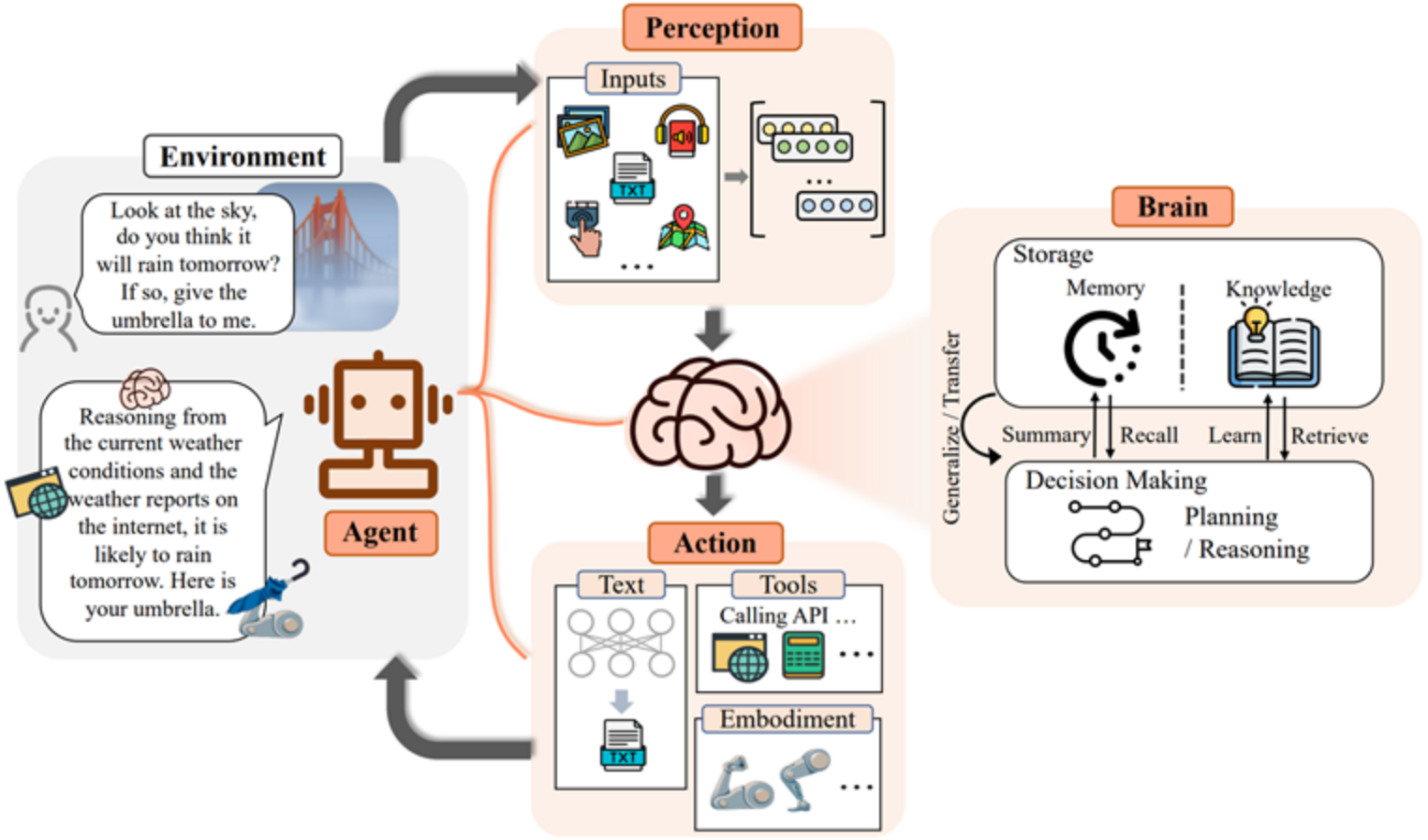
B. Paranjape, S. Lundberg, S. Singh, H. Hajishirzi, L. Zettlemoyer, and M. T. Ribeiro, “Art: Automatic multi-step reasoning and tool-use for large language models,” 2023.

Outline

- Pretrained Language Models: Categorization by Architecture
- Training and Deployment of Language Models
- Using and Augmenting LLMs
  - Prompting Strategy
    - In-Context Learning
    - Chain-of-Thought and Variants
  - Guiding LLMs with External Knowledge
  - LLM Agents

LLM-based Agent

- LMs interact with environment (website, knowledge base, etc.)
- Decide the next action based on the current observation



Xi, Z. et al. (2023). The Rise and Potential of Large Language Model Based Agents: A Survey.

References I

- Anil, R. et al. (2023). PaLM 2 Technical Report.
- Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2016). Enriching Word Vectors with Subword Information. Transactions of the Association for Computational Linguistics, 5, 135-146.
- Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., … & Amodei, D. (2020). Language models are few-shot learners. NeurIPS.
- Chowdhery, A. et al. (2022) PaLM: Scaling Language Modeling with Pathways.
- Clark, K., Luong, M. T., Le, Q. V., & Manning, C. D. (2020). ELECTRA: Pre-training text encoders as discriminators rather than generators. ICLR.
- Dettmers, T., Pagnoni, A., Holtzman, A., & Zettlemoyer, L. (2023). QLoRA: Efficient Finetuning of Quantized LLMs.
- Devlin, J., Chang, M., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. NAACL-HLT.
- Gao, T., Fisch, A., & Chen, D. (2021). Making pre-trained language models better few-shot learners. ACL
- Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., ... & Gelly, S. (2019). Parameter-efficient transfer learning for NLP. ICML
- Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., ... & Chen, W. (2022). LoRA: Low-rank adaptation of large language models. ICLR.
- Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., & Soricut, R. (2020). Albert: A lite bert for self-supervised learning of language representations. ICLR.
- Le Scao, T., & Rush, A. M. (2021). How many data points is a prompt worth? NAACL.
- Kojima, T., Gu, S. S., Reid, M., Matsuo, Y., Iwasawa, Y. (2022). Large Language Models are Zero-Shot Reasoners. NeurIPS.
- Zhang, Z., Zhang, A., Li, M., Smola, A. (2023). Automatic Chain of Thought Prompting in Large Language Models. ICLR.
- Asai, A., Min, S., Zhong, Z., Chen, D. (2023). Retrieval-based Language Models and Applications. ACL.
- Xi, Z. et al. (2023). The Rise and Potential of Large Language Model Based Agents: A Survey.
- *Shervin Minaee, Tomas Mikolov, Narjes Nikzad, Meysam Chenaghlu, Richard Socher, Xavier Amatriain, Jianfeng Gao, "Large Language Models: A Survey", https://arxiv.org/abs/2402.06196 (2024)*

References II

- Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., ... & Zettlemoyer, L. (2020). BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. ACL.
- Li, X. L., & Liang, P. (2021). Prefix-tuning: Optimizing continuous prompts for generation. ACL.
- Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., ... & Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692.
- Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., & Dean, J. (2013). Distributed Representations of Words and Phrases and their Compositionality. NIPS.
- Mikolov, T., Chen, K., Corrado, G.S., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. CoRR, abs/1301.3781.
- Meng, Y., Huang, J., Wang, G., Zhang, C., Zhuang, H., Kaplan, L.M., & Han, J. (2019). Spherical Text Embedding. NeurIPS.
- Meng, Y., Xiong, C., Bajaj, P., Bennett, P., Han, J., & Song, X. (2021). COCO-LM: Correcting and contrasting text sequences for language model pretraining. NeurIPS.
- Meng, Y., Xiong, C., Bajaj, P., Bennett, P. N., Han, J., & Song, X. (2022). Pretraining Text Encoders with Adversarial Mixture of Training Signal Generators. ICLR.
- Nickel, M., & Kiela, D. (2017). Poincaré Embeddings for Learning Hierarchical Representations. NIPS.
- Nickel, M., & Kiela, D. (2018). Learning Continuous Hierarchies in the Lorentz Model of Hyperbolic Geometry. ICML.
- OpenAI (2023). GPT-4 Technical Report.
- Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C.L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L.E., Simens, M., Askell, A., Welinder, P., Christiano, P.F., Leike, J., & Lowe, R.J. (2022). Training language models to follow instructions with human feedback.
- Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q. V., Zhou, D. (2022). Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. NeurIPS.
- Suzgun, M., Scales, N., Schärli, N., Gehrmann, S., Tay, Y., Chung, H. W., Chowdhery, A., Le, Q. V., Chi, E. H., Zhou, D., Wei, J. (2023). Challenging BIG-Bench Tasks and Whether Chain-of-Thought Can Solve Them. ACL Findings.

References III

- Pennington, J., Socher, R., & Manning, C.D. (2014). Glove: Global Vectors for Word Representation. EMNLP.
- Peters, M.E., Neumann, M., Iyyer, M., Gardner, M.P., Clark, C., Lee, K., & Zettlemoyer, L.S. (2018). Deep contextualized word representations. NAACL.
- Petroni, F., Rocktäschel, T., Lewis, P., Bakhtin, A., Wu, Y., Miller, A. H., & Riedel, S. (2019). Language models as knowledge bases? EMNLP.
- Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving language understanding by generative pre-training. OpenAI blog.
- Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multitask learners. OpenAI blog, 1(8), 9.
- Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., ... & Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. JMLR.
- Schick, T., & Schütze, H. (2021). Exploiting cloze questions for few shot text classification and natural language inference. EACL.
- Tifrea, A., Bécigneul, G., & Ganea, O. (2019). Poincare Glove: Hyperbolic Word Embeddings. ICLR.
- Touvron, H et al. LLaMA: Open and Efficient Foundation Language Models
- Touvron, H et al. LLaMA 2: Open Foundation and Fine-Tuned Chat Models
- Turian, J.P., Ratinov, L., & Bengio, Y. (2010). Word Representations: A Simple and General Method for Semi-Supervised Learning. ACL.
- Wei, J., et al. (2022). Emergent Abilities of Large Language Models. TMLR.
- Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R., & Le, Q. V. (2019). XLNet: Generalized Autoregressive Pretraining for Language Understanding. NeurIPS.
- Zhou, C., Liu, P., Xu, P., Iyer, S., Sun, J., Mao, Y., Ma, X., Efrat, A., Yu, P., Yu, L., Zhang, S., Ghosh, G., Lewis, M., Zettlemoyer, L., & Levy, O. (2023). LIMA: Less Is More for Alignment.
- Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T. L., Cao, Y., Narasimhan, K. (2023). Tree of Thoughts: Deliberate Problem Solving with Large Language Models. NeurIPS.

Q&A

Tutorial Website: