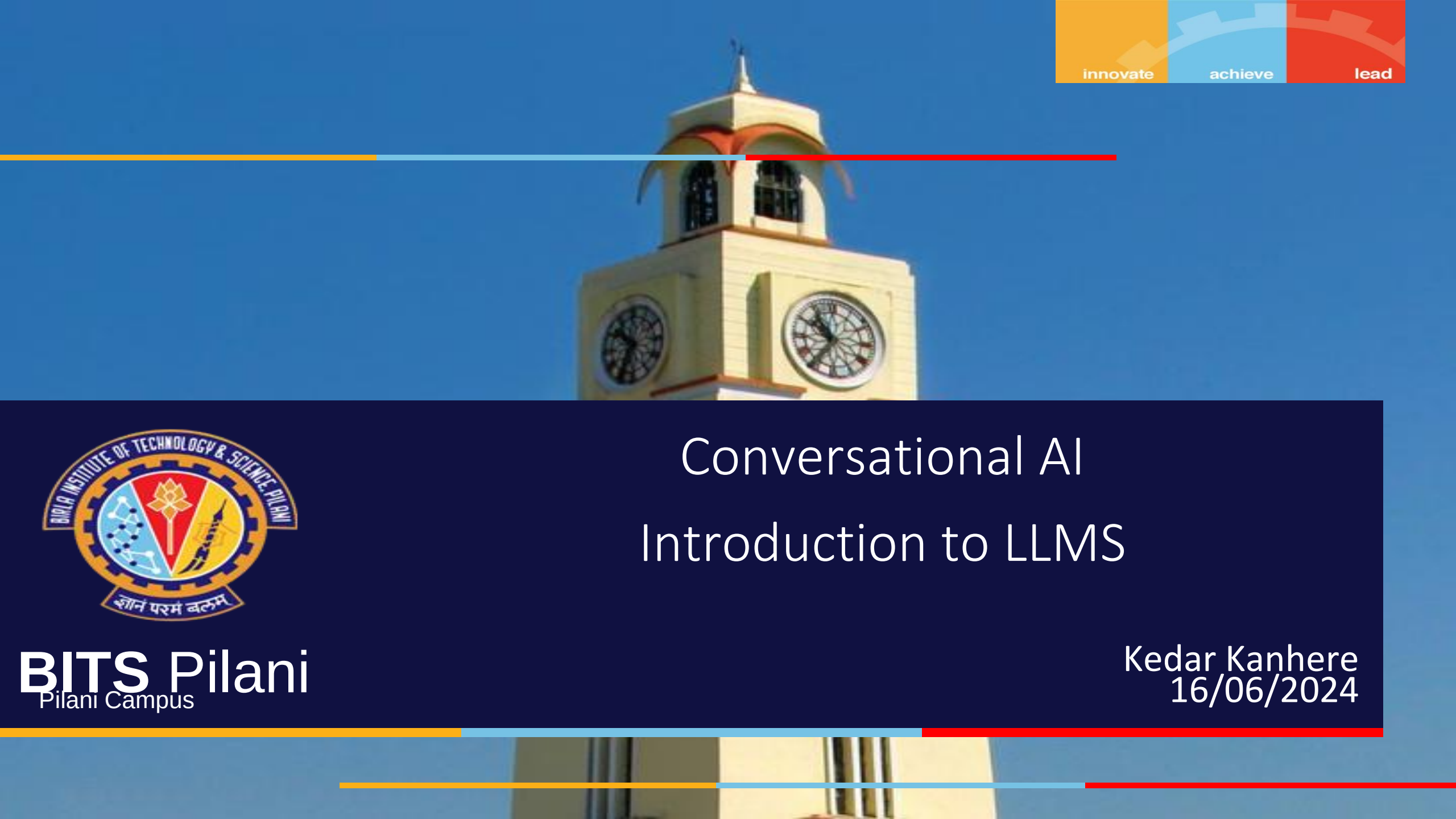


innovate achieve lead

Conversational AI
Introduction to LLMS

Kedar Kanhere
16/06/2024

BITS Pilani
Pilani Campus

Session Content

- Introduction to LLM’s

Language Models

- The classic definition of a language model (LM) is a probability distribution over each token sequence

  $w_1, w_2, \ldots, w_n$, whether it's a good or bad one.

- Sally fed my cat with meat: P(I, feed, my, cat, with, meat) = 0.03,
- My cat fed Sally with meat: P(My, cat, fed, Sally, with, meat) = 0.005,
- fed cat meat my my with: P(fed, cat, meat, my, my, with) = 0.0001

Autoregressive language models

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

Conditional probability

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

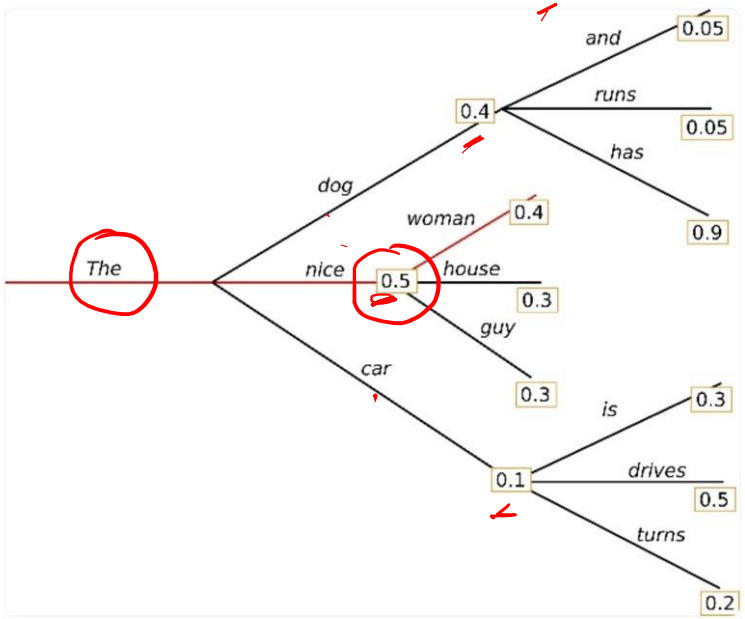
Generation

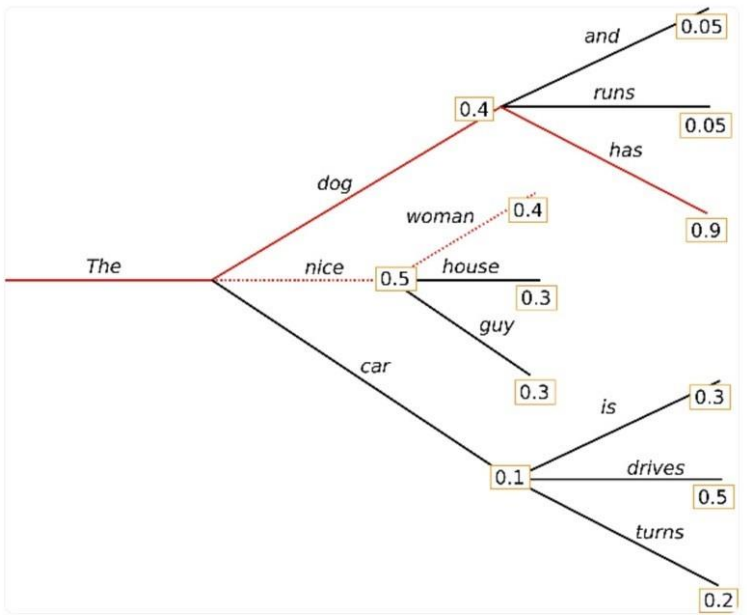
- If we already have a good language model, a given text prompt $w_{[1:n]}$, and we want the model to generate a good sentence completion with the length of L: How to find $w_{[n+1:n+L]}$ with the highest probability?
- Enumerate over all possible combinations?

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

Different Decoding Approaches

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





Language models

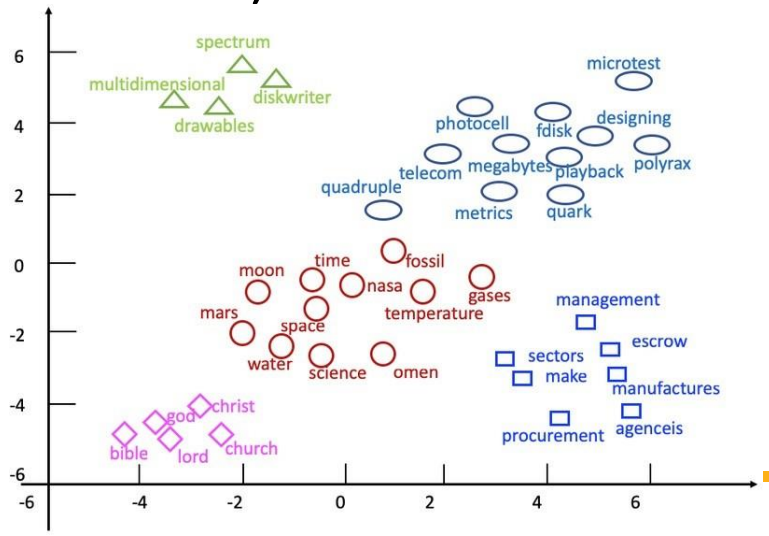
Q: How to train a good language model?

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

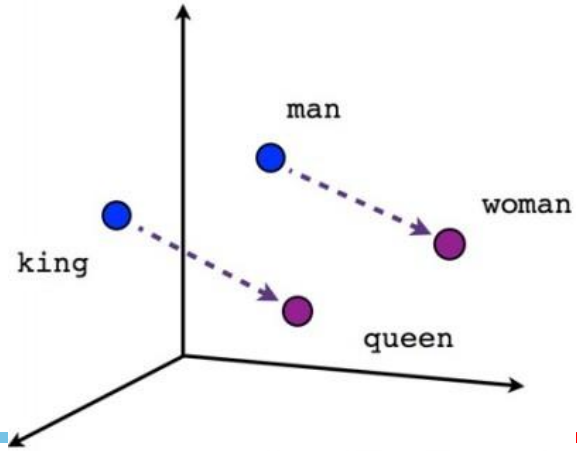
Representing Text in Language Models

A milestone in NLP and ML:

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



Word Similarity



Word Analogy

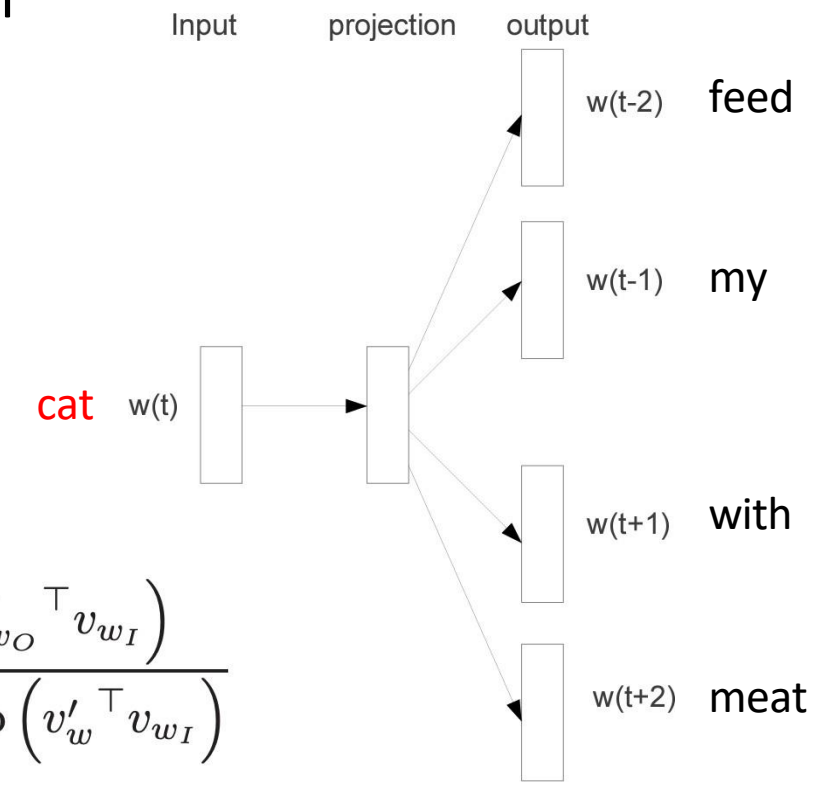
Distributed Representations: Word2Vec

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

Co-occurred words in a local context window

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

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



Considering subwords - fasttext

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

Tri-gram extraction

<where> ⟹ <wh, whe, her, ere, re>

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

Word2Vec probability expression

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

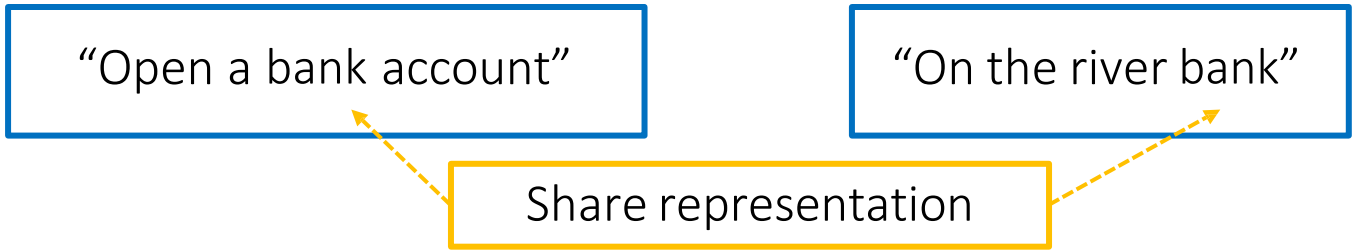
$$\sum_{g \in \mathcal{G}_w} \mathbf{z}_g^{\top} \mathbf{v}_c.$$

N-gram embedding

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

Limitations of embeddings

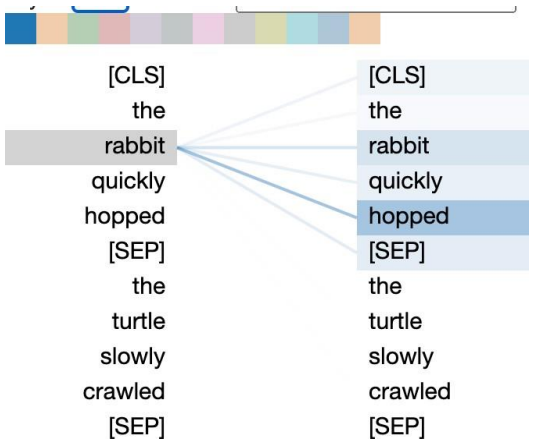
- 1) They are **context-free** embeddings: each word is mapped to only one vector regardless of its context!
  - E.g. “bank” is a polysemy, but only has one representation

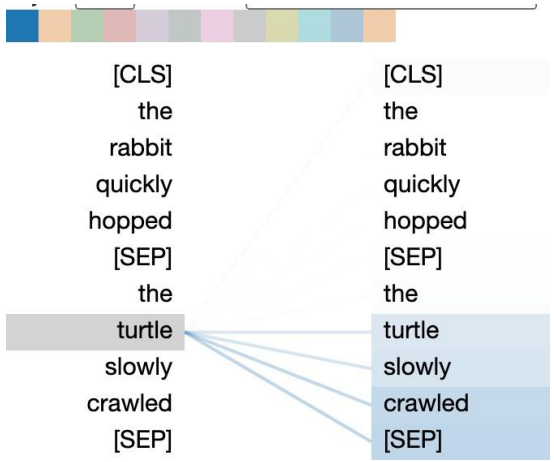


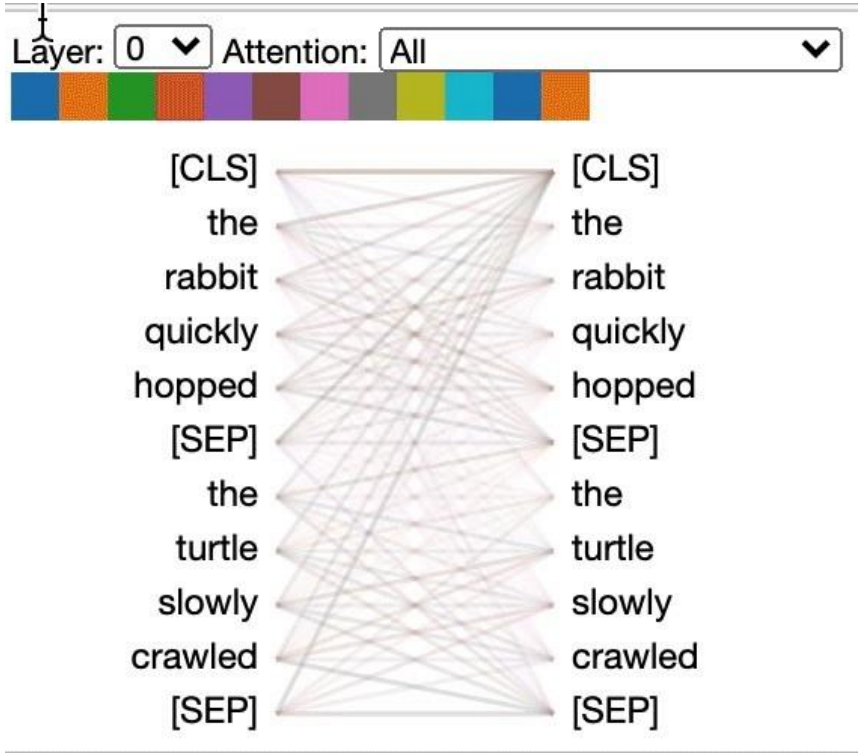
- 2) It does not consider the order of words
- 3) It treats the words in the context window equally

Attention is all you need

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

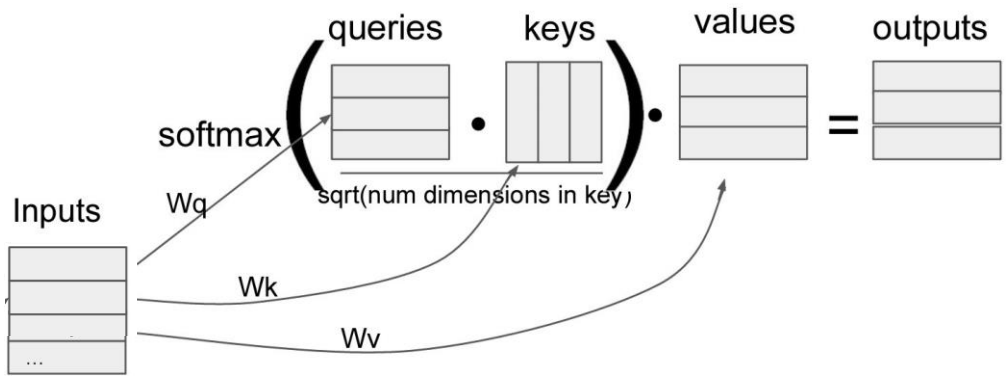




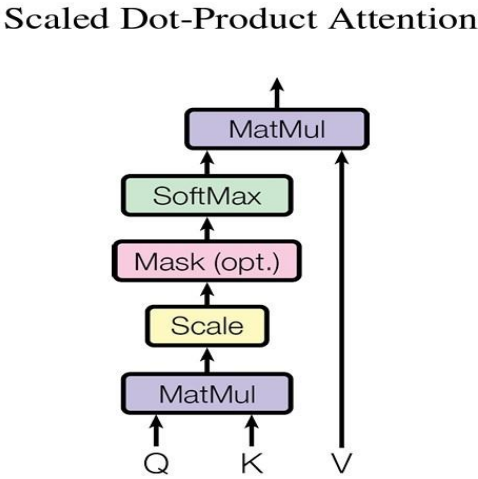


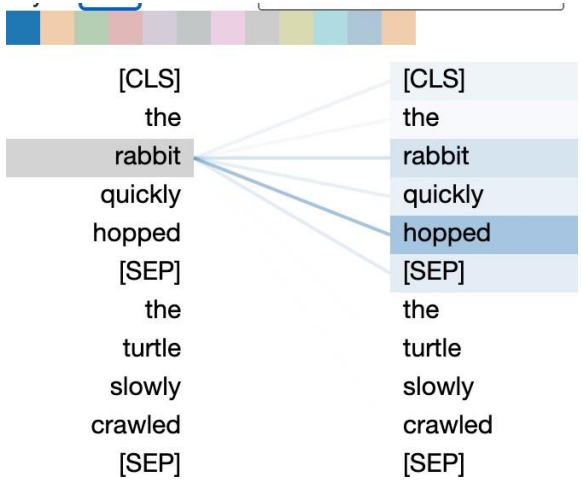
Self-Attention

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



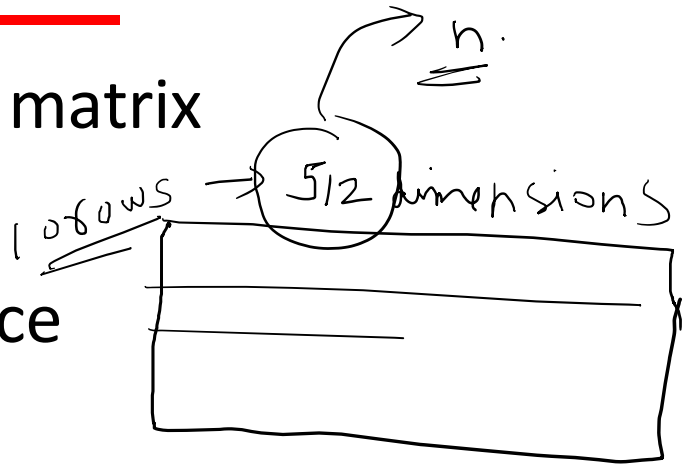
$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$





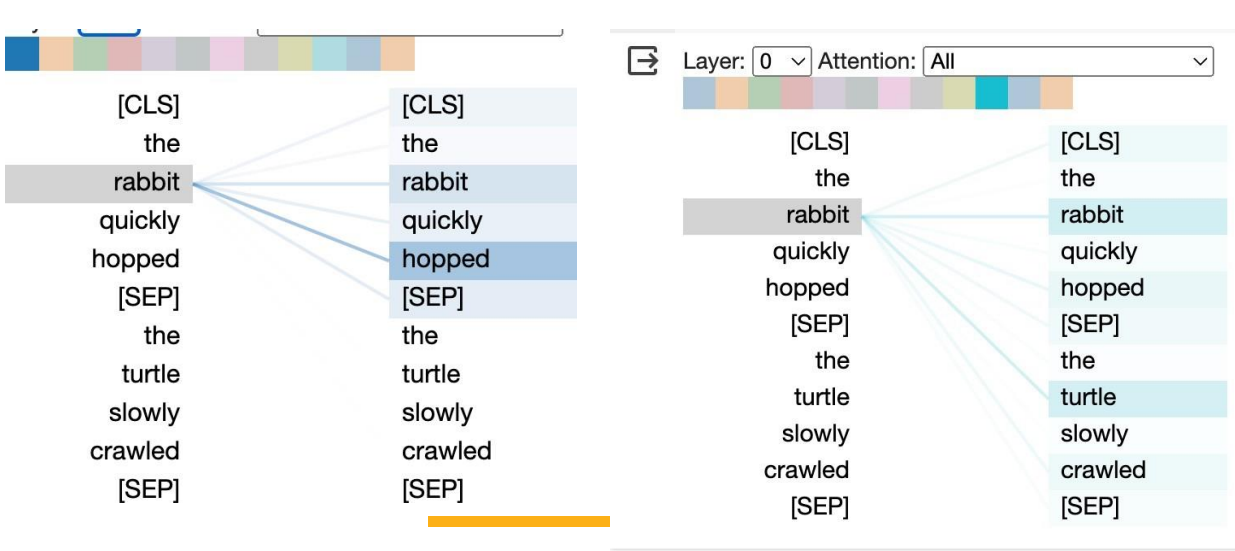
Multi-Head Attention

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

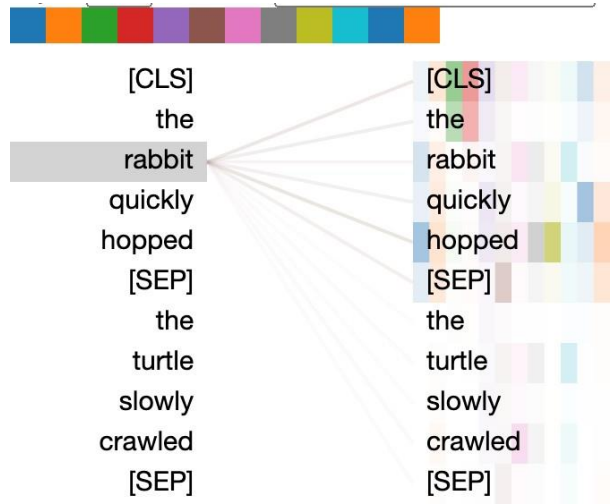


$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_\text{h})W^O$$

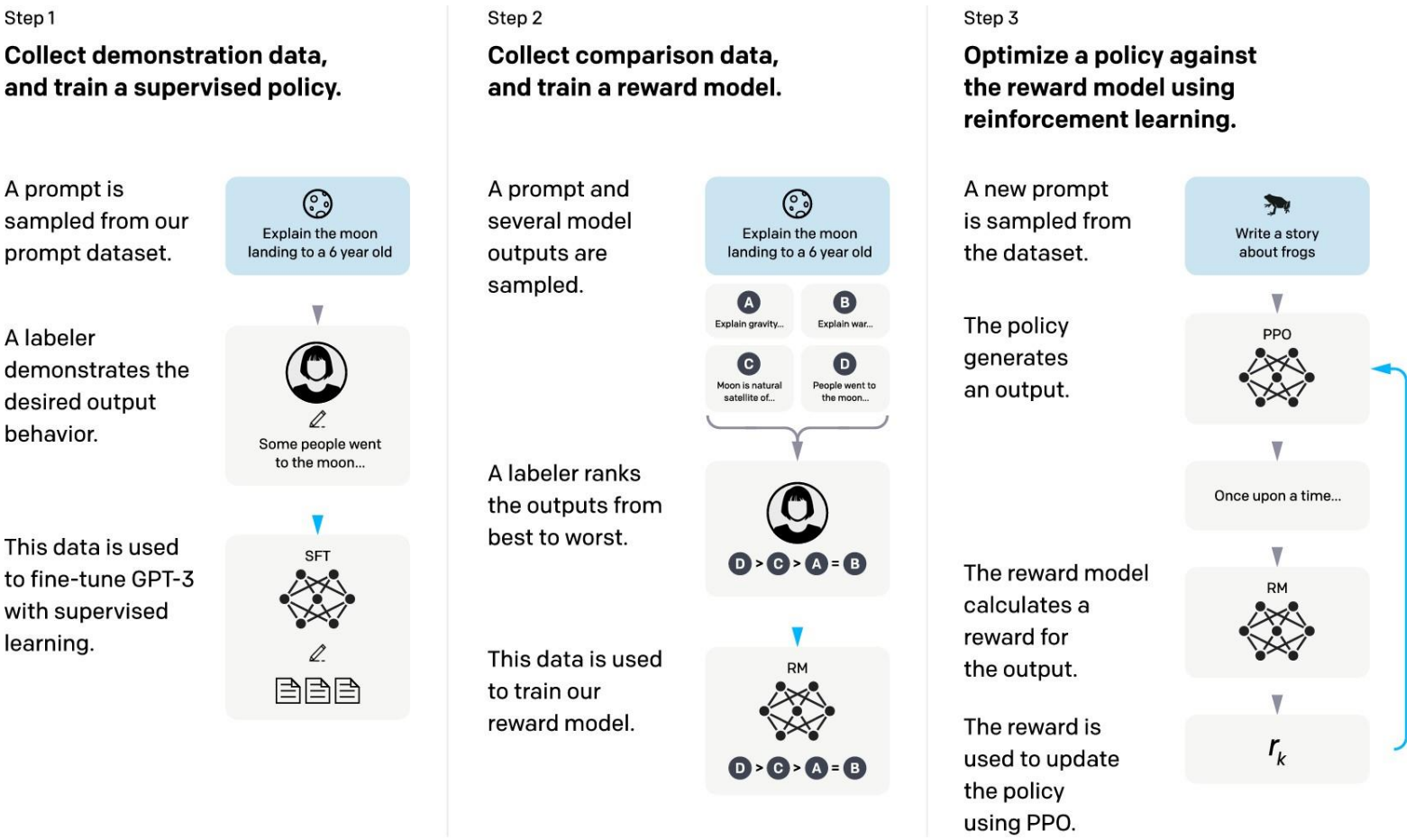
$$\text{where head}_\text{i} = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$



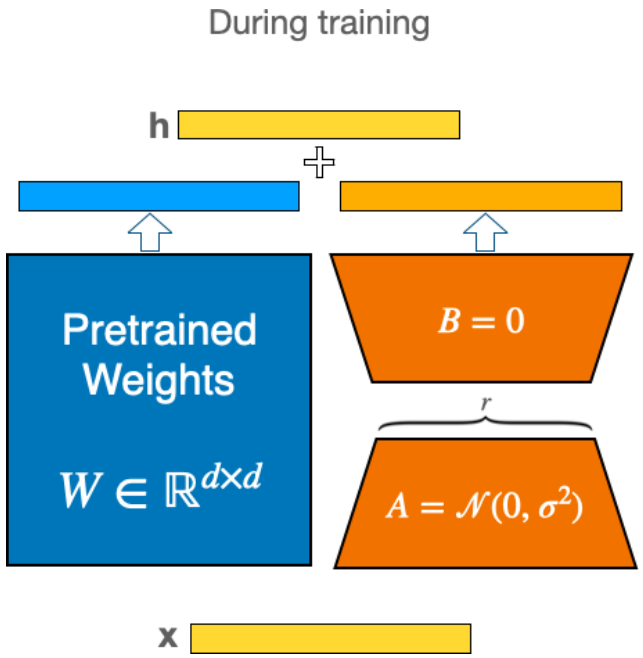
Concatenation



LLM Pre Training Framework



Efficient Fine-Tuning



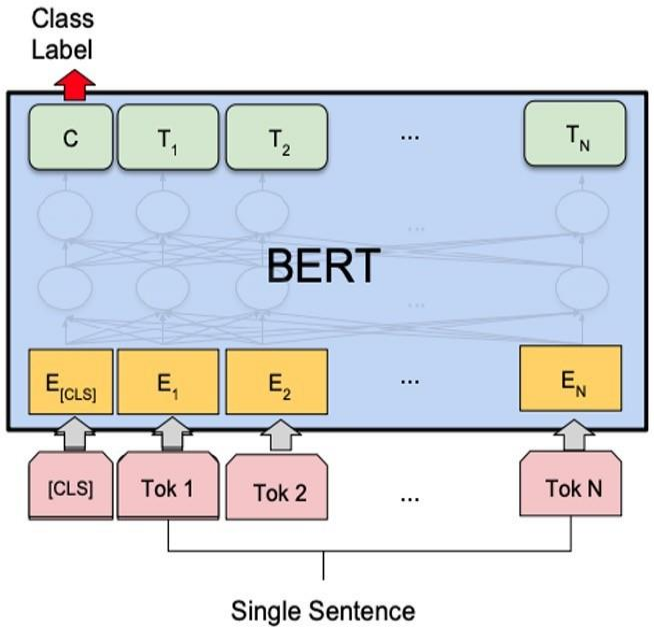
Unsupervised/Self-supervised;
On large-scale general domain corpus

→

Task-specific supervision;
On target corpus

Common NLP Tasks

- Single-sentence classification tasks
  - Text Classification Tasks
    - Input: The bike is too small and I want to return it.
    - Output: <refund, return, check_status>
  - Sentiment Analysis
    - Input: The restaurant is crowded and I waited my food for 30 minutes!
    - Output: <positive, negative>



Common NLP Tasks

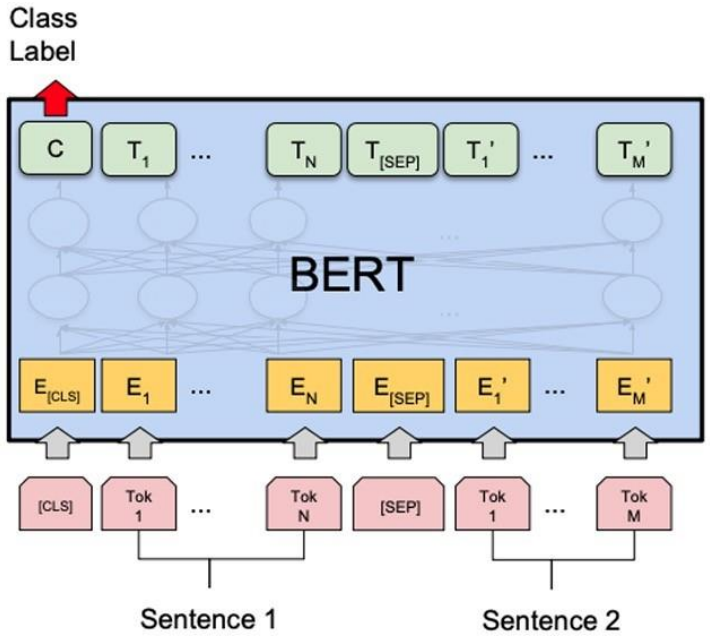
- Sentence-pair classification tasks
  - Sentence entailment
    - Input:

      Sentence 1: Our Large Language Model Course meets on Tuesdays and Thursdays in WashU.

      Sentence 2: There is a large language model course in WashU.

    - Output:

      <entailment, contradiction, neutral>

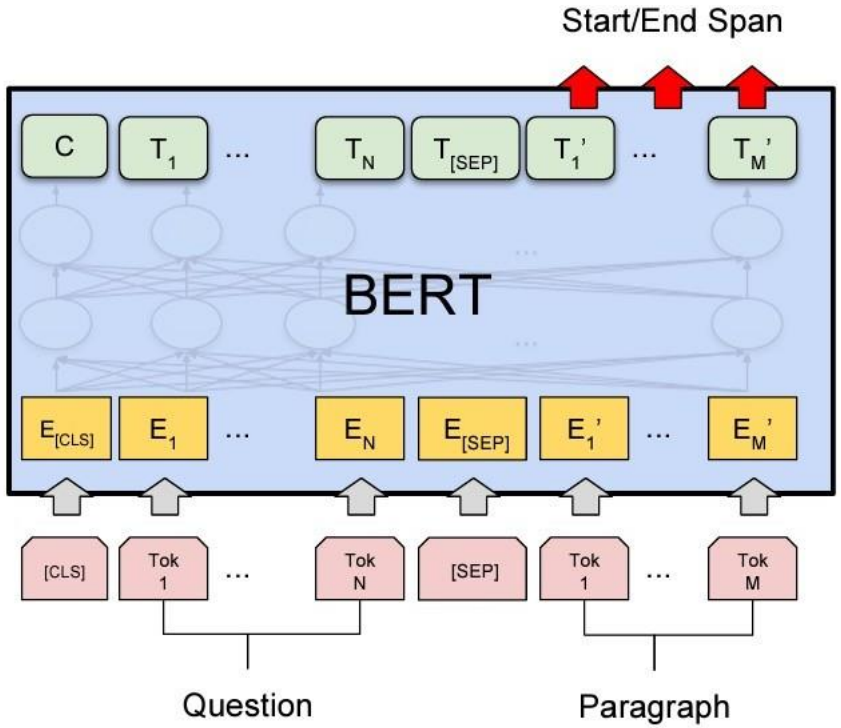


Common NLP Tasks

- Sentence generation tasks
  - Machine Translation
    - Input:

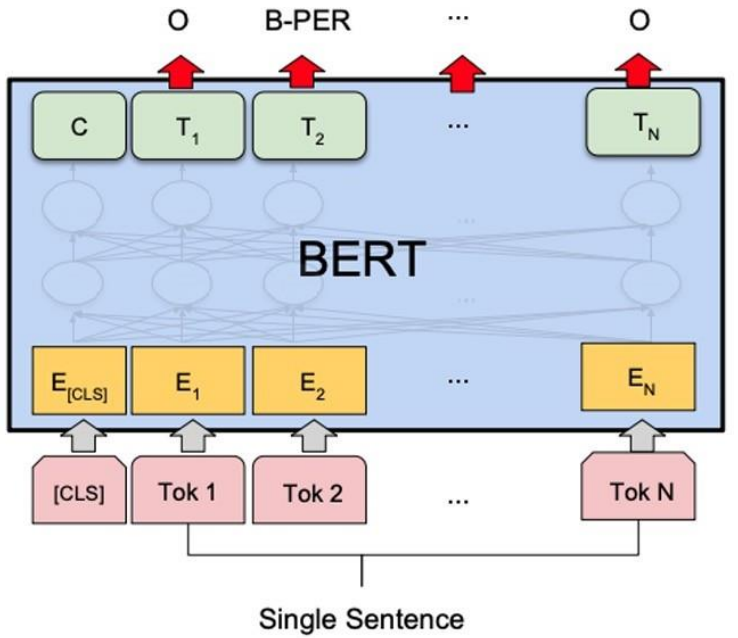
    English: This is good. Germany:
    - Output:

    Das ist gut.



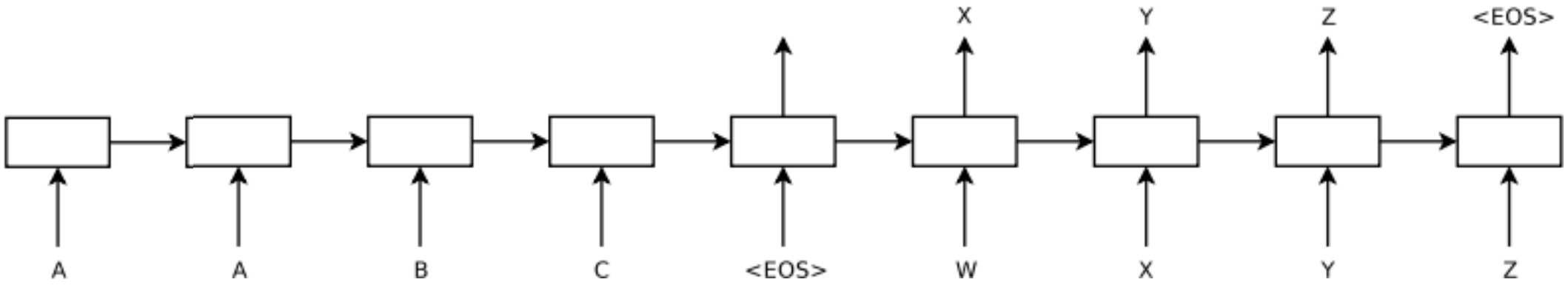
Common NLP Tasks

- Token-level tasks
  - Named Entity Recognition
  - Input: St. Louis is located in the state of Missouri .
  - Output: <Begin-Location> <Inside-location> O O O O O O <Begin-Location> O



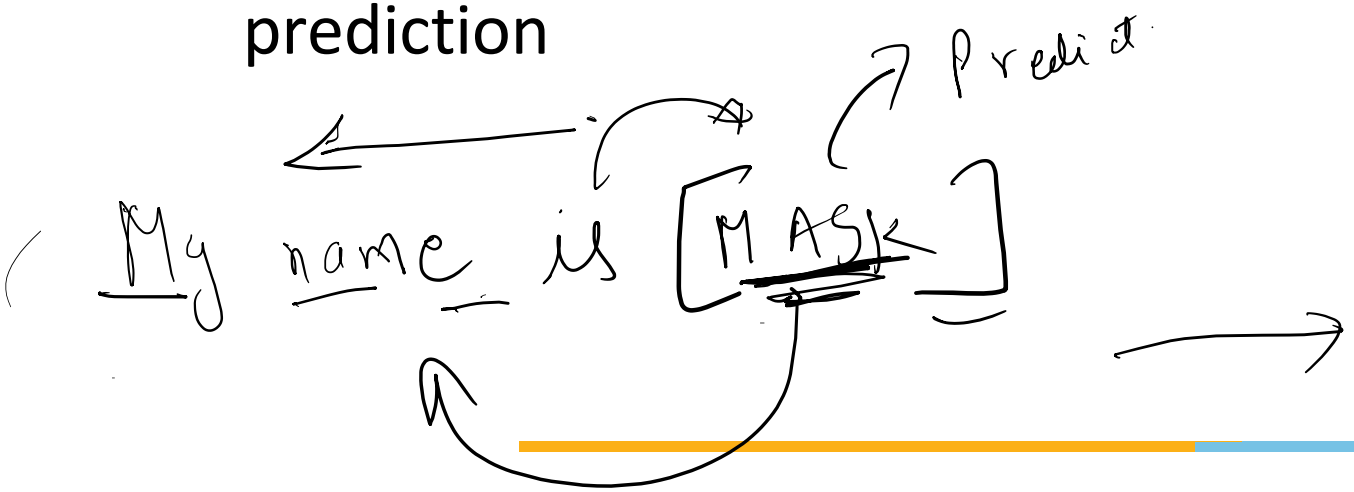
Encoder and Decoder

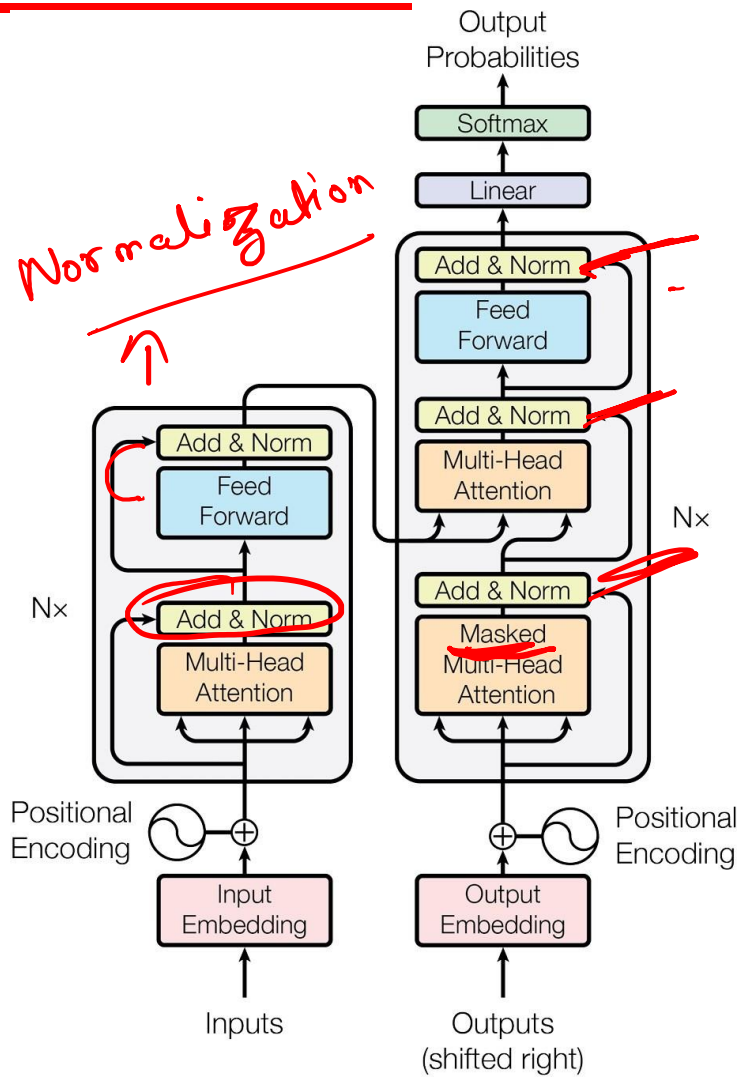
- NLP tasks can be generally decomposed into language understanding and language generation.
- Encoder models are generally used to understand input sentences, and decoder models are generally used to generate sentences.



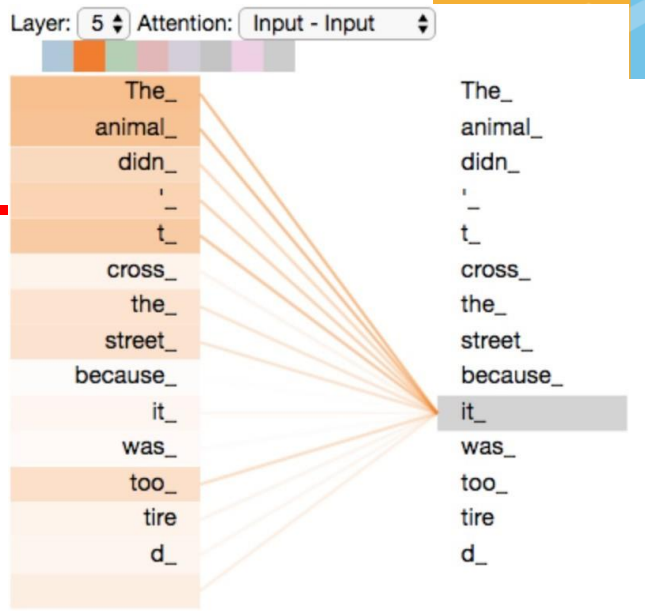
Transformer Model Architecture

- Input Embedding
- Positional Encoding
- 12 Transformer layers
  - 6 encoder layers
  - 6 decoder layers
- Linear + Softmax layer for next word prediction





Encoder Model

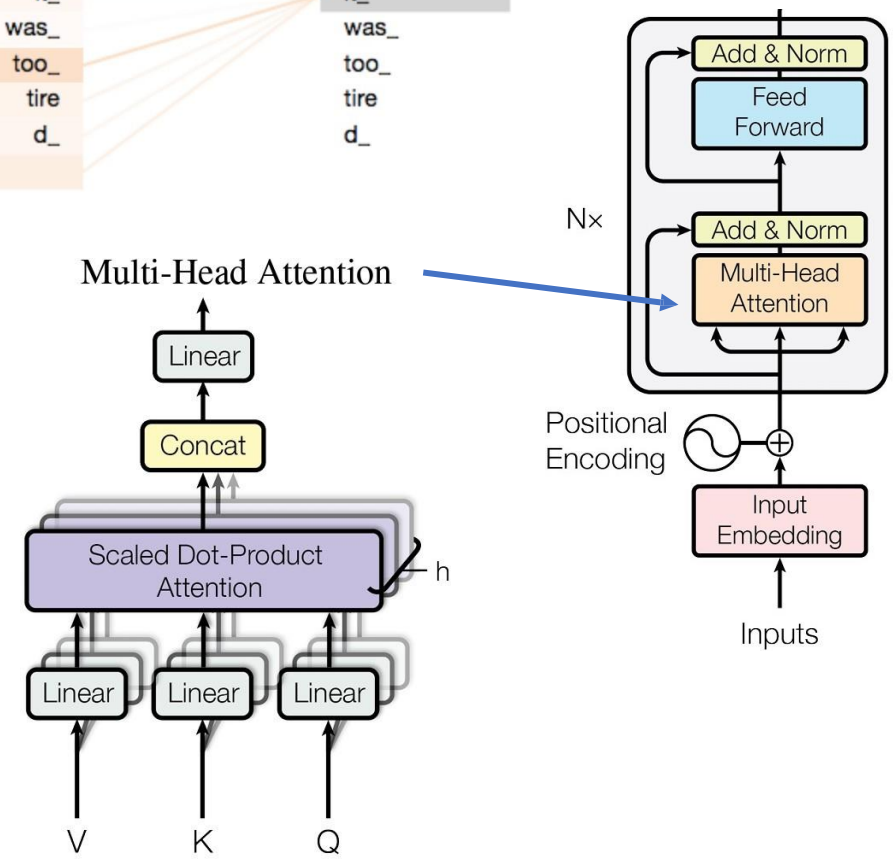


- Multi-head attention layer captures information from different subspaces at different positions

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_\text{h})W^O$$
$$\text{where head}_\text{i} = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

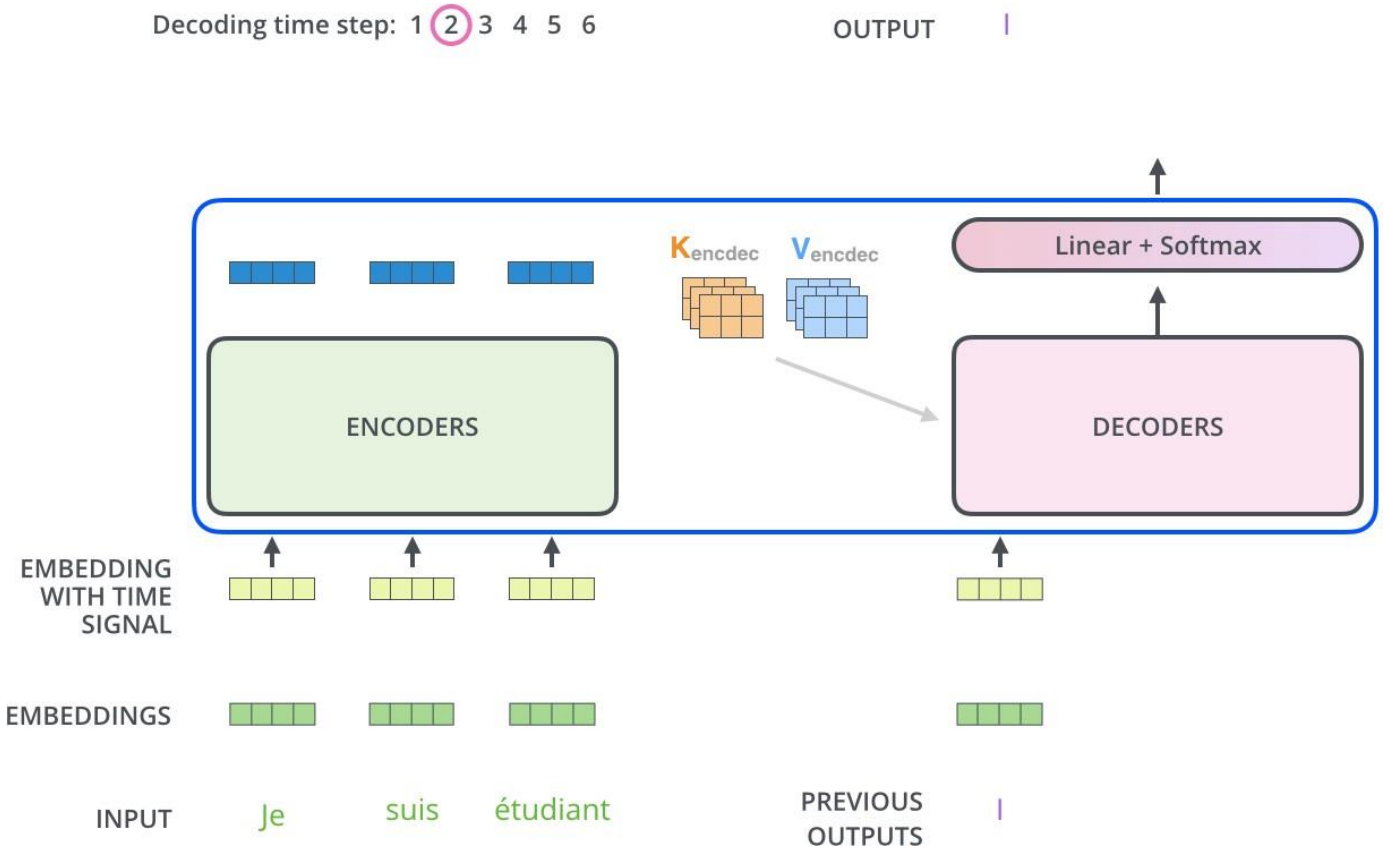
- Feed-forward layer is applied to each token position without interaction with other positions

- Residual connection and layer normalization



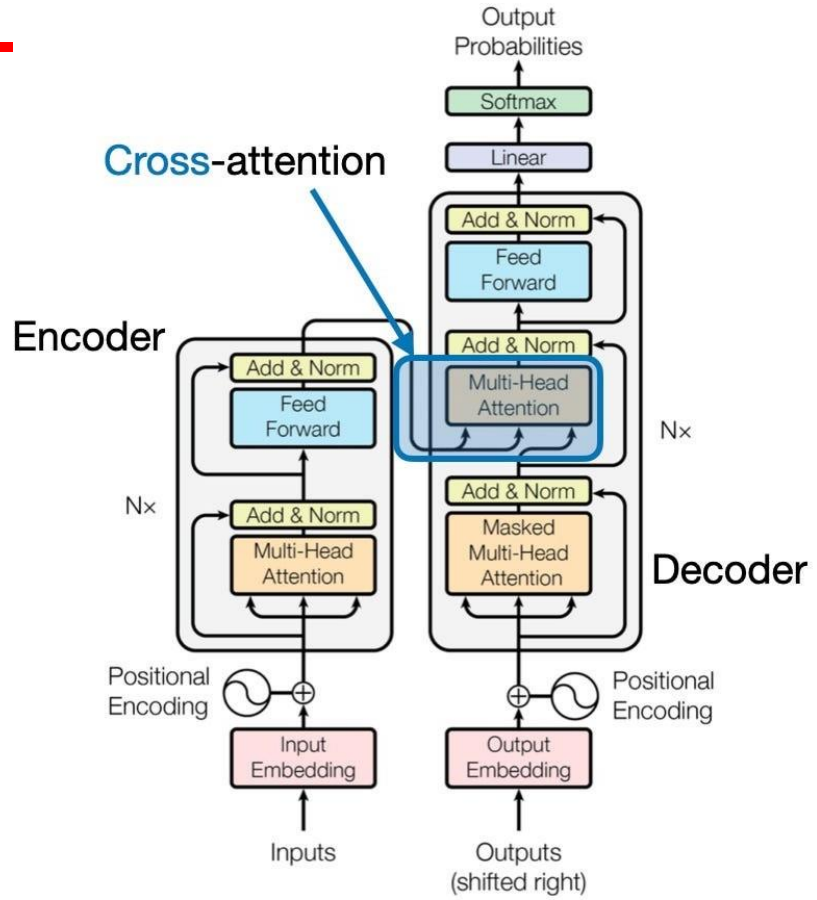
Decoder Model

- Demo from https://jalammar.github.io/illustrated-transformer/



Decoder Model

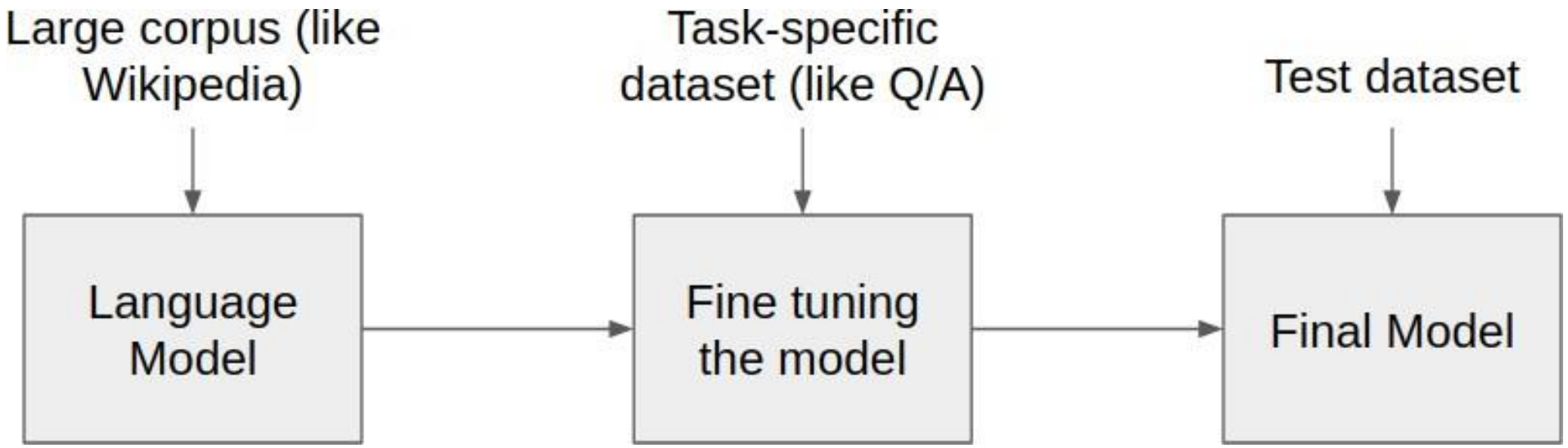
- Multi-head self-attention: only allowed to attend to earlier positions (left side).
  - Q, K, V matrices are both from the previously generated tokens
- Multi-head cross-attention: attend to the input sequence.
  - Q is from the generated tokens
  - K, V matrices are from the input tokens



Source: "Attention Is All You Need" (https://arxiv.org/abs/1706.03762)

Pre-trained Language Models

- "Pretraining": Train deep language models (usually Transformer models) via **self-supervised** objectives on **large-scale general-domain corpora**
- "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



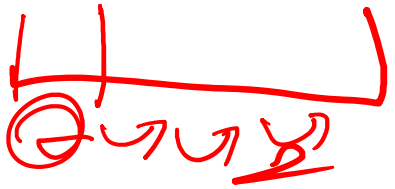
Different Architectures for PLMs

- **Decoder-Only (Unidirectional) PLM** (e.g., GPT): Predict the next token based on previous tokens, usually used for **language generation tasks**
- **Encoder-Only (Bidirectional) PLM** (e.g., BERT, XLNet, ELECTRA): Predict masked/corrupted tokens based on all other (uncorrupted) tokens, usually used for **language understanding/classification tasks**
- **Encoder-Decoder (Sequence-to-Sequence) PLM** (e.g., T5, BART): Generate output sequences given masked/corrupted input sequences, can be used for both **language understanding and generation tasks**

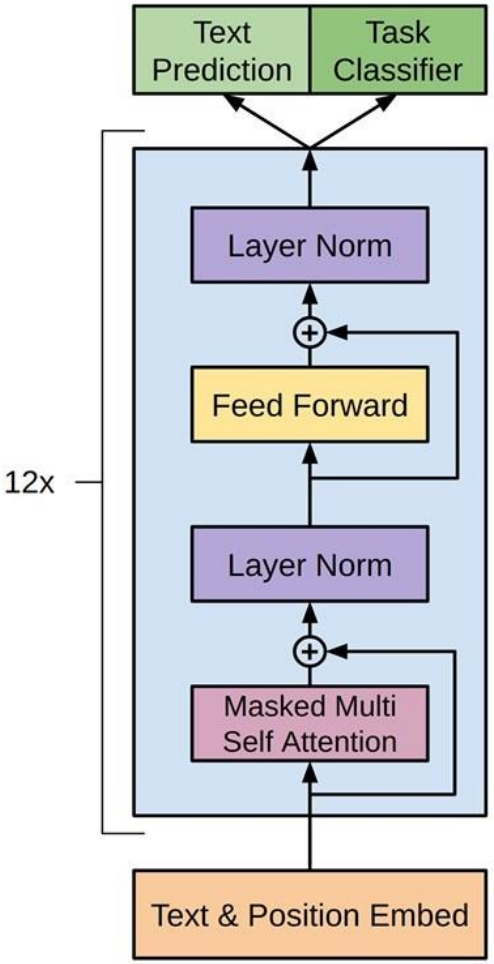
Generative Pretraining (GPT)

- Model Architecture: A multi-layer transformer decoder
- Leverage unidirectional context (usually left-to-right) for next token prediction (i.e., language modeling)

k previous tokens as context



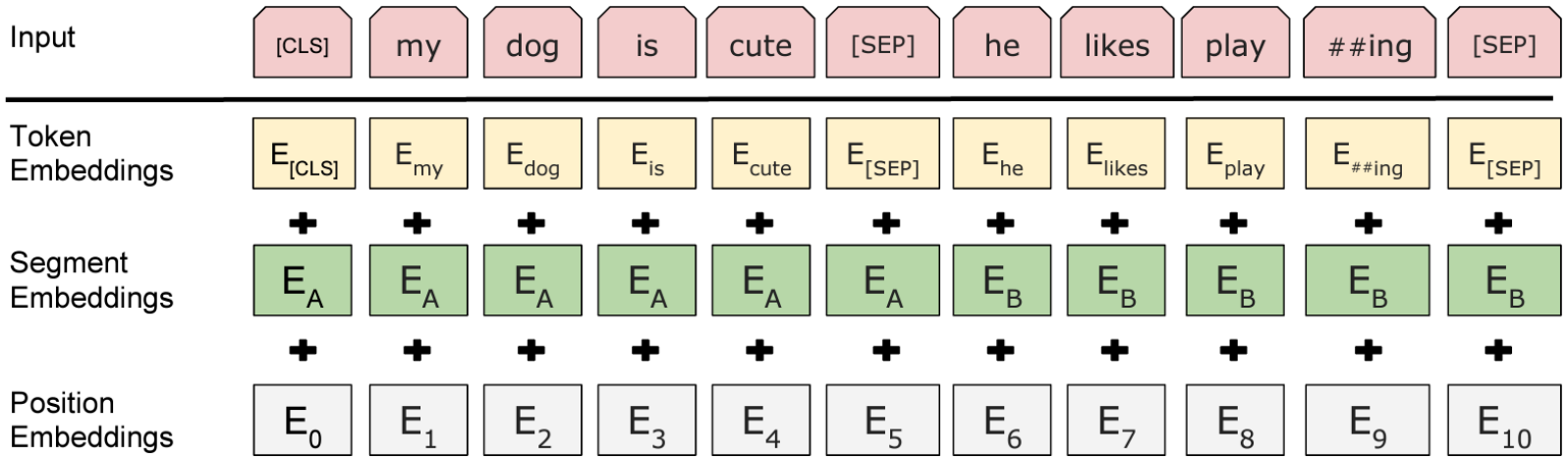
- The Transformer uses **unidirectional** attention masks (i.e., every token can only attend to previous tokens)



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.

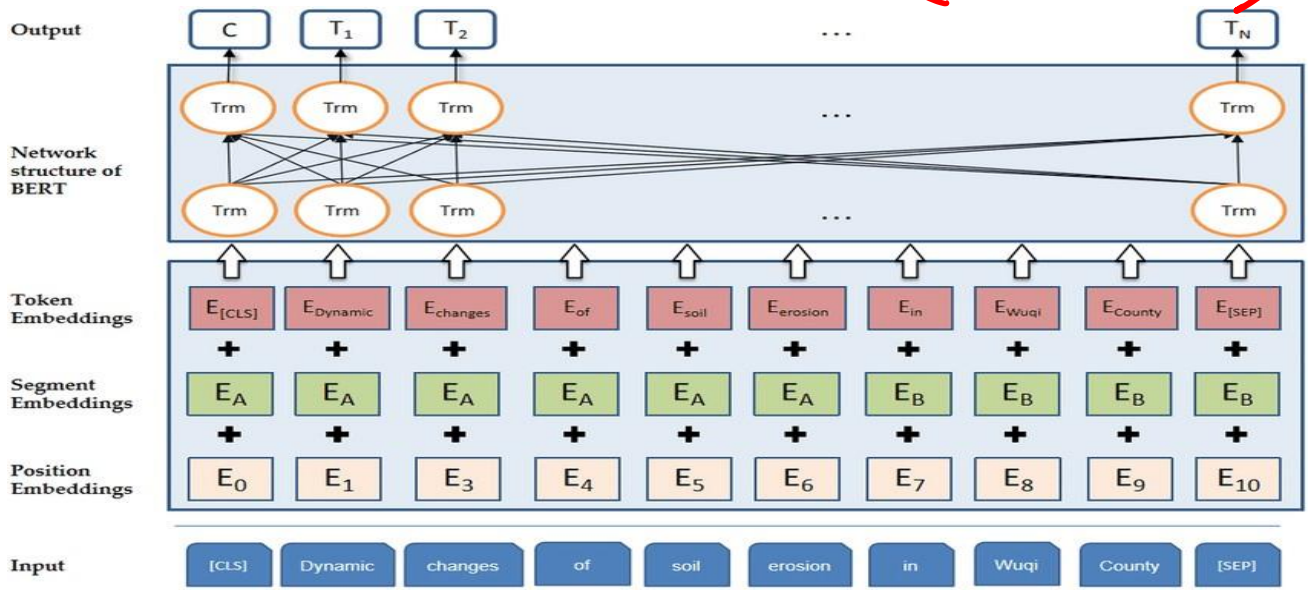
BERT Model Architecture

- Input: Sentence Pairs with special tokens [CLS] and [SEP].
  - Pair-wise tasks: question answering, translation, sentence entailment
  - [CLS]: beginning of a sentence
  - [SEP]: separation of two sentences
- WordPiece embedding



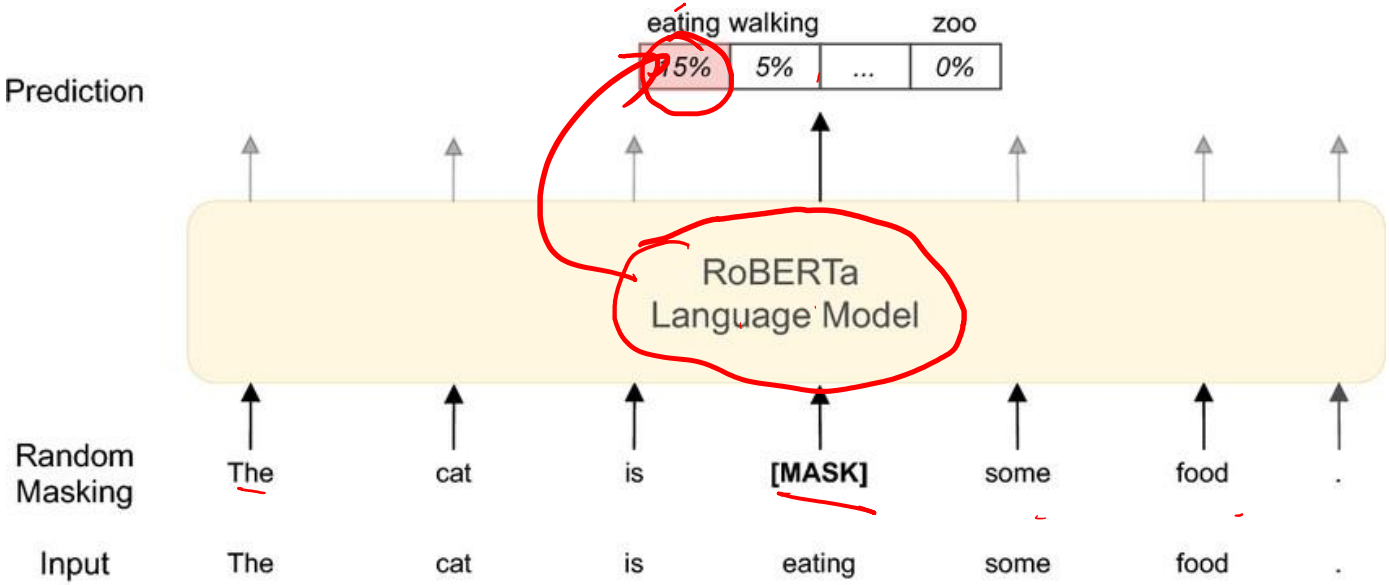
BERT Model Architecture

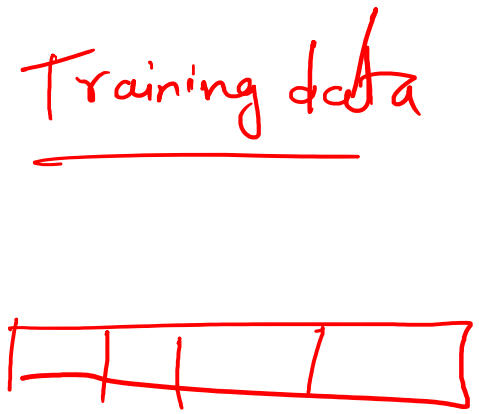
- Multi-layer Transformer encoder model
  - Bert-base: 12 layers, 768 hidden size, 12 attention heads, 110M parameters
  - Bert-large: 24 layers, 1024 hidden size, 16 attention heads, 340M parameters
- Bidirectional: each token can attend to its left and right context for self-attention



BERT Training Objective (I): Masked Language Modelling

- Masked Language Modeling (MLM)
  - Randomly mask a few words in the original sentences.
  - Predict the masked words with its left and right context.
  - Mask ratio: 15%
  - Demo: https://huggingface.co/bert-base-cased



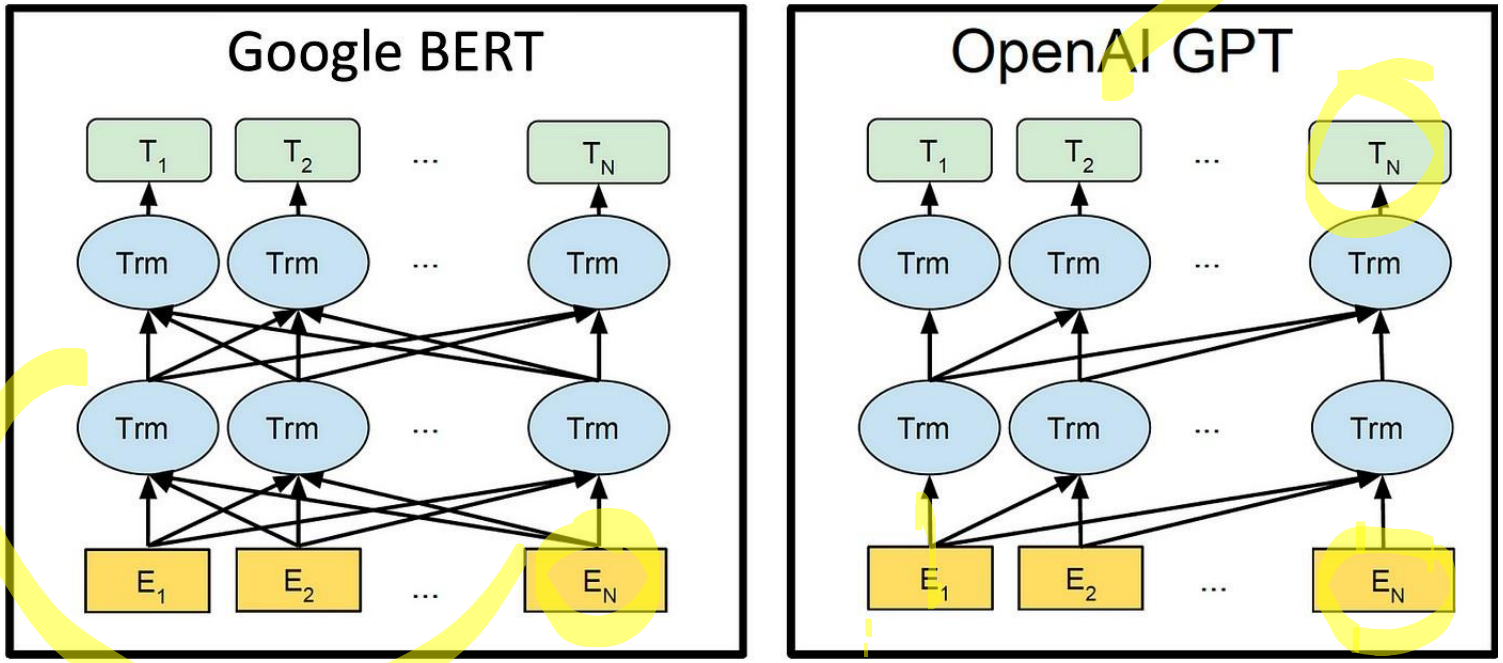


BERT Training Objective (I): Masked Language Modelling

- Discrepancy between pre-training and fine-tuning: There is no [MASK] word in a test set!
- 80-10-10 replacing rule for the sampled 15% tokens.
  - 80%: Replace the original word with [MASK] token
    - cat eating food -> cat [MASK] food
  - 10%: Replace the original word with a random token
    - cat eating food -> cat paper food
  - 10%: Do not replace the original word
    - cat eating food -> cat eating food

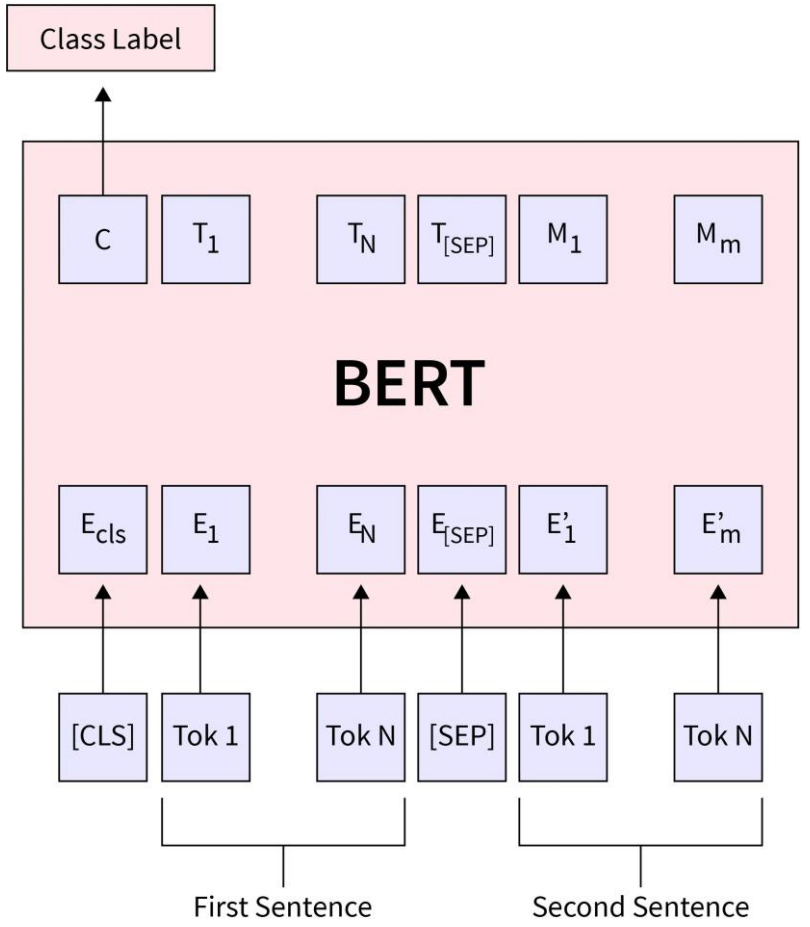
Comparison with GPT-2 Model

- Training objective: MLM prediction vs. left-to-right token prediction



BERT Training Objective (II): Next Sentence Prediction

- Next Sentence Prediction (NSP)
  - Predict whether Sentence B is the next sentence of Sentence A.
  - Positive samples: two contiguous sentences in the corpus.
  - Negative samples: sample another sentence for sentence A.
  - Class Labels: <is_next, not_next>

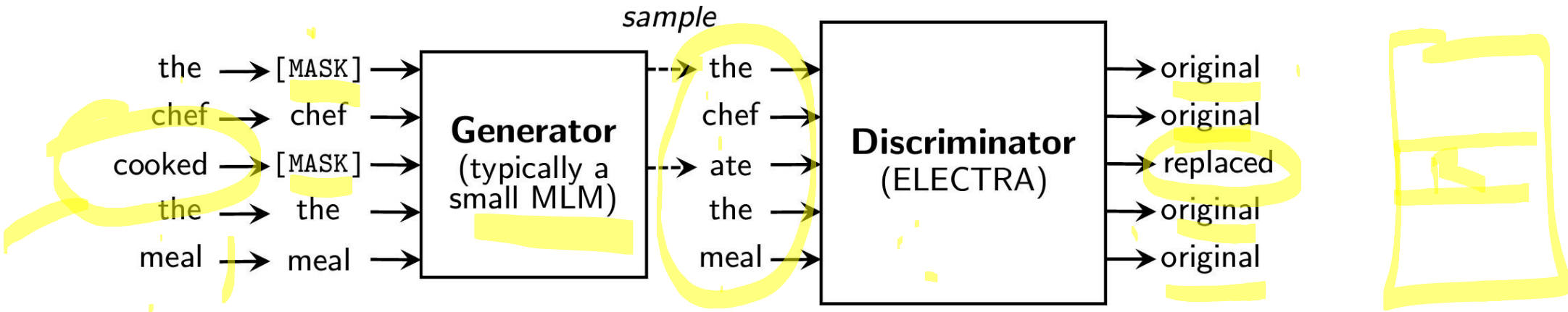


Variants of BERT Model

- RoBERTa (RoBERTa: A Robustly Optimized BERT Pretraining Approach. Liu et al. 2019)
  - Training the model longer on more data with bigger batches
  - Remove the next sentence prediction objective
  - Dynamically change the [MASK] patterns in each epoch

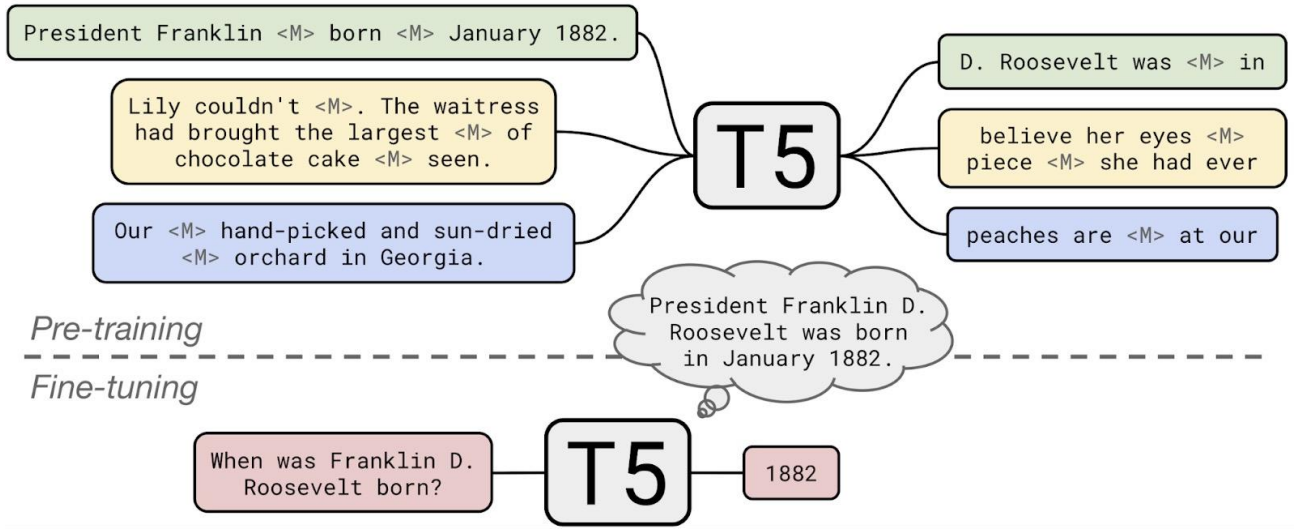
Variants of BERT Model

- ELECTRA (ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators. Clark et al. 2020)
  - Replaced token detection by corrupting text sequences with an auxiliary MLM
  - Works better than BERT because the input text for ELECTRA does not contain [MASK] tokens (no discrepancy between training and test data)



T5 Model

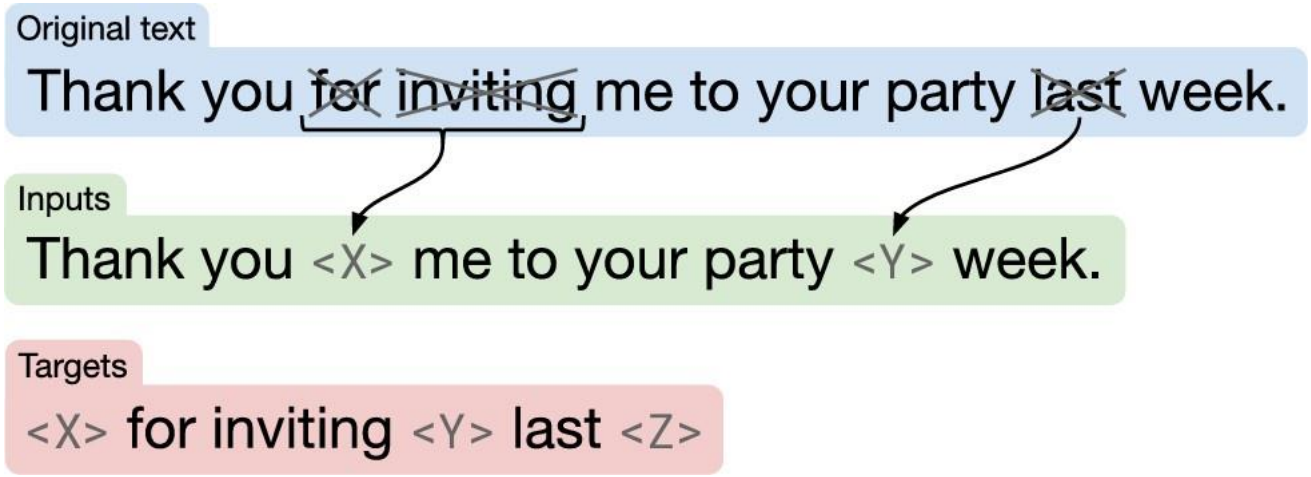
- How to predict a span of masked tokens within a sentence?
- BERT model requires the number of [MASK] token to be given in prior, while GPT models are causal left-to-right models
- T5: **T**ext-**t**o-**T**ext **T**ransfer **T**ransformer (parameters: 60M~11B)



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.

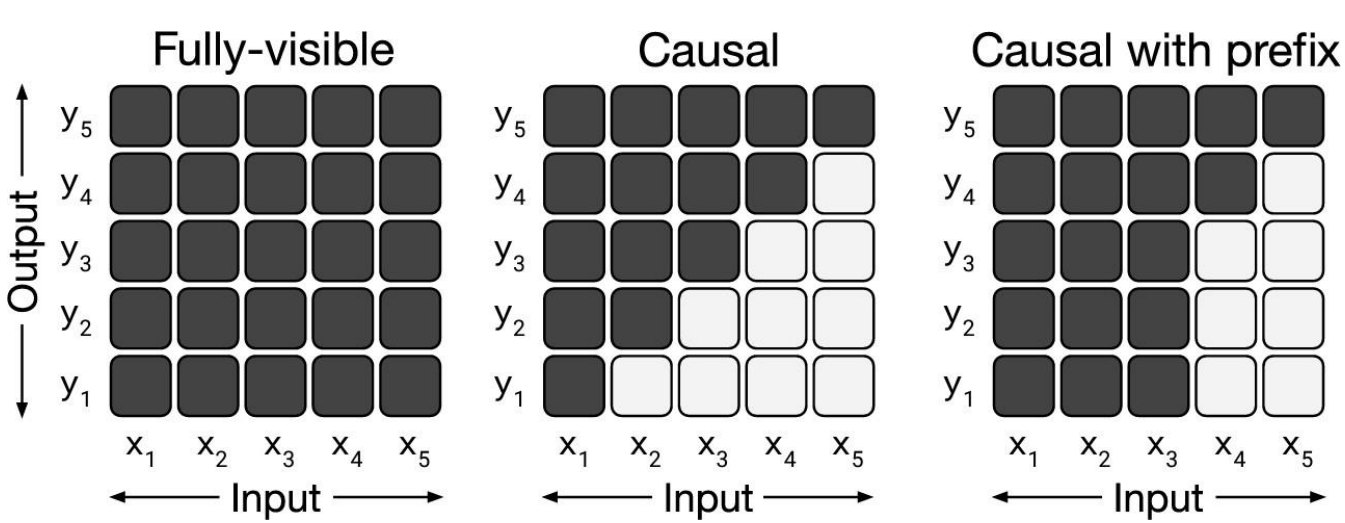
Training of T5 Model

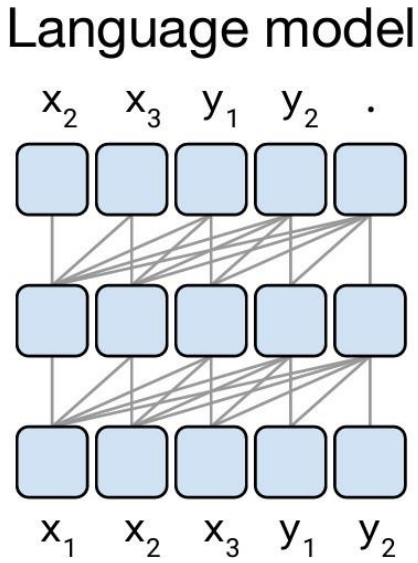
- Pretraining: Mask out spans of texts; generate the original spans
- Fine-Tuning: Convert every task into a sequence-to-sequence generation problem

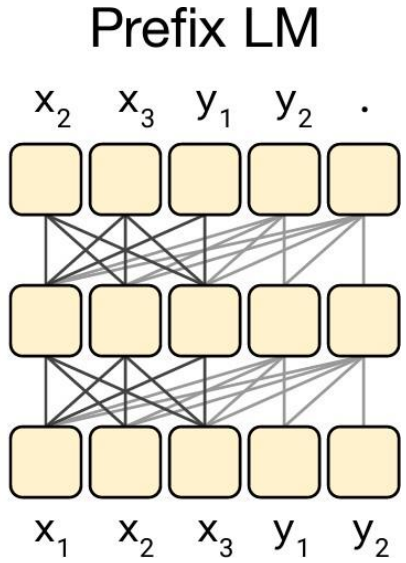


T5 Attention

- A “fully-visible” attention mechanism is placed at the input sequence.
  - Input Sequence:
    - translate English to German : That is good . target :
  - Target Output:
    - Das ist gut .

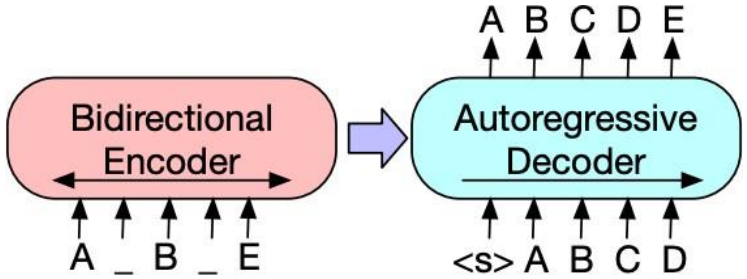




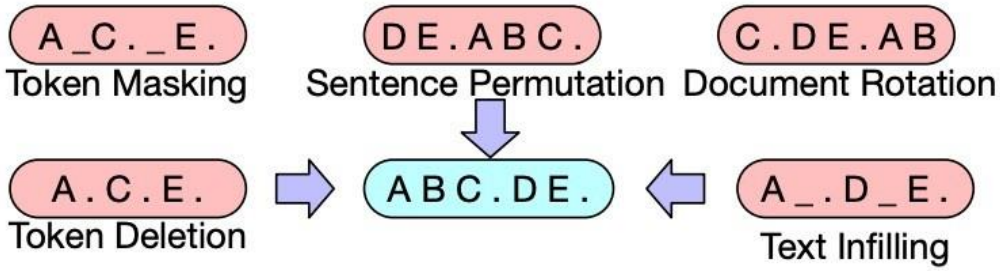


BART Model

- 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



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.

Scaling up Language Models

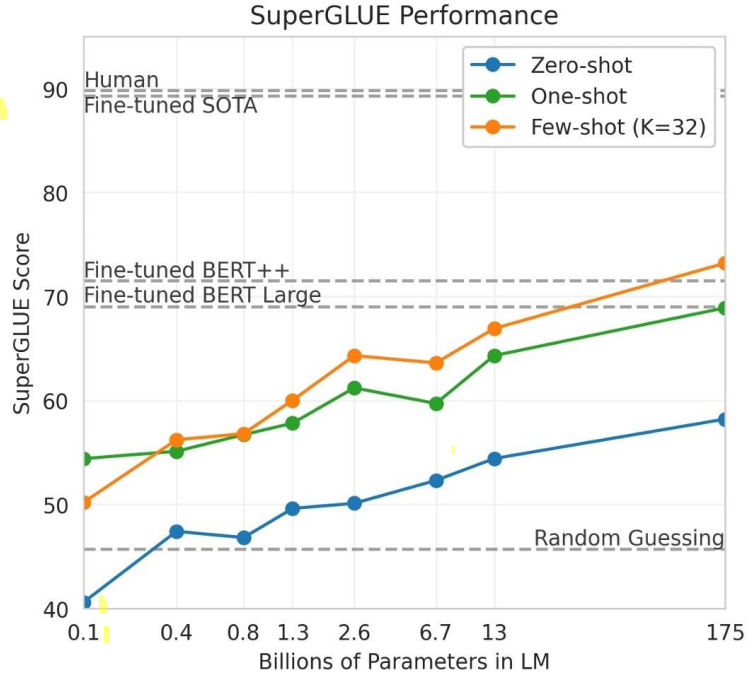
- GPT-2 model size: 1.5 billion parameters
- Pre-trained model can be very, very large (GPT-3 has 175 billion parameters!) and have very strong text generation abilities.

Model Name	n_{params}	n_{layers}	d_{model}	n_{heads}	d_{head}	Batch Size	Learning Rate
GPT-3 Small	125M	12	768	12	64	0.5M	6.0×10^{-4}
GPT-3 Medium	350M	24	1024	16	64	0.5M	3.0×10^{-4}
GPT-3 Large	760M	24	1536	16	96	0.5M	2.5×10^{-4}
GPT-3 XL	1.3B	24	2048	24	128	1M	2.0×10^{-4}
GPT-3 2.7B	2.7B	32	2560	32	80	1M	1.6×10^{-4}
GPT-3 6.7B	6.7B	32	4096	32	128	2M	1.2×10^{-4}
GPT-3 13B	13.0B	40	5140	40	128	2M	1.0×10^{-4}
GPT-3 175B or "GPT-3"	175.0B	96	12288	96	128	3.2M	0.6×10^{-4}

Performance of Zero-Shot/Few-Shot GPT-3

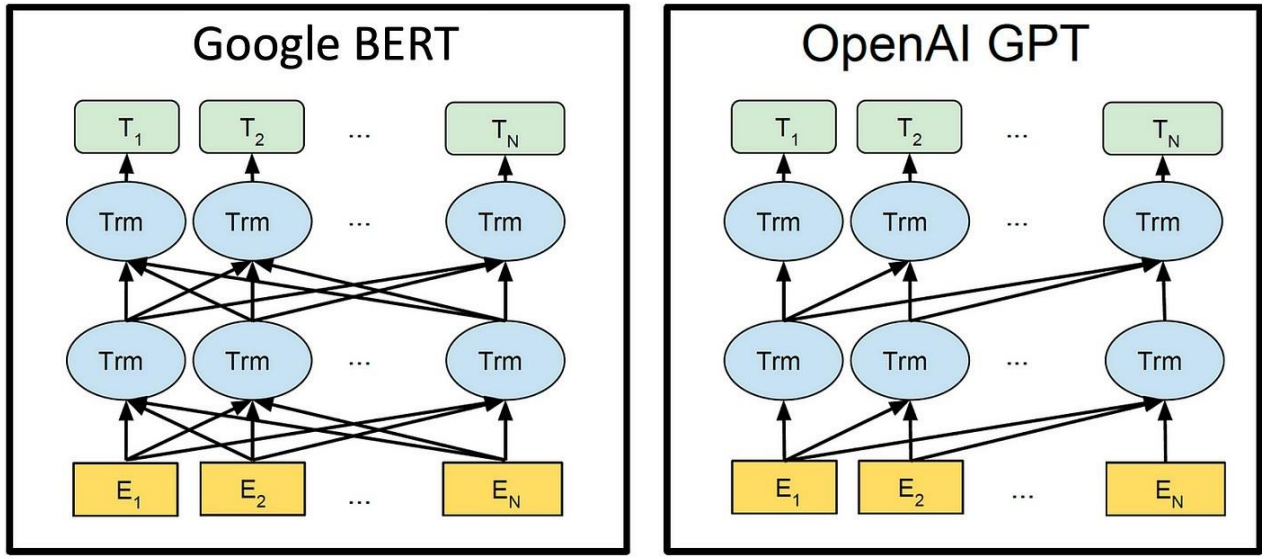
	SuperGLUE Average	BoolQ Accuracy	CB Accuracy	CB F1	COPA Accuracy	RTE Accuracy
Fine-tuned SOTA	**89.0**	**91.0**	**96.9**	**93.9**	**94.8**	**92.5**
Fine-tuned BERT-Large	69.0	77.4	83.6	75.7	70.6	71.7
GPT-3 Few-Shot	71.8	76.4	75.6	52.0	92.0	69.0

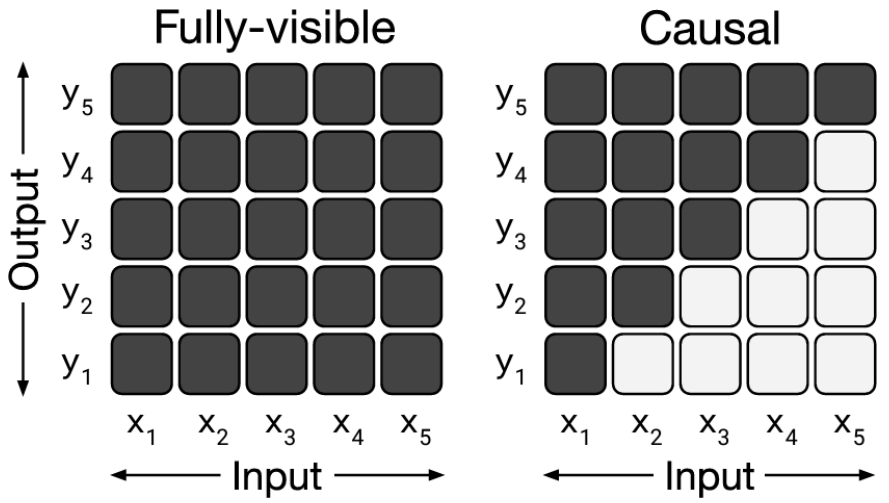
	WiC Accuracy	WSC Accuracy	MultiRC Accuracy	MultiRC F1a	ReCoRD Accuracy	ReCoRD F1
Fine-tuned SOTA	**76.1**	**93.8**	**62.3**	**88.2**	**92.5**	**93.3**
Fine-tuned BERT-Large	69.6	64.6	24.1	70.0	71.3	72.0
GPT-3 Few-Shot	49.4	80.1	30.5	75.4	90.2	91.1



Discussion Question

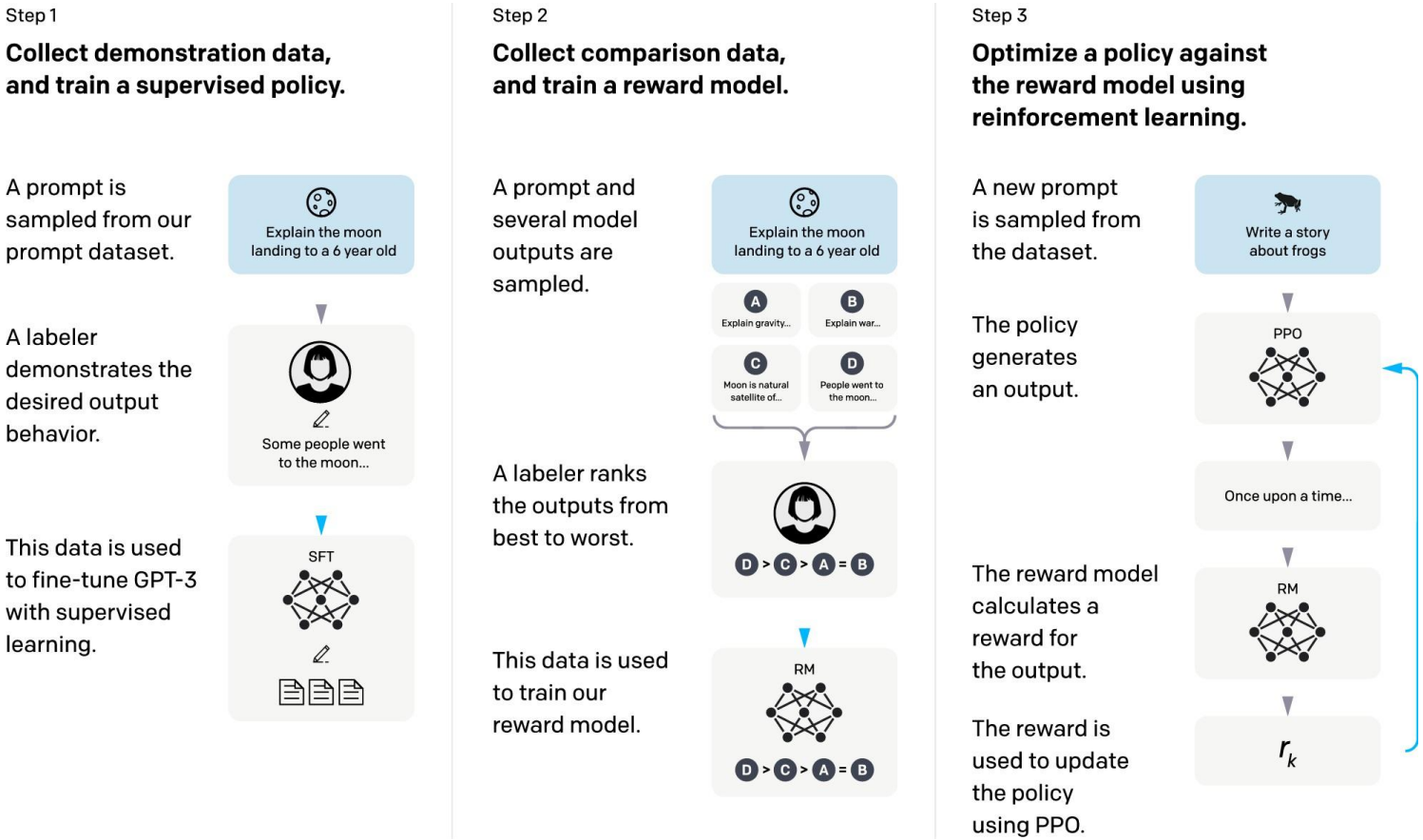
- There has been fewer successful attempts to scaling up BERT-based bidirectional models (e.g., 100x) than unidirectional models. Why does unidirectional model has better scaling performance than bidirectional models?



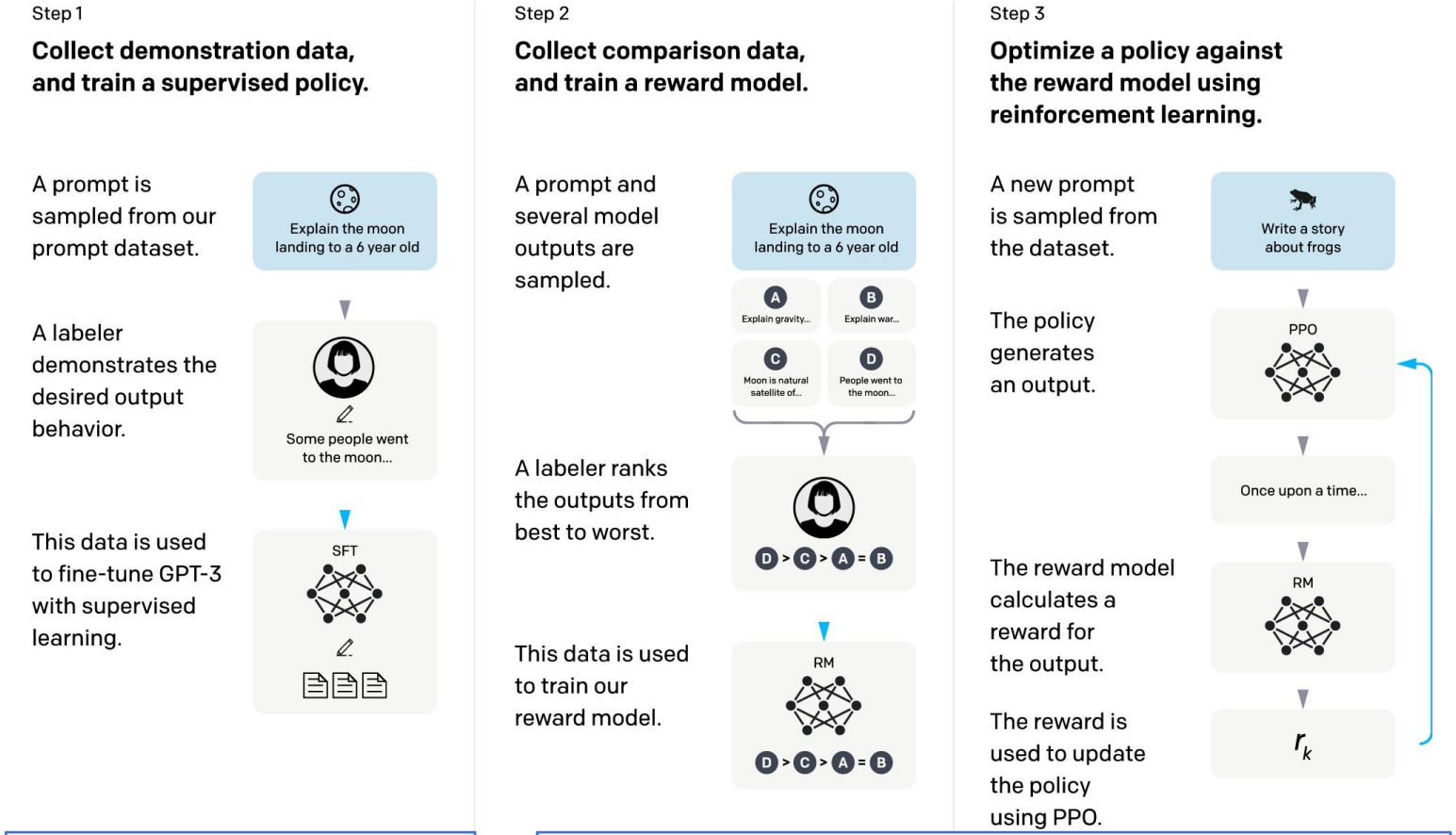


Large Language Model Pre-training Framework

- ChatGPT training procedure
  - Self-supervised pre-training
  - Supervised training on pairs of human-written data (Step 1)
  - Model generate multiple outputs for a prompt, train a reward model on human-labeled ranking list (Step 2)
  - Optimize the language model with the trained reward model (Step 3)



Large Language Model Pre-training Framework



Instruction-Tuning

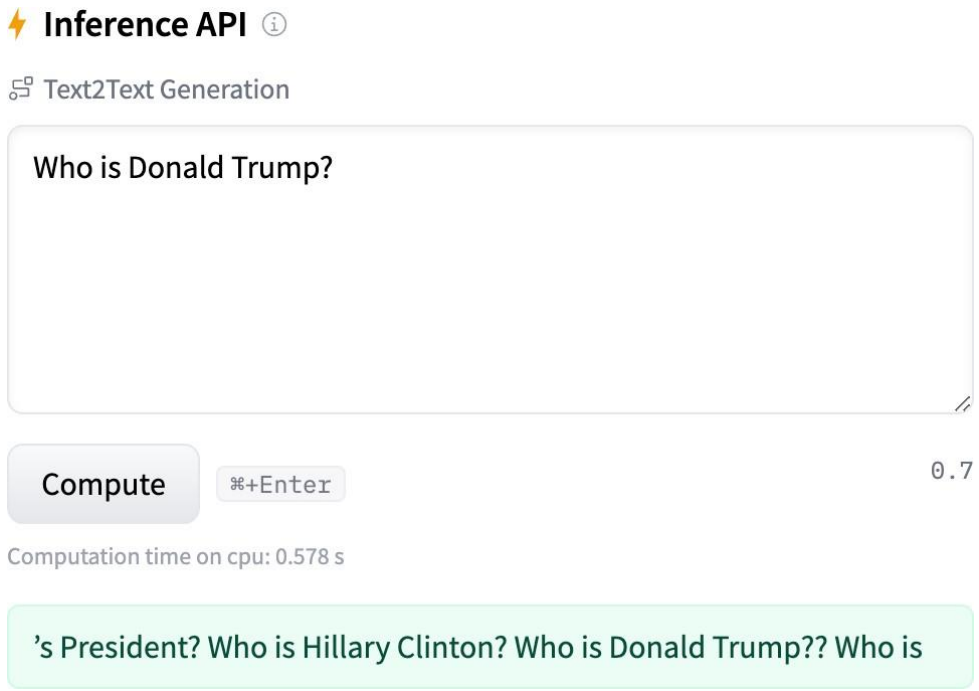
Reinforcement Learning from Human Feedback

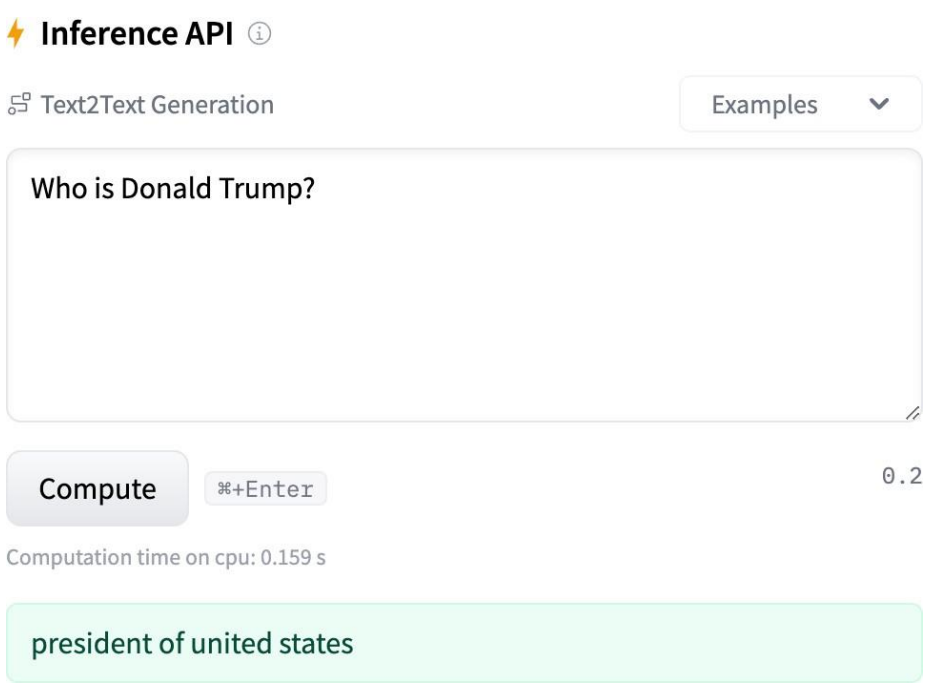
LLaMA: Open and Efficient Foundation Language Models

- ChatGPT (175B): **closed-source/proprietary models** where we don't know about their pre-training corpus, and don't have access to their pre-trained weights, and can only do inference or training through APIs
- LLaMA (7B/13B/33B/65B): **open-source models** where the pre-trained corpus is transparent and we have full access to their pre-trained weights
  - Pre-trained Corpus: English CommonCrawl, C4, Github, Wikipedia, Gutenberg and Books3, ArXiv, Stack Exchange.
  - Smaller models allow researchers with limited computing resources to understand how and why these language models work

Language Models and User intents

- Language models pre-trained on large corpus not necessarily aligned with user intents

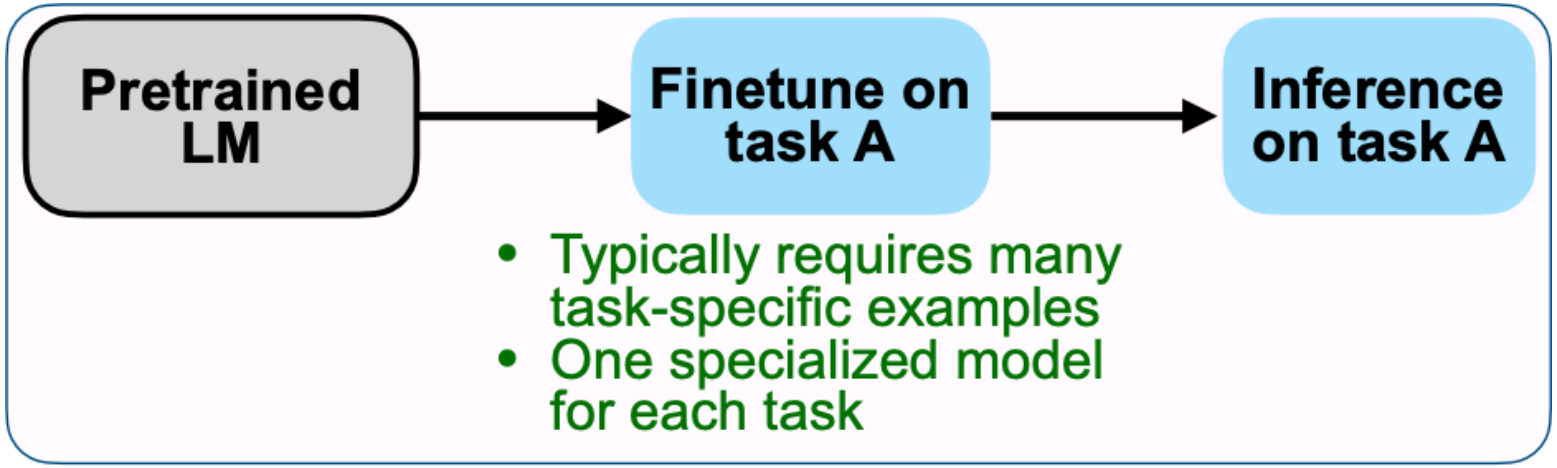




Inference with T5 model and instruction-tuned T5 model

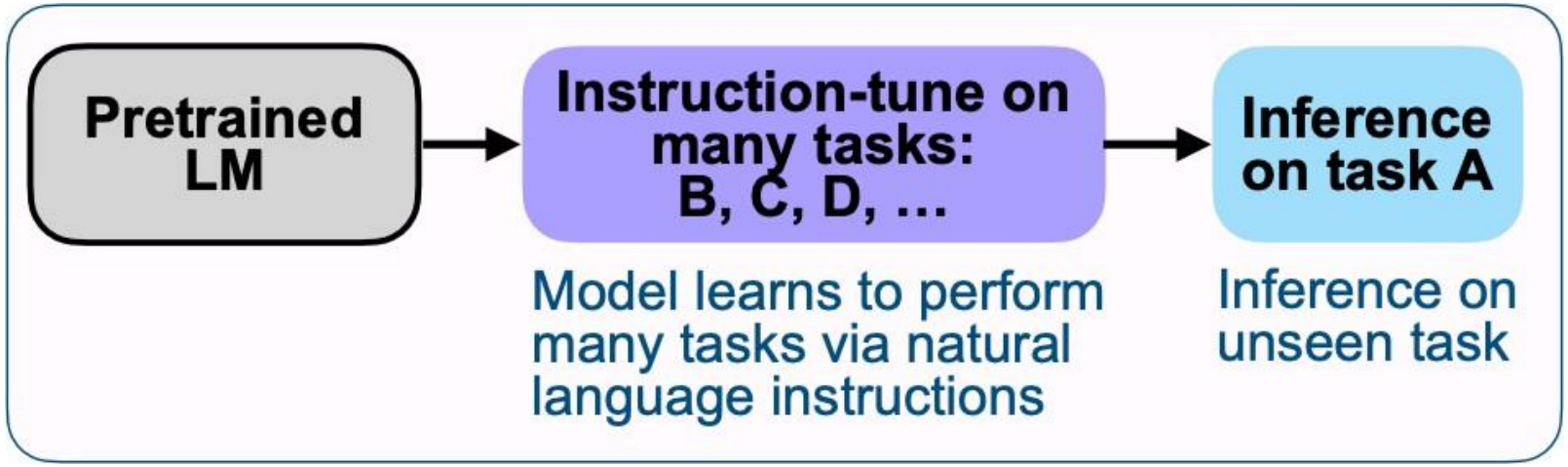
Recall Finetuning

- The pre-training stage let language models learn generic representation and knowledge from the corpus, but they are not specifically fine-tuned on any form of user tasks.
- To adapt language models to a specific downstream task, we could use task-specific datasets for fine-tuning

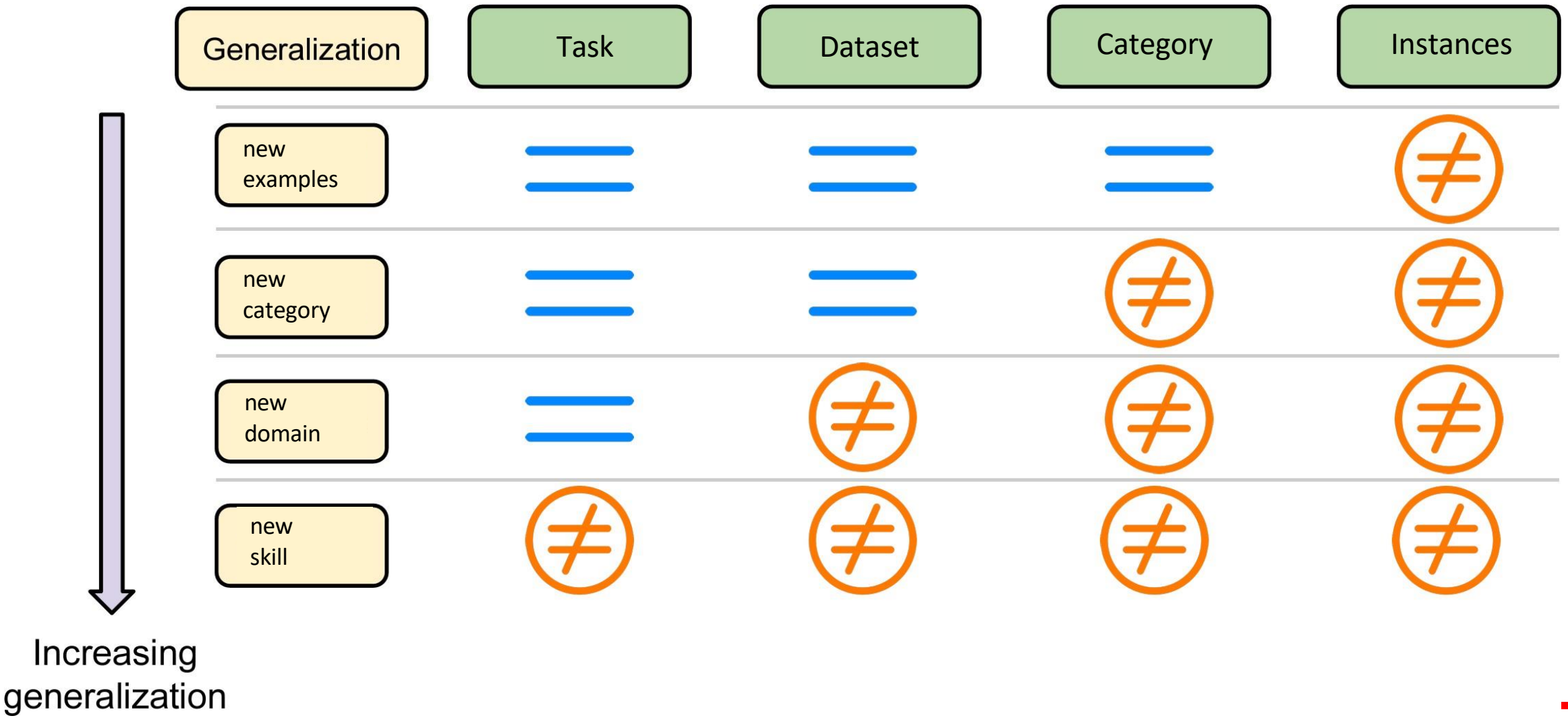


Instruction Tuning

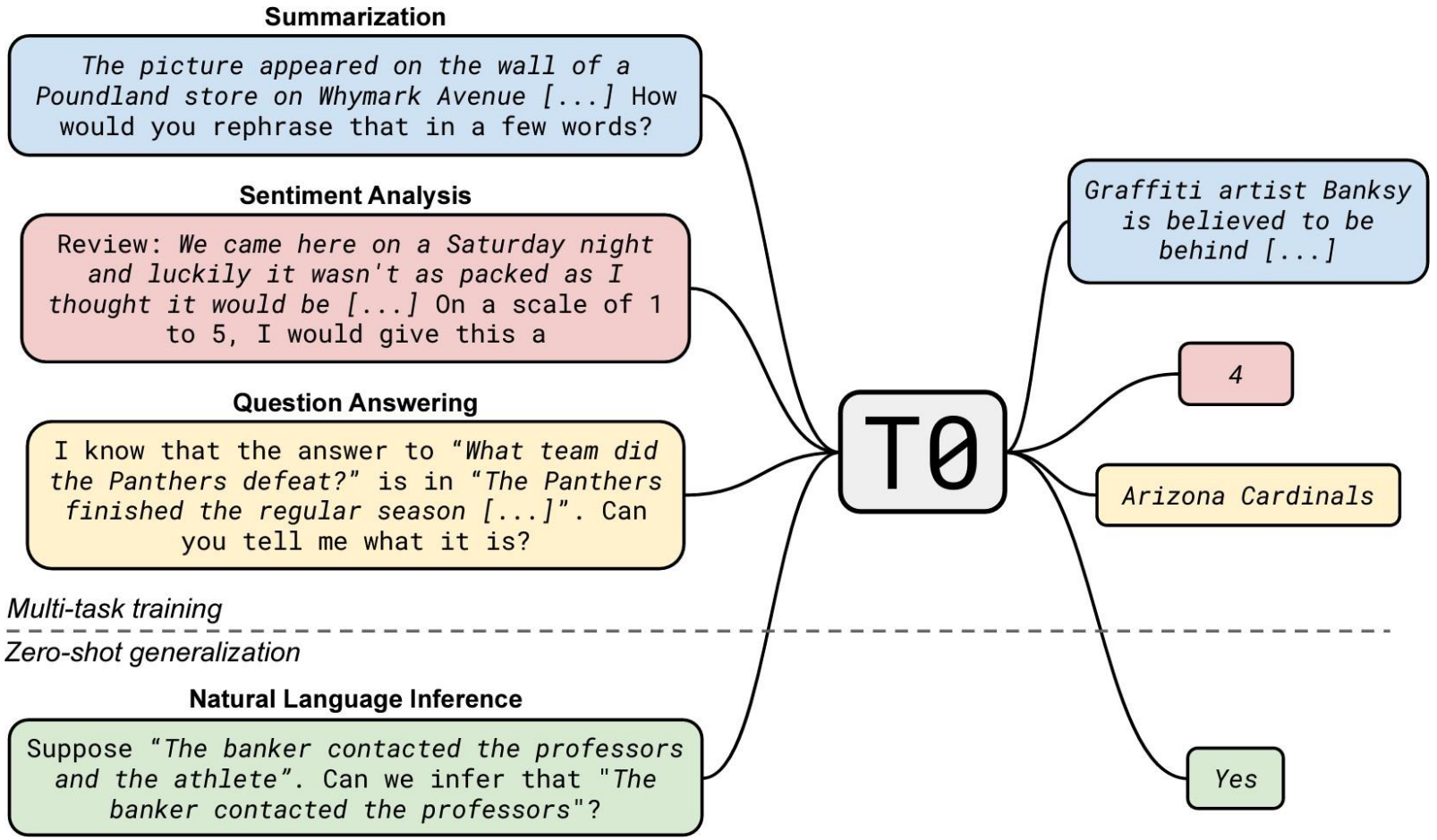
- Fine-tuning on many tasks! Teach language models to follow different natural language instructions, so that it could perform better on downstream tasks and generalize to unseen tasks!
- Fine-tuning -> Instruction Pre-training



Increasing Generalization

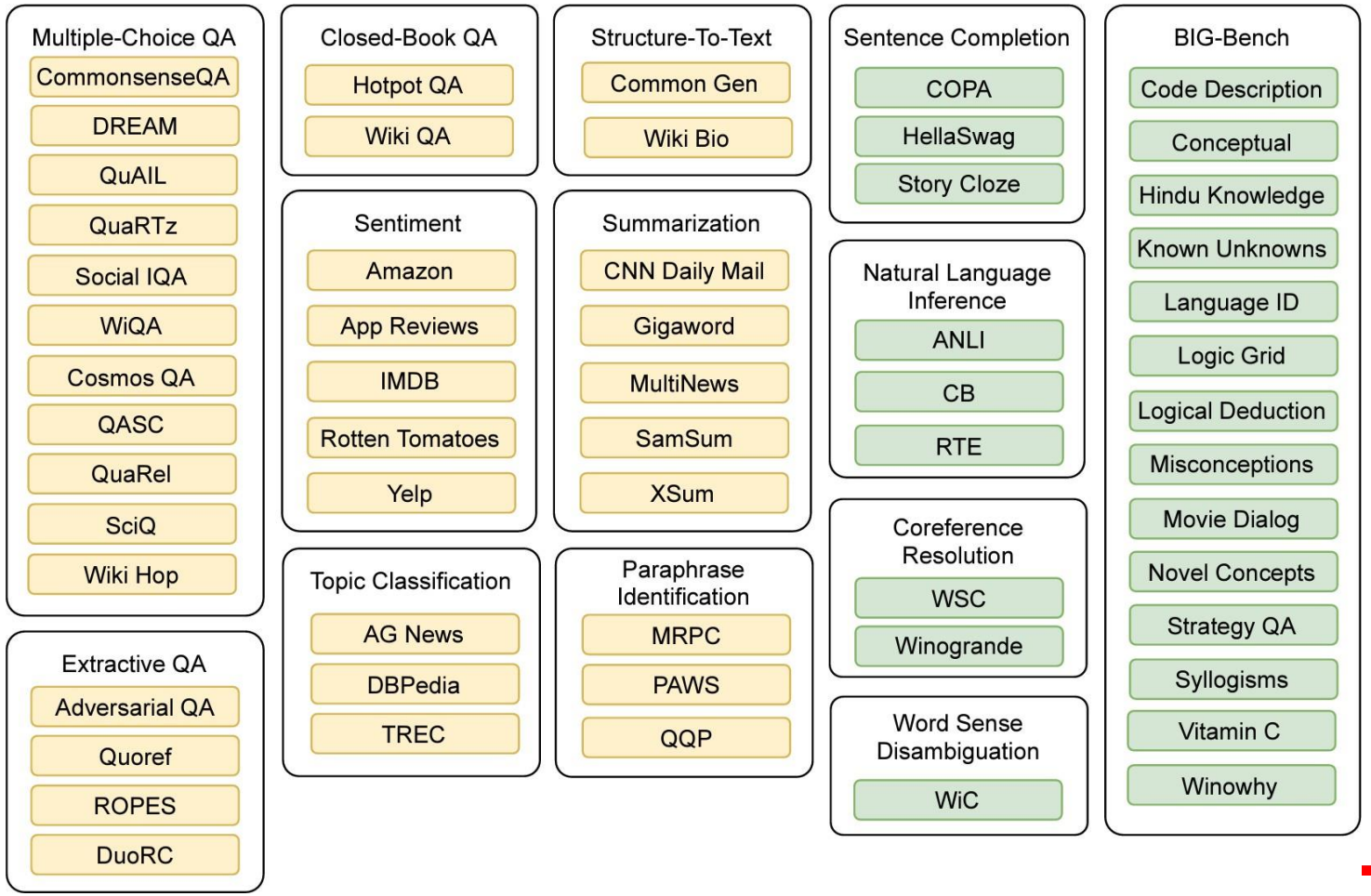


Multitask Prompted Training Enables Zero-Shot Task Generalization (Sanh et. al, 2021)



T0 Training Datasets

- Collecting from multiple public NLP datasets

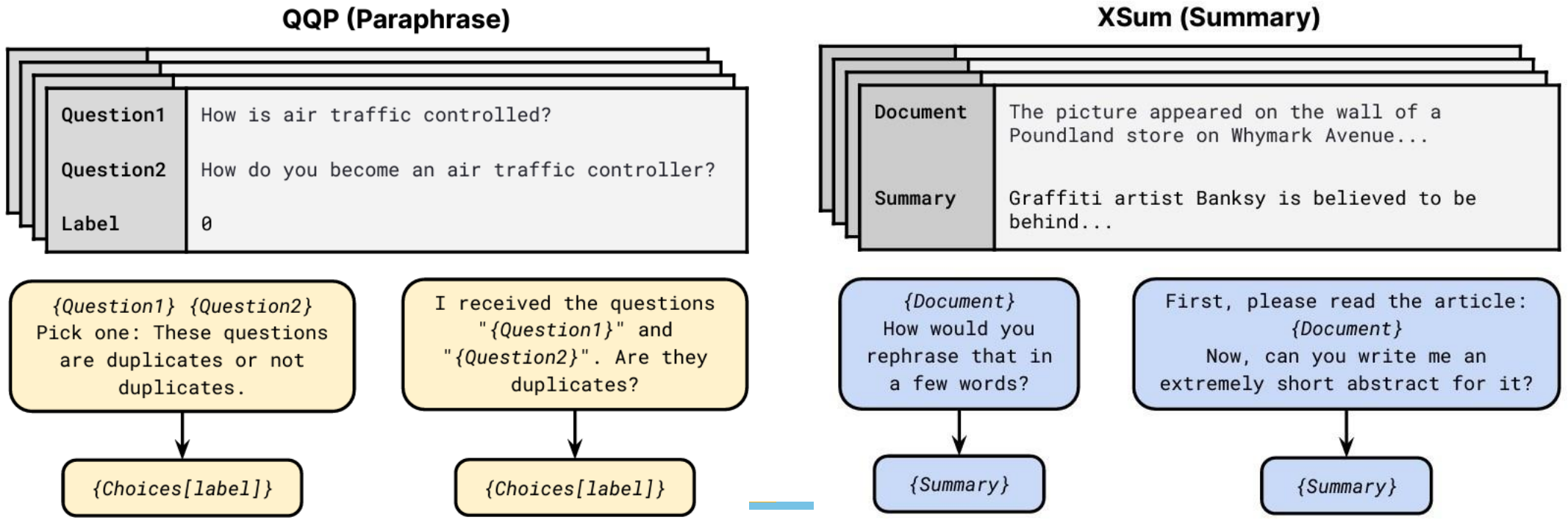


Training Mixtures and Held-Out Sets

- Training mixtures:
  - QA (Question Answering tasks), structure-to-text, summarization
  - Sentiment analysis, topic classification, paraphrase identification
- Held-out test set:
  - Sentence completion, BIG-Bench
  - Natural language inference, coreference resolution, word sense disambiguation

Task Adaptation with Prompt Templates

- Instead of directly using pairs of input and output, add specific instructions to explain each task (different templates per task)
- the outputs are tokens instead of class labels

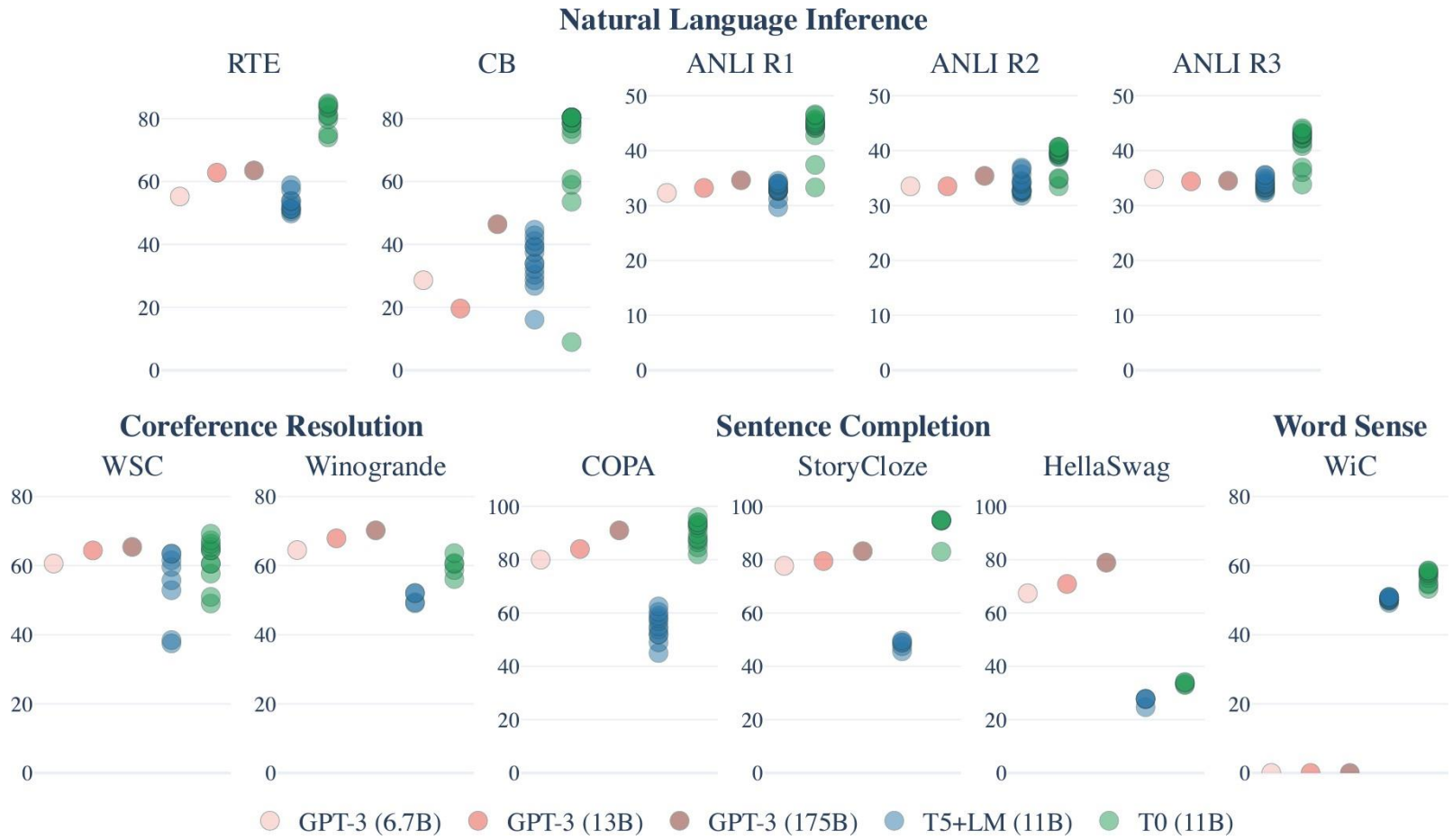


Experiments

- The proposed model is called T0, trained on T5-LM (11B model) with multitask mixture of training sets
- Baselines

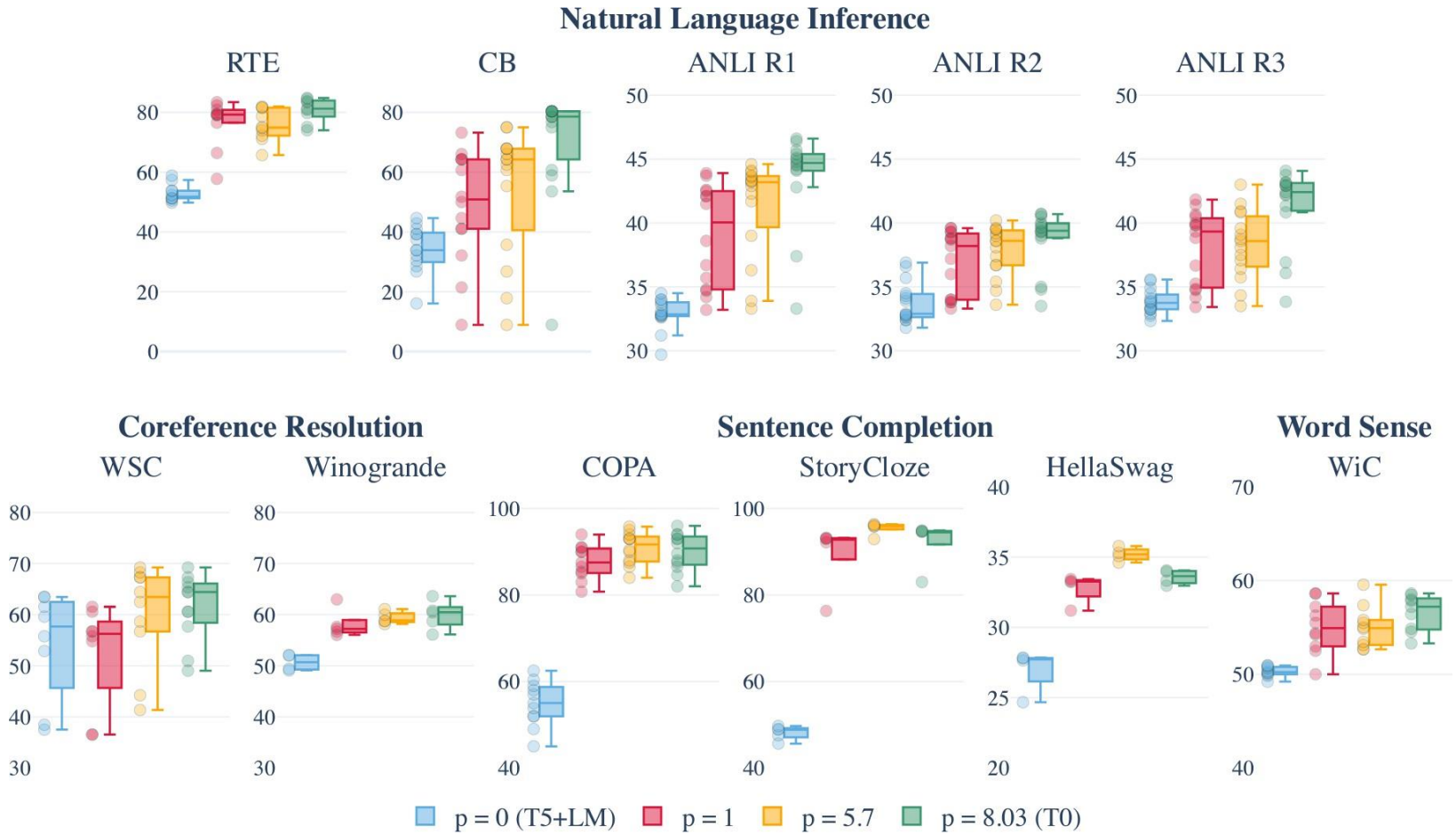
GPT-3 (6.7B) GPT-3 (13B) GPT-3 (175B) T5+LM (11B)

Performance on Held-Out Tasks

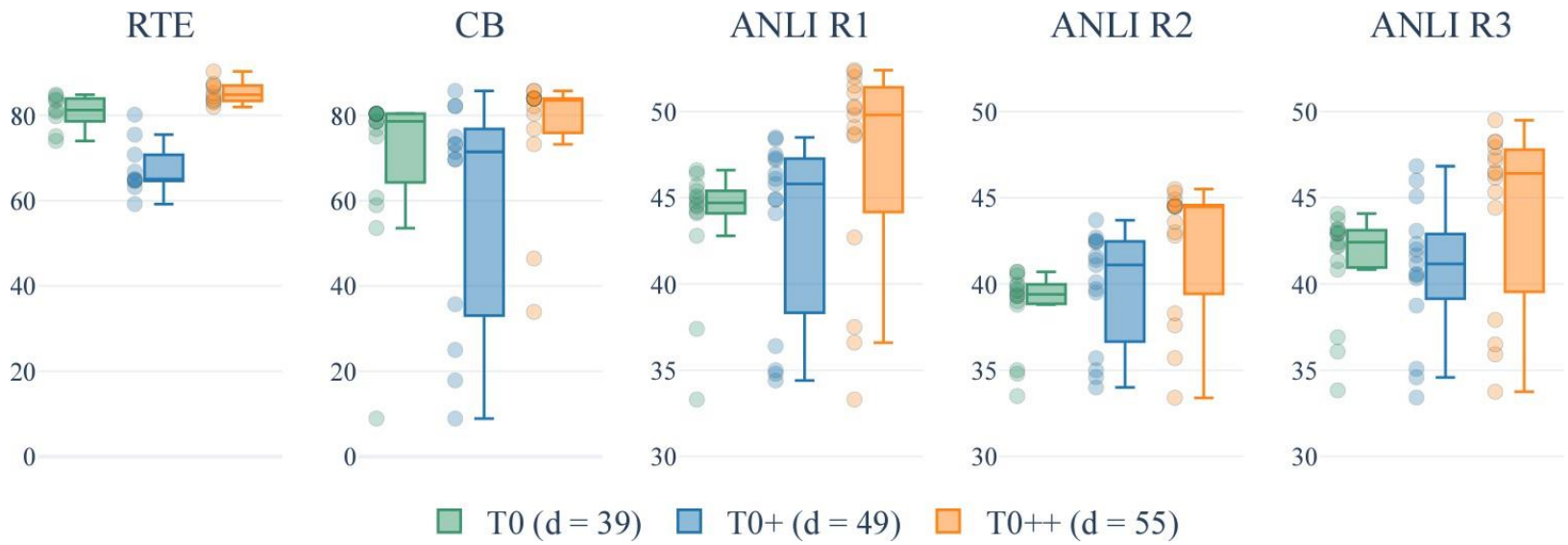


- For T5 and T0 models, each dot represents one evaluation prompt.

Effects of Prompt Numbers



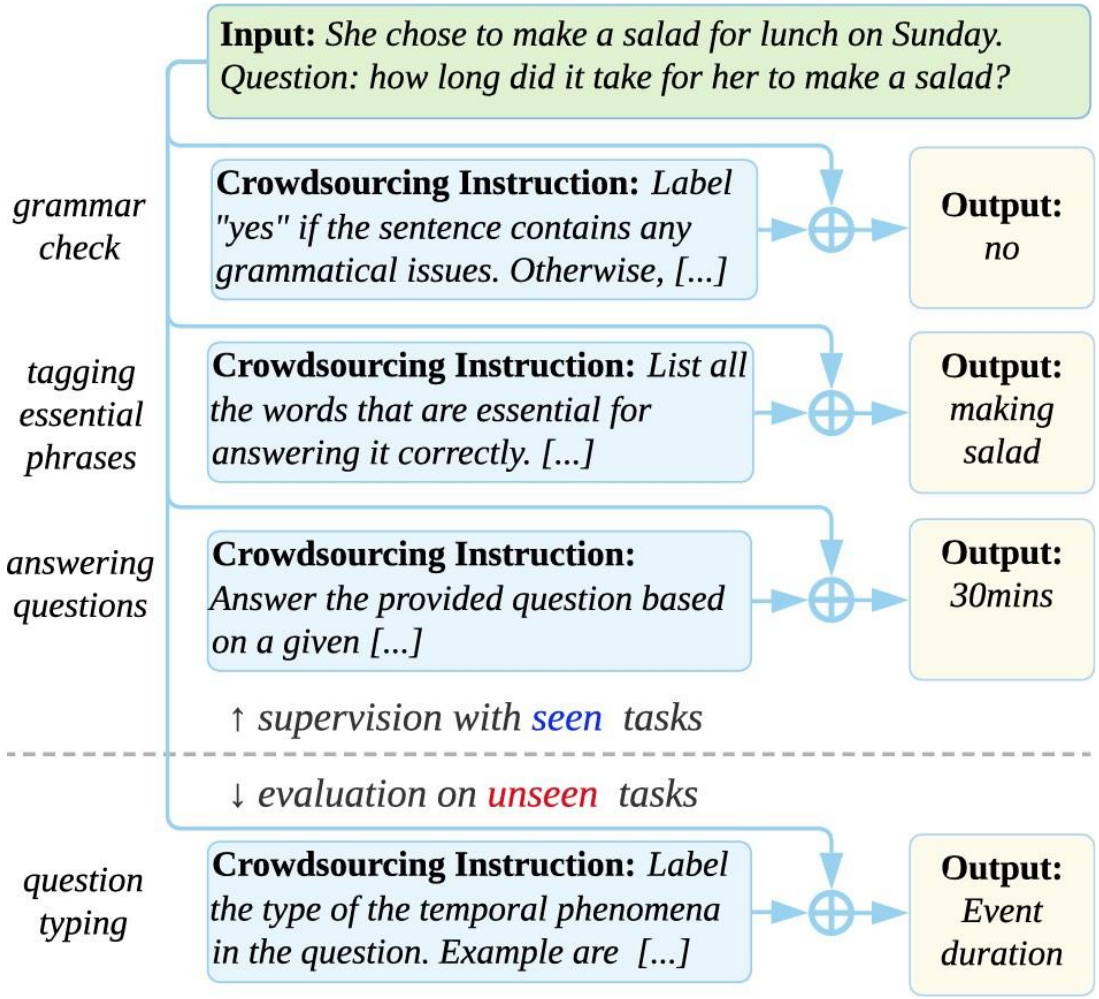
Effects of More Training Datasets



- Adding more datasets consistently leads to higher median performance

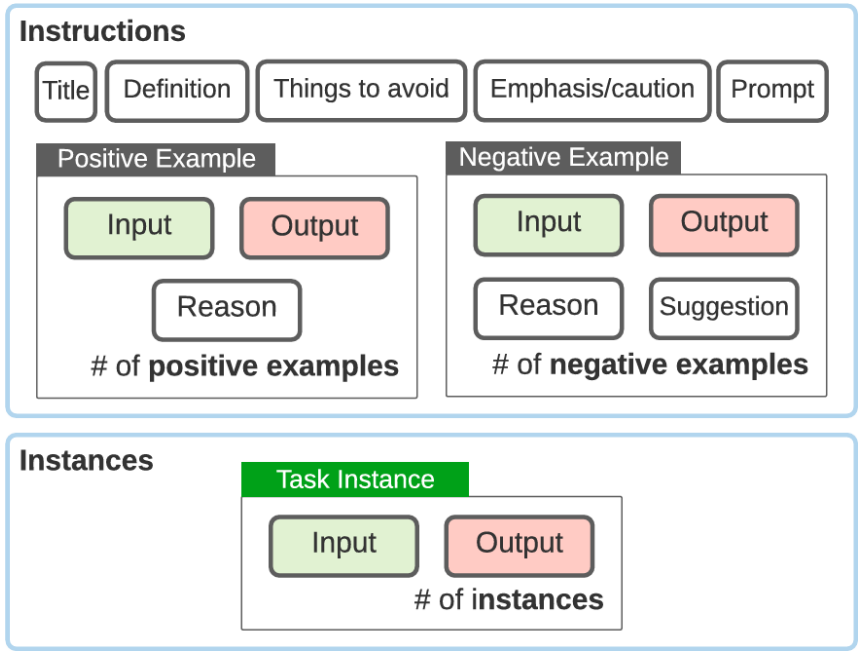
Cross-Task Generalization via Natural Language Crowdsourcing Instructions (Mishra et. al, 2021)

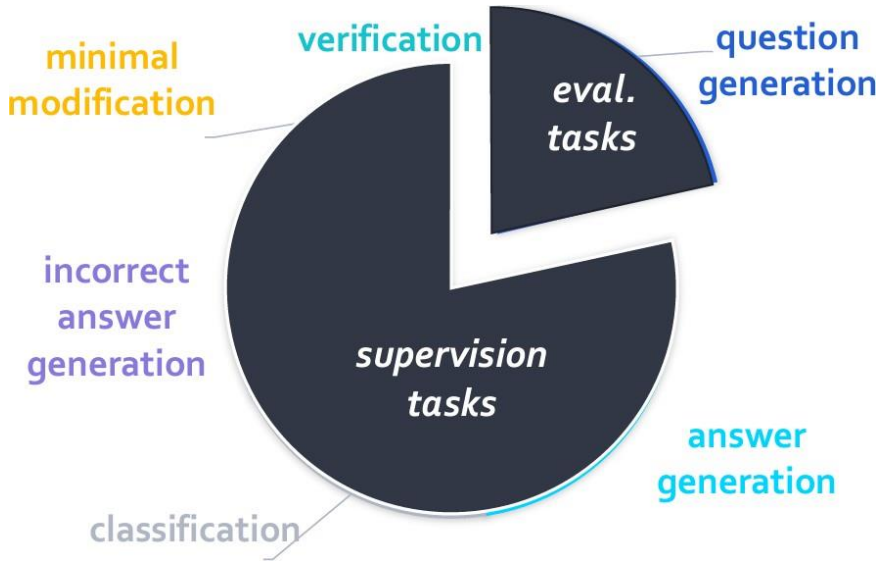
- Contribution:
- A crowdsourced dataset: Natural Instructions
  - 61 distinct tasks
  - 193k instances (input -> output)
- A more complete instruction schema



Crowdsourced Dataset

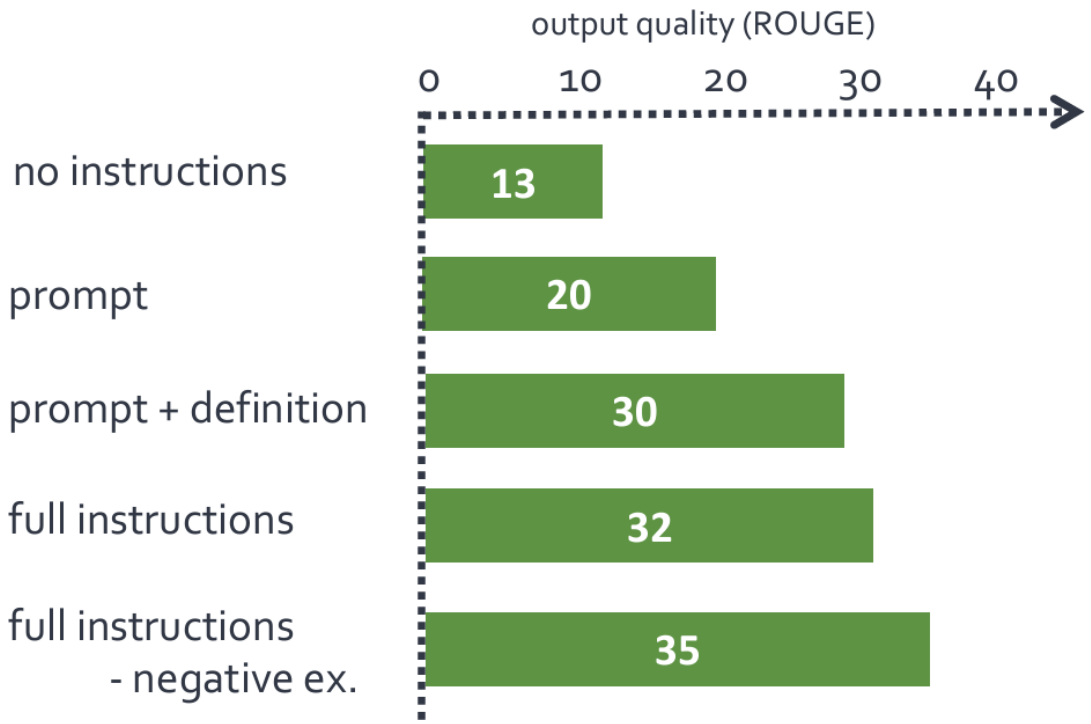
- 1. Randomly split the tasks (12 evaluation tasks, 49 supervision tasks)
- 2. Leave-one-category-out





Experiments: Generalization to Unseen Tasks

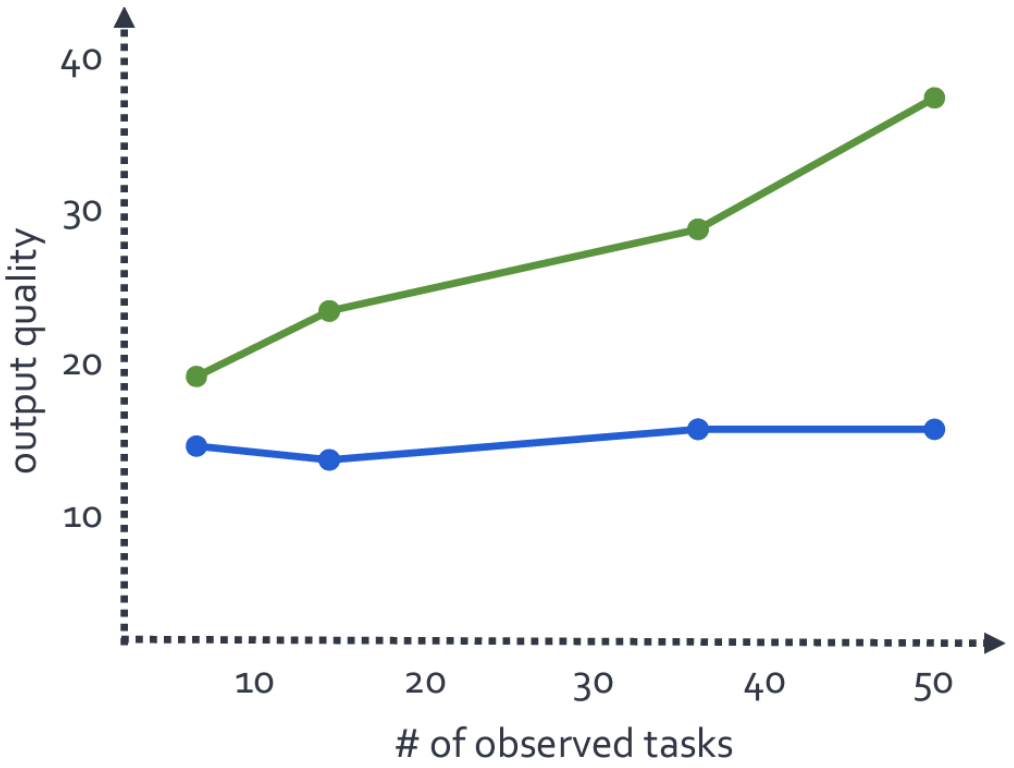
- Model: BART (140M params., instruction-tuned)



- All instruction elements (except negative examples) help improve model performance on unseen tasks.

Experiments: Number of Training Tasks

- Generalization to unseen tasks improves with more observed tasks

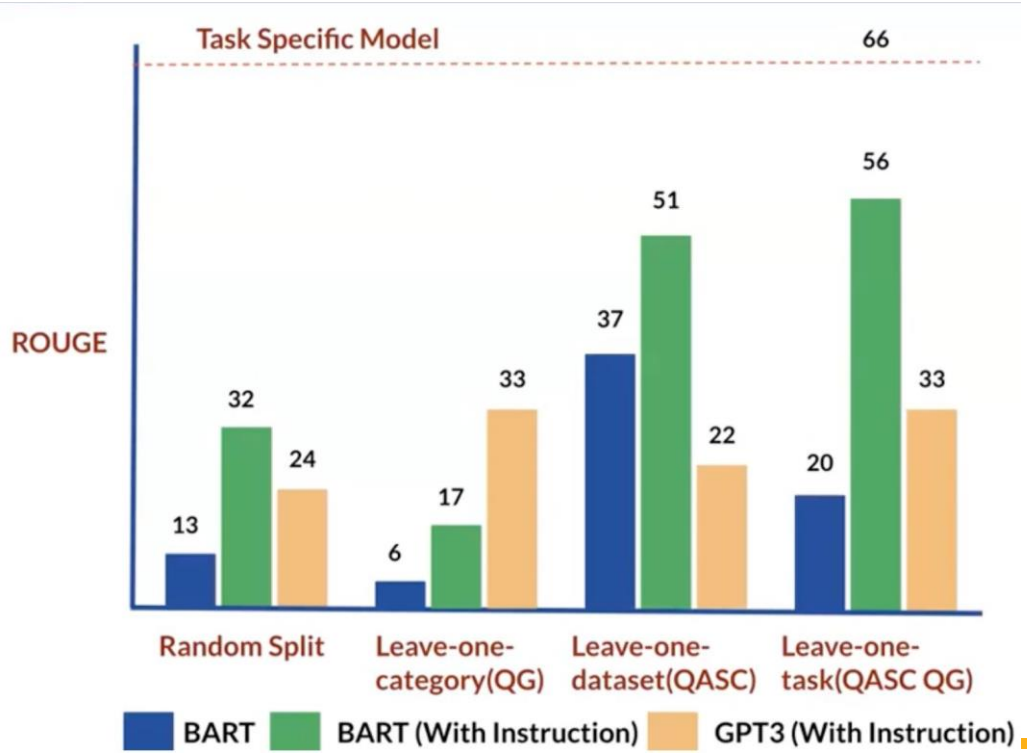


Full Instructions

No Instructions

Experiments: Comparison with Larger Model

- Model: BART (140M params., instruction-tuned)
- Baseline: GPT3 (175B params., not instruction-tuned)



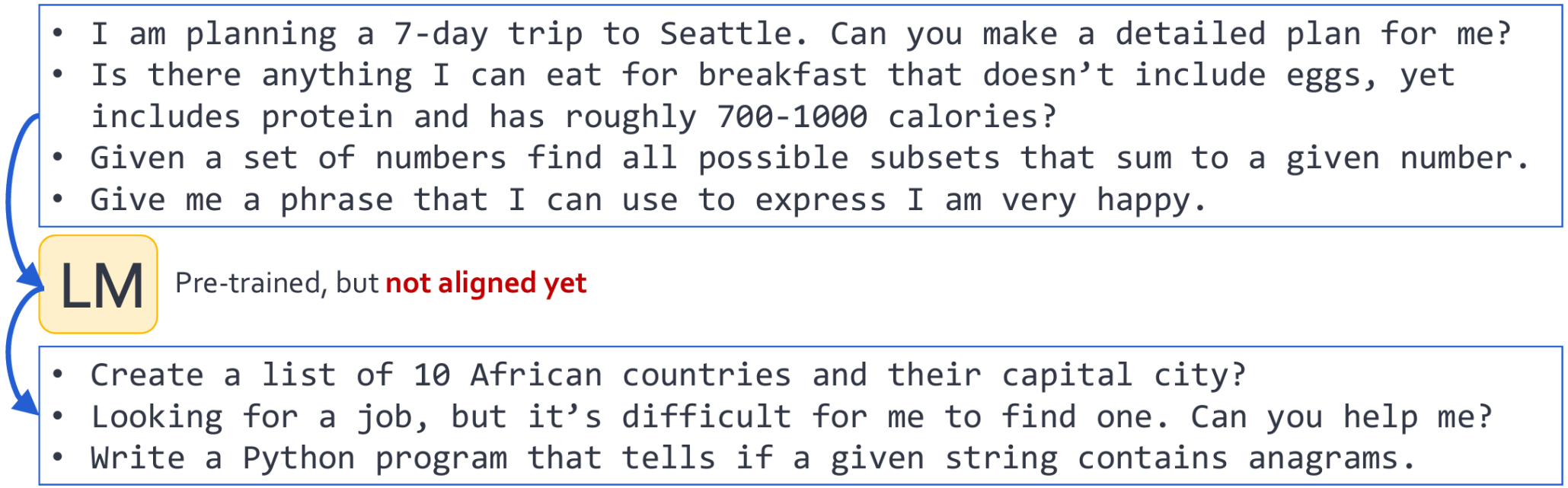
- Instructions consistently improve model performance on unseen tasks.
- When both having access to instructions on the held-out set, BART (instruction-tuned) can often outperform GPT3 (not instruction-tuned)

Self-Instruct: Aligning Language Models with Self-Generated Instructions (Wang et. al, 2022)

- Human-written instruction data can be very expensive!
- Can we reduce the human annotations?
- Idea: bootstrap from off-the-shelf LMs

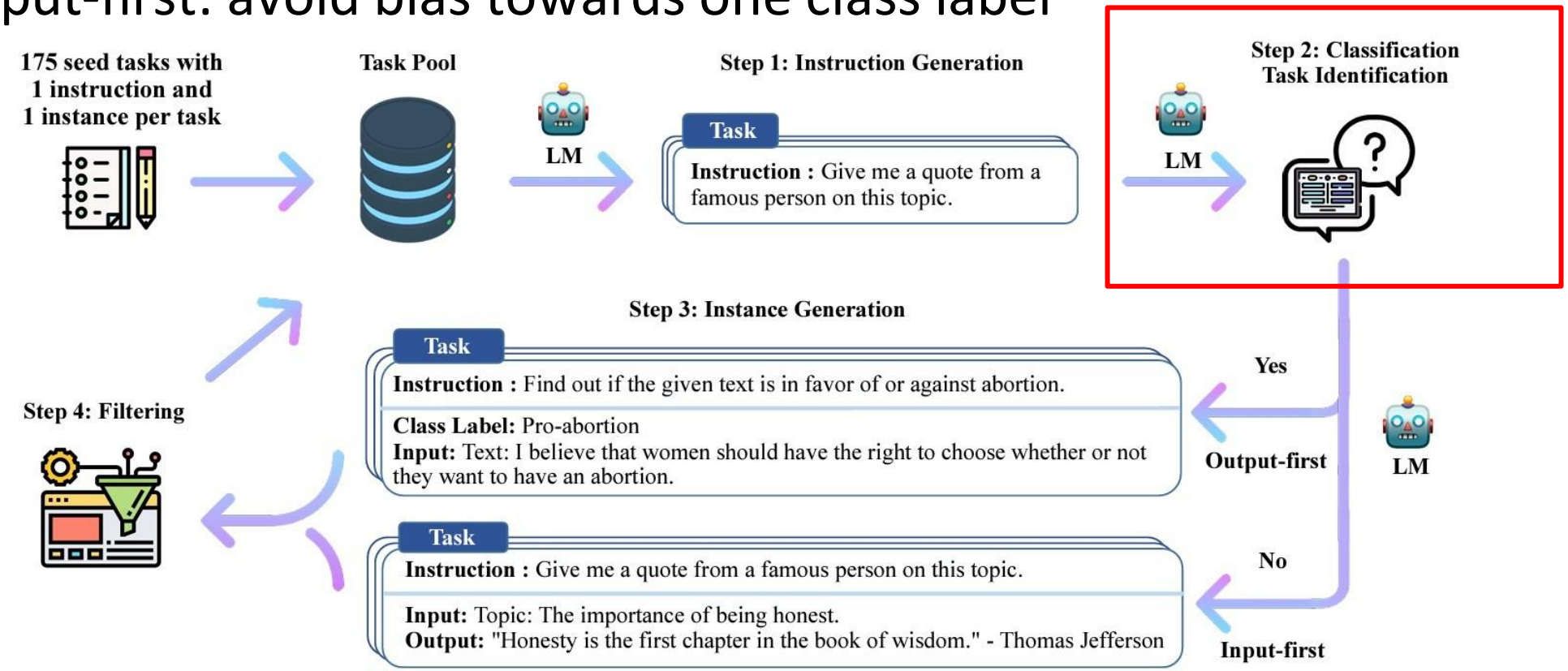
Self-Instruct: Aligning Language Models with Self-Generated Instructions (Wang et. al, 2022)

- Human written seed tasks to bootstrap off-the-shelf language models (GPT-3)



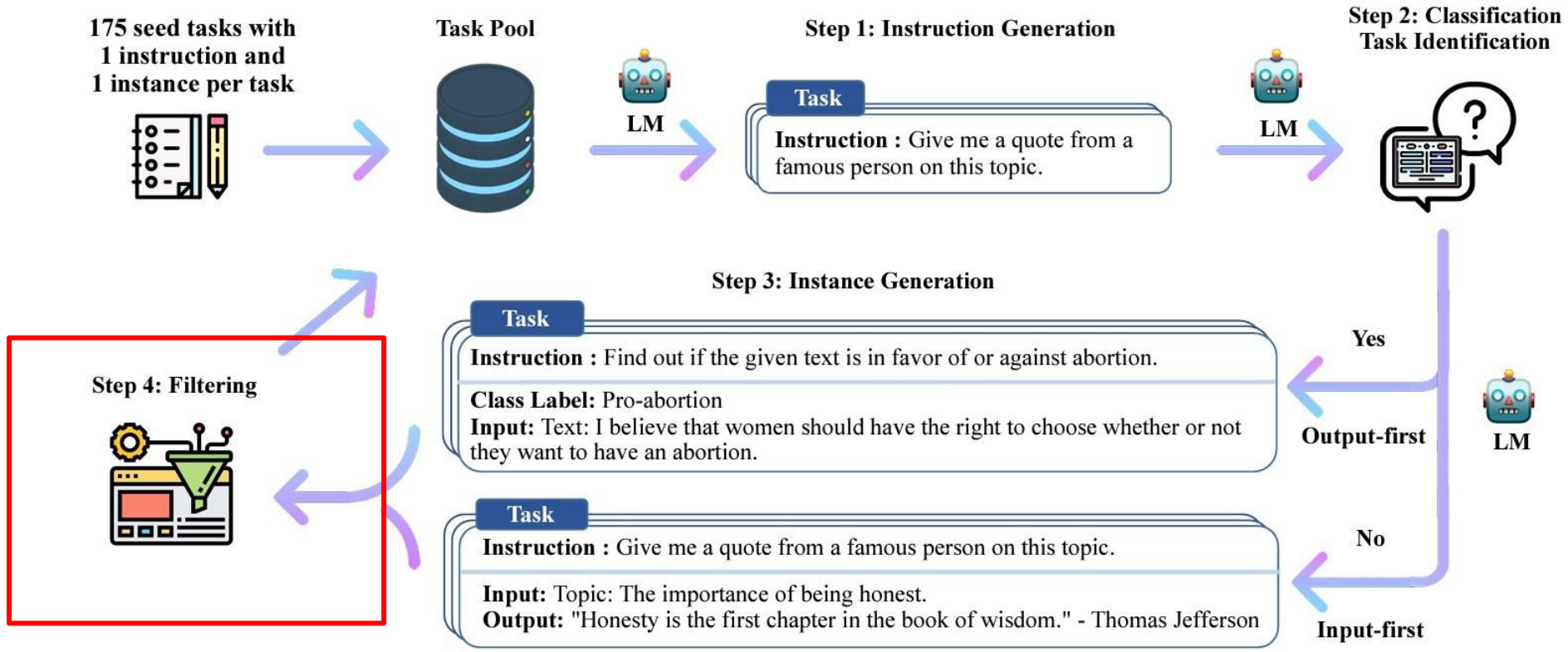
Self-Instruct Framework

- Classify whether the generated instruction is a classification task
- Output-first: avoid bias towards one class label



Self-Instruct Framework

- Filter out instructions similar with existing ones
- Add newly generated tasks into the task pool for next iteration

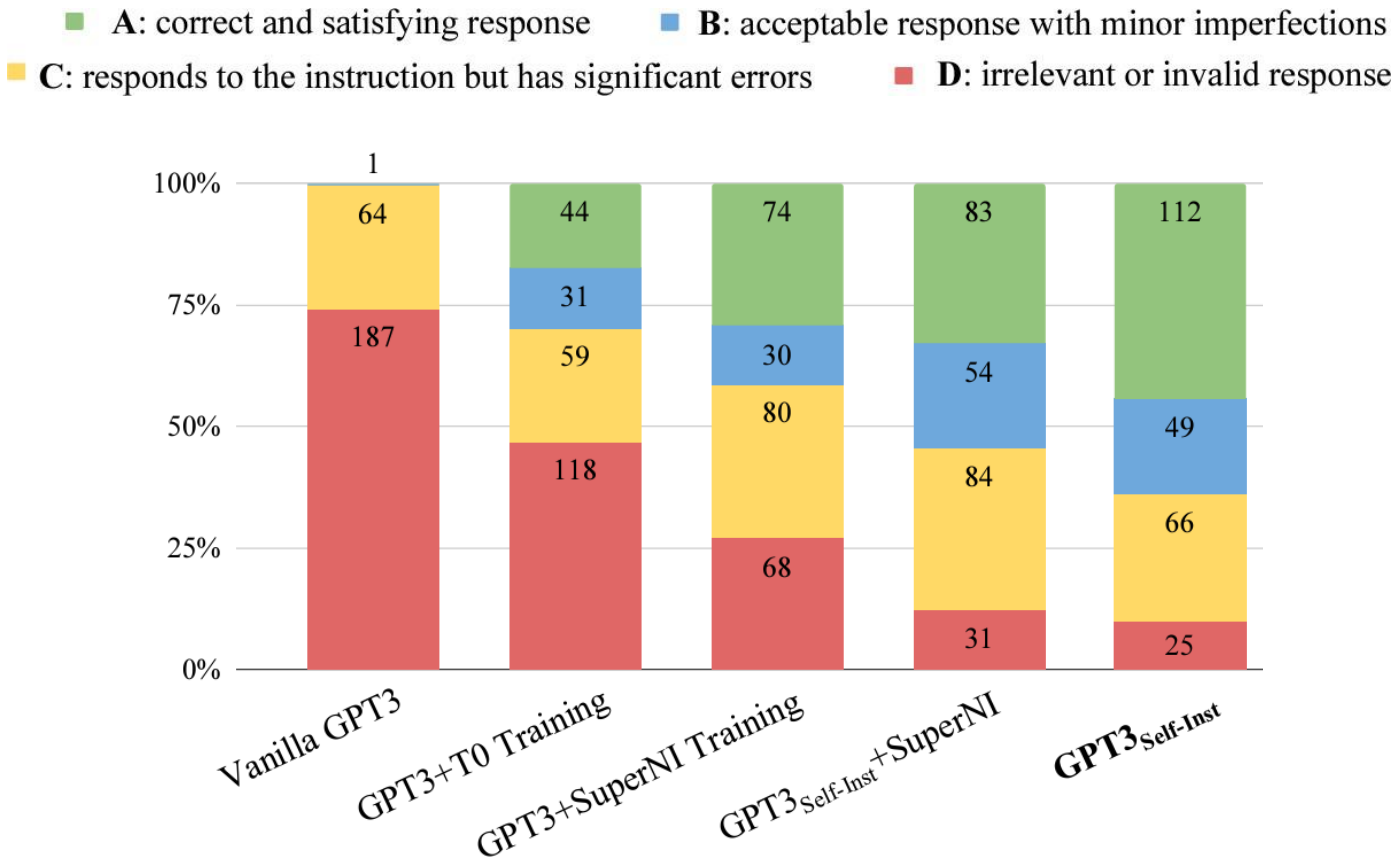


Experiment Results

- Use a GPT-3 (“davinci”) model to generate new instruction tasks, and fine-tune the GPT-3 model itself
- 175 seed tasks -> 52K instructions and 82K instances

statistic	
# of instructions	52,445
- # of classification instructions	11,584
- # of non-classification instructions	40,861
# of instances	82,439
- # of instances with empty input	35,878
ave. instruction length (in words)	15.9
ave. non-empty input length (in words)	12.7
ave. output length (in words)	18.9

Evaluation on User-Oriented Instructions



- Self-training the model by bootstrapping instruction tasks from limited human-written seed tasks can improve model alignment

LIMA: Less Is More for Alignment (Zhou et. al, 2023)

- Can we use a small number of data to generalize to new tasks?
- Hypothesis: A model's knowledge and capabilities are learn almost entirely during pre-training, while alignment teaches it the right format to be used when interacting with users

LIMA: Less Is More for Alignment (Zhou et. al, 2023)

- 1000 training examples: no more distillation data and with minor human annotations (200)
  - 750 top questions selected from community forums
  - manually write 250 examples of prompts and responses to emphasize the response style of an AI assistant
  - Finally train a 65B LLaMa model on 1000 demonstrations.

Source	#Examples	Avg Input Len.	Avg Output Len.
Training			
Stack Exchange (STEM)	200	117	523
Stack Exchange (Other)	200	119	530
wikiHow	200	12	1,811
Pushshift r/WritingPrompts	150	34	274
Natural Instructions	50	236	92
Paper Authors (Group A)	200	40	334
Dev			
Paper Authors (Group A)	50	36	N/A
Test			
Pushshift r/AskReddit	70	30	N/A
Paper Authors (Group B)	230	31	N/A

LIMA: Less Is More for Alignment

- Quality and diversity are the keys
- Quality Control:
  - Public data: select data with higher user ratings
  - In-house authored data: uniform tone and format
- Diversity Control:
  - Public data: Stratified sampling to increase domain diversity
  - In-house authored data: Increase task/scenario

Comparing LIMA with other LLMs

- Ask human crowd workers and GPT-4 which model response is better

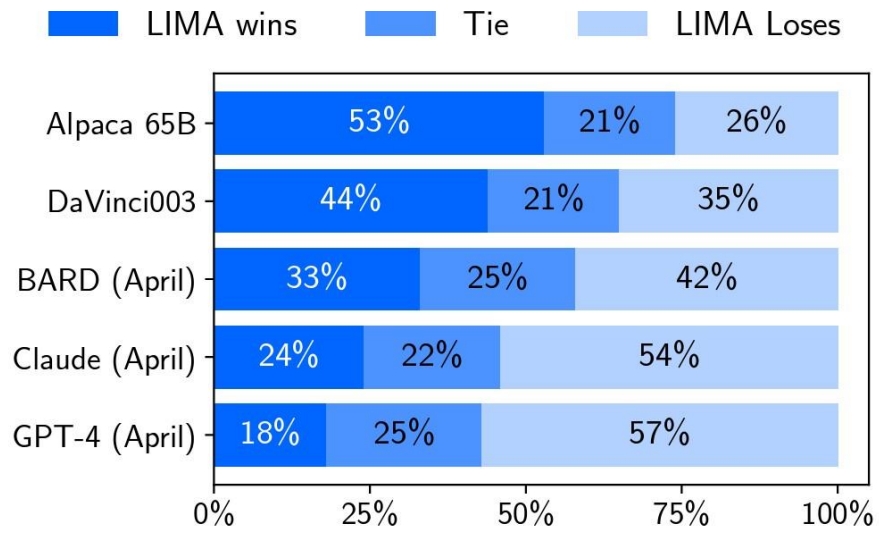


Figure 1: Human preference evaluation, comparing LIMA to 5 different baselines across 300 test prompts.

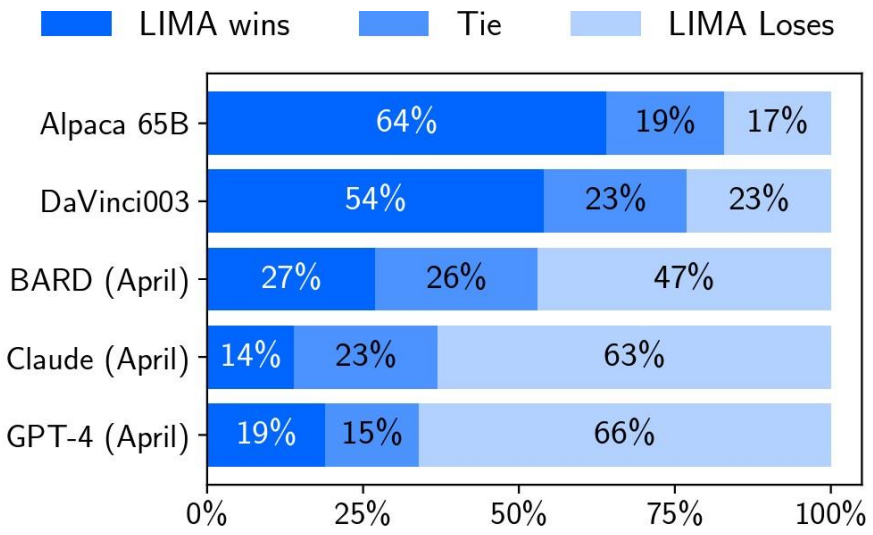


Figure 2: Preference evaluation using GPT-4 as the annotator, given the same instructions provided to humans.

Quality vs. Quantity vs. Diversity

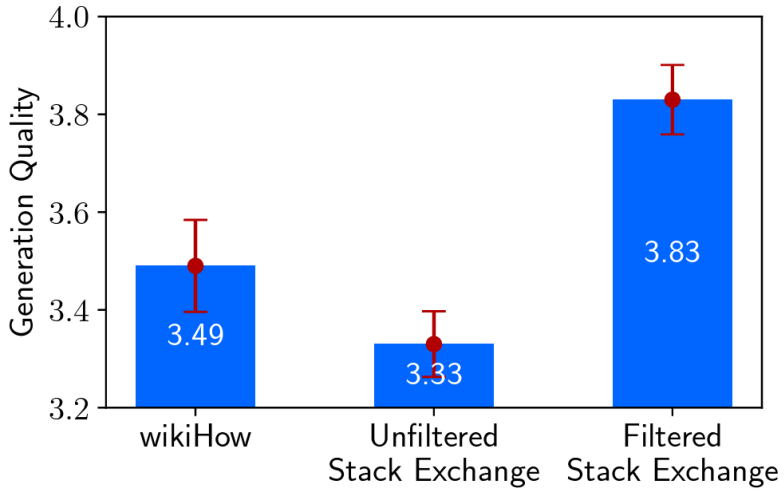
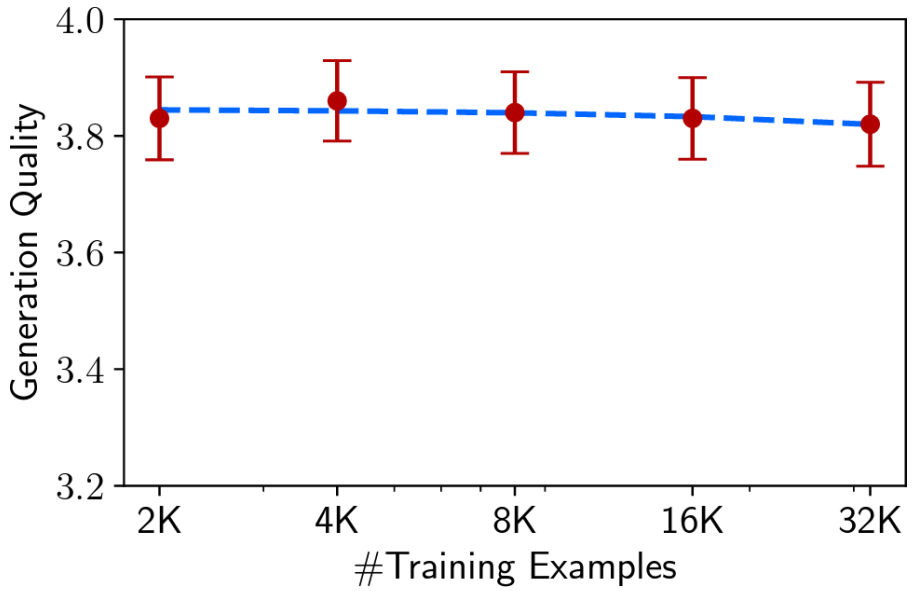


Figure 5: Performance of 7B models trained with 2,000 examples from different sources. **Filtered Stack Exchange** contains diverse prompts and high quality responses; **Unfiltered Stack Exchange** is diverse, but does not have any quality filters; **wikiHow** has high quality responses, but all of its prompts are "how to" questions.

- Scaling up training data does not necessarily improve the model response quality



Quality of Long-Form Highly Structured Response

- LIMA w. or w/o 6 format constraint exmaples
  - generating a product page with *highlights, about the product* and *how to use*
  - paper reviews with *summary, strengths, weakness* and *potentials*

LIMA w/o 6 Format Constraint Examples (994 Examples)	LIMA (1,000 Examples)

[Marketing Plan (Out of Distribution)]
I want to start a local coffee shop. Create a marketing plan with the following elements: Marketing Goals and Objectives, Define Target Audience, Research Marketing Tactics, Plan Marketing Tactics and Develop Your Timeline and Budget.

Coffee Shop Marketing Plan

Executive Summary

The following marketing plan for Java Jolt Coffee Bar outlines an aggressive approach to establishing a thriving coffee shop business in downtown Pittsburgh. Java Jolt will capitalize on the popularity of specialty coffees and the growing demand for WiFi hot spots by offering a comfortable environment where customers can enjoy premium coffee drinks while taking advantage of free Internet access.

Market research indicates that there is a significant opportunity for a new coffee bar located in the heart of the downtown business district. Market trends suggest that consumers are willing to pay a higher price for quality specialty coffees served in a welcoming atmosphere. Additionally, many office workers and students living in the area do not have easy access to high-speed Internet service, presenting an additional revenue stream for the coffee bar.

This marketing plan includes the following elements:

Here's a marketing plan for your local coffee shop.

Marketing Goals and Objectives

- Increase awareness of our brand among consumers [...]
- Drive foot traffic [...] during slow business hours [...]
- Build relationships with influencers in our community who [...]

Define Target Audience

Our target audiences include:

- Residents of the neighborhood surrounding our coffee shop
- Office workers near our coffee shop
- Students at nearby schools

We will focus our marketing efforts on these audiences because they are most likely to become regular customers at our [...]

Research Marketing Tactics

Here are some marketing tactics that we could use to reach to reach

Tulu: How Far Can Camels Go?

innovate achieve lead

Exploring the State of Instruction Tuning on Open Resources (Wang et. al, 2023)

- A comprehensive study on different instruction-tuning datasets
- Two mixtures of datasets
  - Human data mixture
  - Human + GPT data mixture

Comparison of Using Different Instruction Tuning Datasets

- There is not a single best instruction tuning dataset across all tasks
- Combining datasets results in the best overall performance

	MMLU (factuality)	GSM (reasoning)	BBH (reasoning)	TydiQA (multilinguality)	Codex-Eval (coding)	AlpacaEval (open-ended)	Average
	EM (0-shot)	EM (8-shot, CoT)	EM (3-shot, CoT)	F1 (1-shot, GP)	P@10 (0-shot)	Win % vs Davinci-003	
Vanilla LLaMa 13B	42.3	14.5	39.3	43.2	28.6	-	-
+SuperNI	49.7	4.0	4.5	**50.2**	12.9	4.2	20.9
+CoT	44.2	40.0	41.9	47.8	23.7	6.0	33.9
+Flan V2	**50.6**	20.0	40.8	47.2	16.8	3.2	29.8
+Dolly	45.6	18.0	28.4	46.5	31.0	13.7	30.5
+Open Assistant 1	43.3	15.0	39.6	33.4	31.9	58.1	36.9
+Self-instruct	30.4	11.0	30.7	41.3	12.5	5.0	21.8
+Unnatural Instructions	46.4	8.0	33.7	40.9	23.9	8.4	26.9
+Alpaca	45.0	9.5	36.6	31.1	29.9	21.9	29.0
+Code-Alpaca	42.5	13.5	35.6	38.9	34.2	15.8	30.1
+GPT4-Alpaca	46.9	16.5	38.8	23.5	**36.6**	63.1	37.6
+Baize	43.7	10.0	38.7	33.6	28.7	21.9	29.4
+ShareGPT	49.3	27.0	40.4	30.5	34.1	**70.5**	42.0
+Human data mix.	50.2	38.5	39.6	47.0	25.0	35.0	39.2
+Human+GPT data mix.	49.3	**40.5**	**43.3**	45.6	35.9	56.5	**45.2**

Different Base Models

- Base model quality is extremely important for downstream performance
- LLaMA is pre-trained on more tokens than other models

	MMLU (factuality)	GSM (reasoning)	BBH (reasoning)	TydiQA (multilinguality)	Codex-Eval (coding)	AlpacaEval (open-ended)	Average
	EM (0-shot)	EM (8-shot, CoT)	EM (3-shot, CoT)	F1 (1-shot, GP)	P@10 (0-shot)	Win % vs Davinci-003	
Pythia 6.9B	34.8	16.0	29.2	32.8	20.9	23.5	26.2
OPT 6.7B	32.6	13.5	27.9	24.1	8.9	25.9	22.2
LLaMA 7B	44.8	25.0	38.5	43.5	29.1	48.6	38.3
LLaMA-2 7B	**49.2**	**37.0**	**44.2**	**52.8**	**33.9**	**57.3**	**45.7**

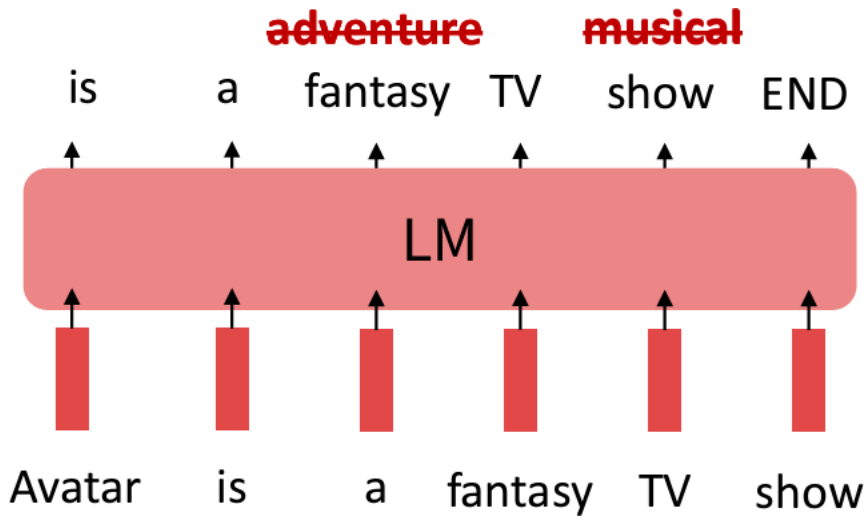
Different Model Sizes

- Smaller models benefit more from instruction-tuning
- Instruction-tuning does not help to enhance strong capabilities already exist in the original model

	MMLU (factuality)	GSM (reasoning)	BBH (reasoning)	TydiQA (multilinguality)	Codex-Eval (coding)	AlpacaEval (open-ended)	Average
	EM (0-shot)	EM (8-shot, CoT)	EM (3-shot, CoT)	F1 (1-shot, GP)	P@10 (0-shot)	Win % vs Davinci-003	
🐫 models trained on our final Human+GPT data mixture ↓							
TÜLU 🐫 7B	44.8 (+13.3)	25.0 (+15.0)	38.5 (+5.5)	43.5 (+5.1)	29.1 (+8.6)	48.6	38.3
TÜLU 🐫 13B	49.3 (+7.0)	40.5 (+26.0)	43.3 (+4.0)	45.6 (+2.4)	35.9 (+7.3)	56.5	45.2
TÜLU 🐫 30B	57.7 (+3.1)	53.0 (+17.0)	51.9 (+2.4)	51.9 (-3.4)	48.0 (+5.2)	62.3	54.1
TÜLU 🐫 65B	59.2 (+0.5)	59.0 (+9.0)	54.4 (-3.7)	56.6 (-0.2)	49.4 (+2.5)	61.8	56.7
🐫 models trained on our final Human+GPT data mixture using LLAMA-2 ↓							
TÜLU-1.1 🐫 7B	49.2 (+7.4)	37.0 (+25.0)	44.2 (+4.9)	52.8 (+1.6)	33.9 (+7.1)	57.3	45.7
TÜLU-1.1 🐫 13B	52.3 (+0.3)	53.0 (+28.0)	50.6 (+1.7)	58.8 (+2.3)	38.9 (+7.4)	64.0	52.9

Limitations of Instruction-Tuning

- Human-written pairs are very expensive
- Mismatch between LM objectives and human preferences
  - factual error vs. imprecise adjectives



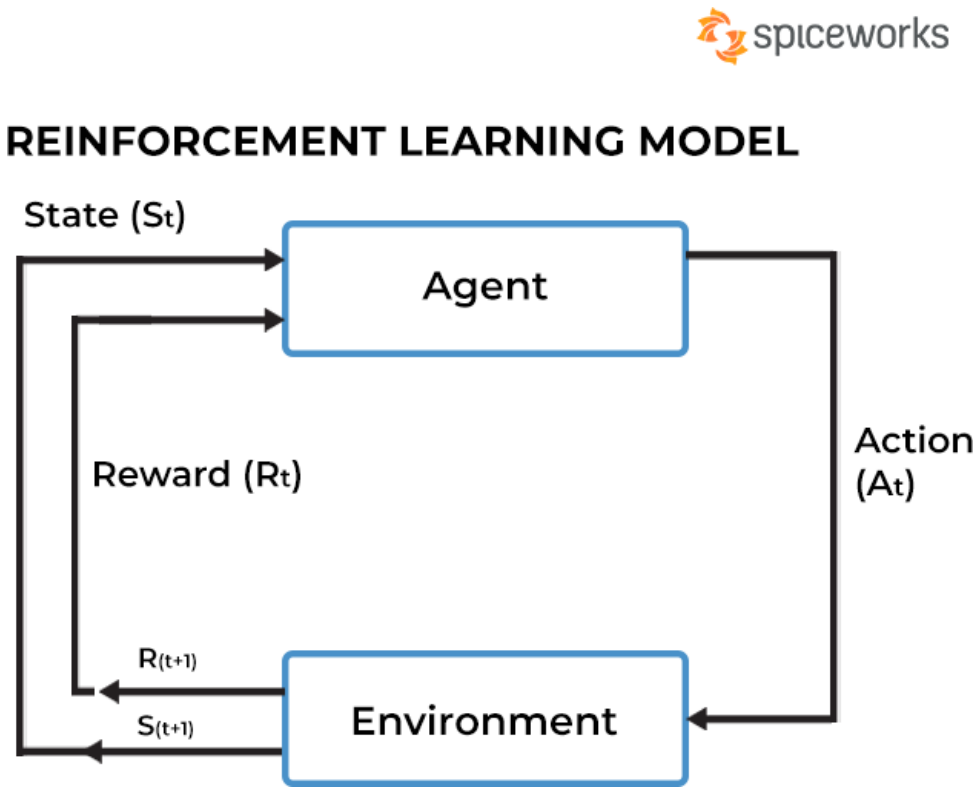
Common Objectives of Learning from Human Feedback

- Align model output with our values
- Trustworthy and robust on factualness
- Fairness on social values
- Explainable with logical rationales

Content

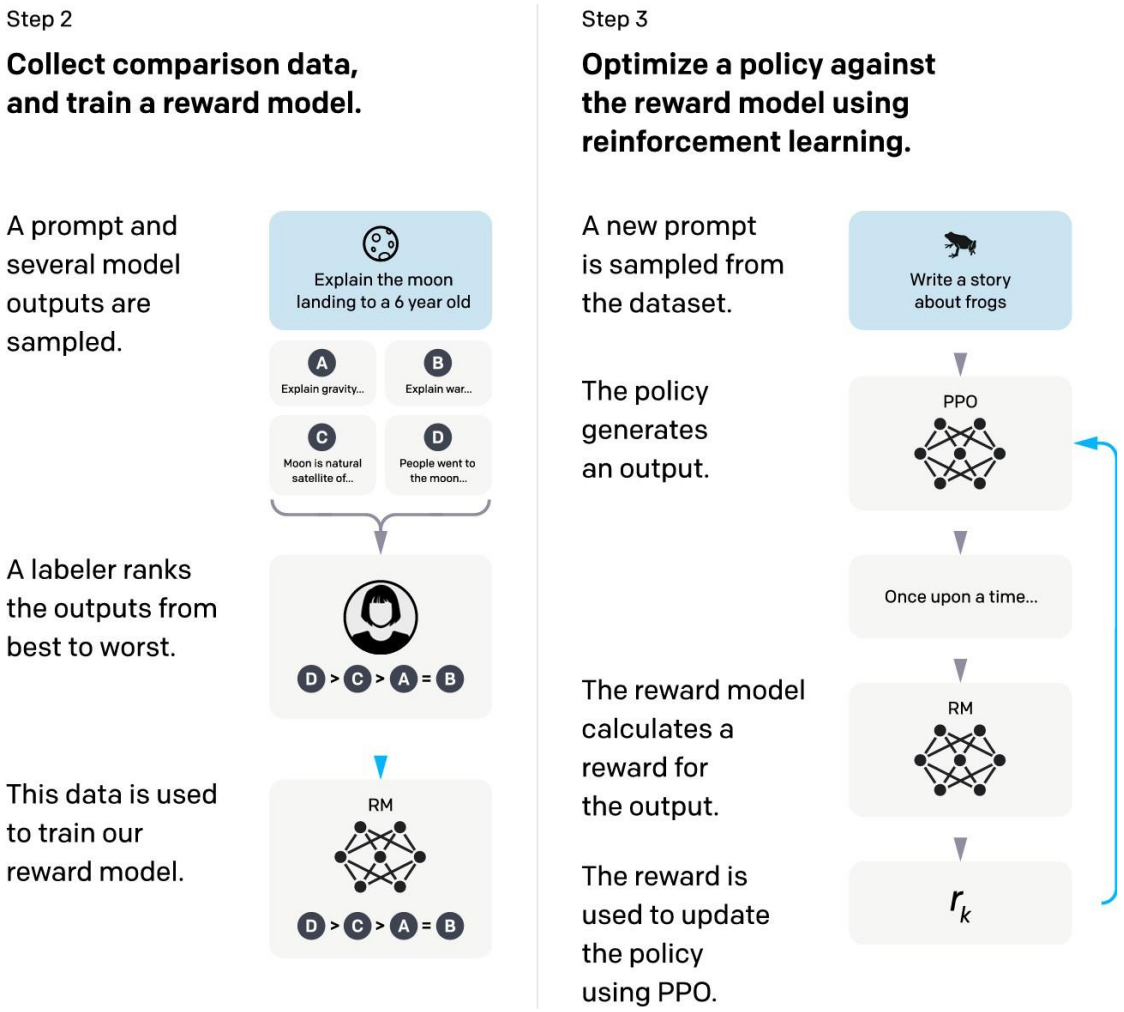
- **InstructGPT (Proximal Policy Optimization)**
- Direct Preference Optimization
- Fine-Grained Human Feedback
- Open problems for RLHF

Reinforcement Learning Model



- An agent has a policy function, which can take action A_t according to the current state S_t.
- As a result of the action, the agent receives a reward R_t from the environment and transit to the next state S_{t+1}.

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



- Agent: language model
- Action: predict the next token
- Policy π_θ : the output distribution of the next token

- Reward: a reward model r_ϕ trained by human evaluations on model responses, so no more human-in-the-loop is needed

Reward Model Training

- Prompt supervised fine-tuned language model with to produce pairs of answers

$$(y_1, y_2) \sim \pi^{\text{SFT}}(y \mid x)$$

- Human annotators decide which one wins / is preferred

$$y_w \succ y_l \mid x$$

- A reward model is trained to score y_w higher than y_l

$$\mathcal{L}_R(r_\phi, \mathcal{D}) = -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}}\left[\log \sigma(r_\phi(x, y_w) - r_\phi(x, y_l))\right]$$

- A reward model is often initialized from π^{SFT} with a linear layer to produce a scalar reward value

Fine-Tuning with RL: PPO[1]

- Optimize the language model π_θ with feedback from the reward model r_ϕ

$$\max_{\pi_\theta} \mathbb{E}_{x\sim\mathcal{D},y\sim\pi_\theta(y|x)}\left[r_\phi(x,y)\right] - \beta\mathbb{D}_{\mathrm{KL}}\left[\pi_\theta(y \mid x) \,||\, \pi_{\mathrm{ref}}(y \mid x)\right]$$

prefer responses with high rewards

control the deviation from the reference policy, the π^{SFT} model

[1] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms, 2017.

Fine-Tuning with RL: PPO[1]

- Optimize the language model π_θ with feedback from the reward model r_ϕ

$$\max_{\pi_\theta} \mathbb{E}_{x\sim\mathcal{D}, y\sim\pi_\theta(y|x)} \left[r_\phi(x, y) \right] - \beta \mathbb{D}_{\mathrm{KL}} \left[\pi_\theta(y \mid x) \,||\, \pi_{\mathrm{ref}}(y \mid x) \right]$$

prefer responses with high rewards

control the deviation from the reference policy, the π^{SFT} model

- prevent mode-collapse to single high reward answers
- prevent the model deviating too far from the distribution where the reward model is accurate

[1] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms, 2017.

Fine-Tuning with RL: PPO[1]

- Optimize the language model π_θ with feedback from the reward model r_ϕ

$$\max_{\pi_\theta} \mathbb{E}_{x\sim\mathcal{D}} \underbrace{_{y\sim\pi_\theta(y|x)}}_{\text{sample y from the current policy}} \underbrace{\left[r_\phi(x,y)\right]}_{\text{prefer responses with high rewards}} - \underbrace{\beta\mathbb{D}_{\mathrm{KL}}\left[\pi_\theta(y \mid x) \,||\, \pi_{\mathrm{ref}}(y \mid x)\right]}_{\text{control the deviation from the reference policy, the } \pi^{\mathrm{SFT}} \text{ model}}$$

- prevent mode-collapse to single high reward answers
- prevent the model deviating too far from the distribution where the reward model is accurate

[1] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms, 2017.

Fine-Tuning with RL: PPO-ptx[1]

- Training objective

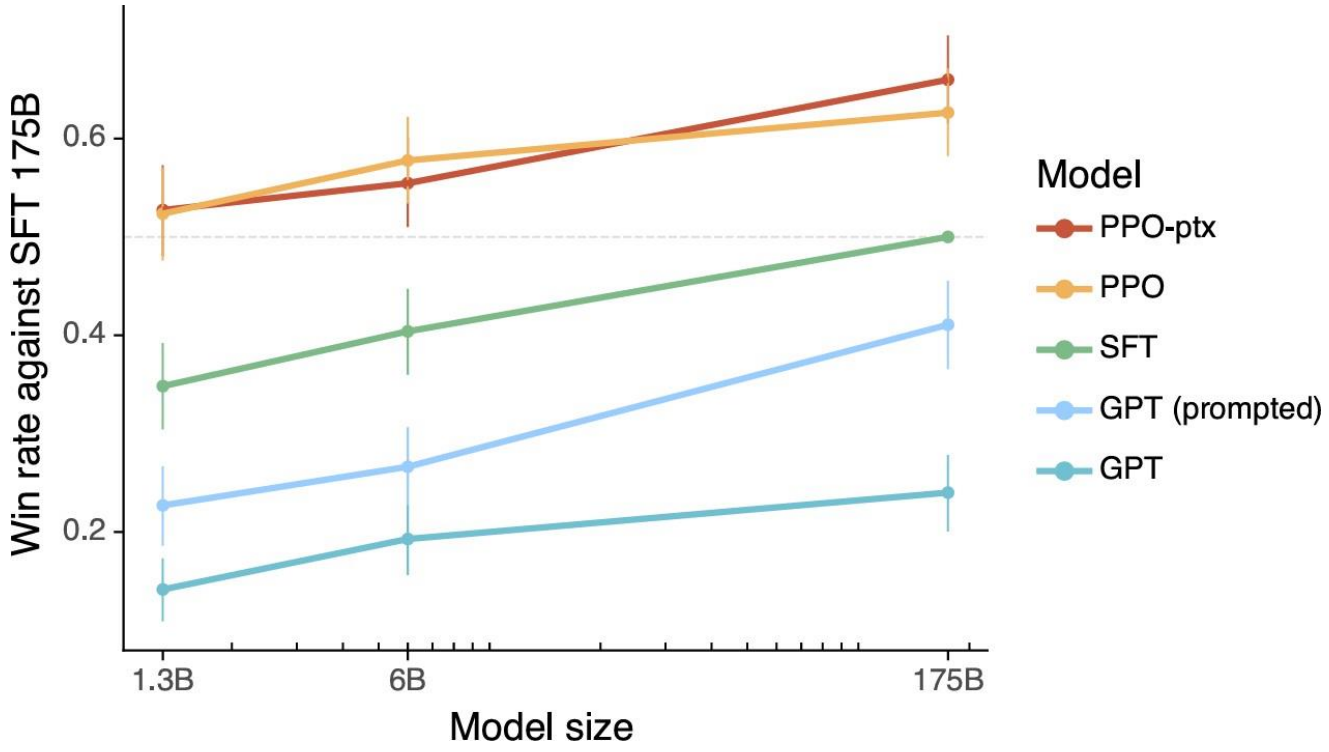
$$E_{(x,y)\sim D_{\pi_\phi^{\mathrm{RL}}}}\left[r_\theta(x,y) - \beta \log\left(\pi_\theta\ (y \mid x)/\pi_{\mathrm{ref}}(y \mid x)\right)\right] +$$
$$\gamma E_{x\sim D_{\mathrm{pretrain}}}\left[\log(\pi_\theta\ (x))\right]$$

- Add pre-training gradients to fix the performance regressions on public NLP tasks
- For PPO models, γ is set to 0

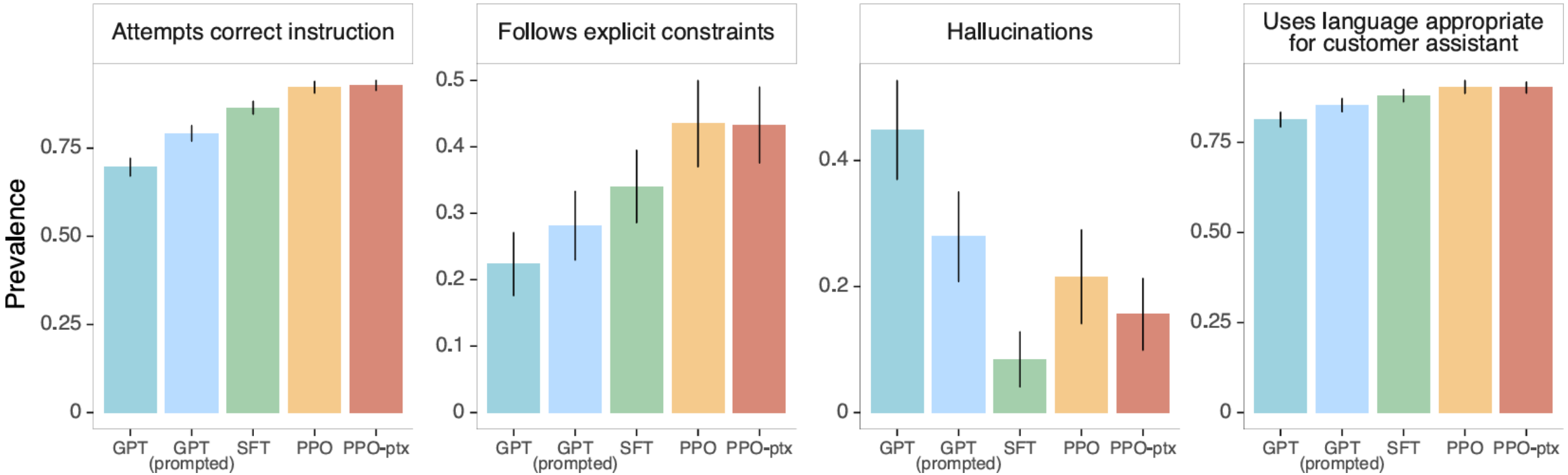
[1] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms, 2017.

Comparison with Baselines

- RLHF models are more preferred by human labelers



Evaluations on Different Aspects



Limitation of PPO methods

- Need to train multiple models: a reward model and a policy model
- Need sampling from LM during fine-tuning
- The RL training process is too complicated!
- Is it possible to directly train a language model from the human preference annotations?

DPO: Direct Preference Optimization: Your Language Model is Secretly a Reward Model (Rafailov et. al, 2023)

- Looking into the PPO objective

$$\max_{\pi_\theta} \mathbb{E}_{x\sim\mathcal{D},y\sim\pi_\theta(y|x)}\left[r_\phi(x,y)\right] - \beta\mathbb{D}_{\mathrm{KL}}\left[\pi_\theta(y \mid x) \,||\, \pi_{\mathrm{ref}}(y \mid x)\right]$$

- Deriving optimal closed-form solution

$$\max_{\pi} \mathbb{E}_{x\sim\mathcal{D},y\sim\pi}\left[r(x,y)\right] - \beta\mathbb{D}_{\mathrm{KL}}\left[\pi(y|x) \,||\, \pi_{\mathrm{ref}}(y|x)\right]$$

$$= \max_{\pi} \mathbb{E}_{x\sim\mathcal{D}}\mathbb{E}_{y\sim\pi(y|x)}\left[r(x,y) - \beta\log\frac{\pi(y|x)}{\pi_{\mathrm{ref}}(y|x)}\right]$$

$$= \min_{\pi} \mathbb{E}_{x\sim\mathcal{D}}\mathbb{E}_{y\sim\pi(y|x)}\left[\log\frac{\pi(y|x)}{\pi_{\mathrm{ref}}(y|x)} - \frac{1}{\beta}r(x,y)\right]$$

$$= \min_{\pi} \mathbb{E}_{x\sim\mathcal{D}}\mathbb{E}_{y\sim\pi(y|x)}\left[\log\frac{\pi(y|x)}{\frac{1}{Z(x)}\pi_{\mathrm{ref}}(y|x)\exp\left(\frac{1}{\beta}r(x,y)\right)} - \log Z(x)\right]$$

partition function:

$$Z(x) = \sum_y \pi_{\mathrm{ref}}(y|x)\exp\left(\frac{1}{\beta}r(x,y)\right)$$

Direct Preference Optimization

- PPO Objective

$$\min_{\pi} \mathbb{E}_{x\sim\mathcal{D}}\mathbb{E}_{y\sim\pi(y|x)}\left[\log \frac{\pi(y|x)}{\frac{1}{Z(x)}\pi_{\text{ref}}(y|x)\exp\left(\frac{1}{\beta}r(x,y)\right)} - \log Z(x)\right]$$

partition function:

$$Z(x) = \sum_{y} \pi_{\text{ref}}(y|x)\exp\left(\frac{1}{\beta}r(x,y)\right)$$

- Partition function is a function of only x and π_{ref} , but does not depend on the policy
- Therefore we can define π $\pi^*(y|x) = \frac{1}{Z(x)}\pi_{\text{ref}}(y|x)\exp\left(\frac{1}{\beta}r(x,y)\right)$
- $\pi^*(y|x)$ is a valid probability, therefore the objective can be seen as a KL divergence between two probability distribution
- The optimal solution of the objective

$$\pi(y|x) = \pi^*(y|x) = \frac{1}{Z(x)}\pi_{\text{ref}}(y|x)\exp\left(\frac{1}{\beta}r(x,y)\right)$$

Direct Preference Optimization

- Every reward function induce an optimal policy

$$\pi_r(y \mid x) = \frac{1}{Z(x)} \pi_{\text{ref}}(y \mid x) \exp\left(\frac{1}{\beta} r(x, y)\right)$$

- Every policy is the optimal policy of some reward function

$$r(x, y) = \beta \log \frac{\pi_r(y \mid x)}{\pi_{\text{ref}}(y \mid x)} + \boxed{\beta \log Z(x)}$$

This term is intractable!

- Key idea: train the policy model so that r(x,y) fits the human preference data!

Direct Preference Optimization

- Recall the reward model training loss

$$\mathcal{L}_R(r_\phi, \mathcal{D}) = -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}}\left[\log \sigma(r_\phi(x, y_w) - r_\phi(x, y_l))\right]$$

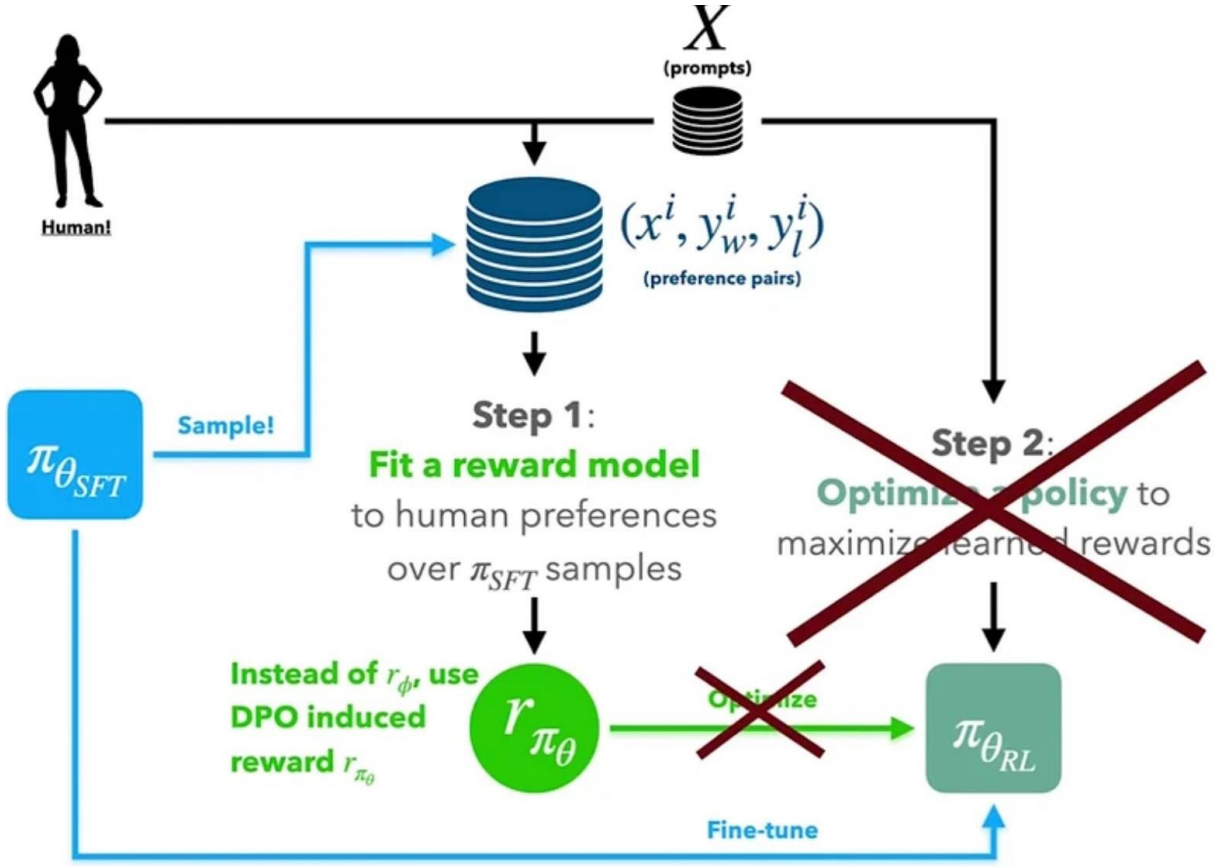
- The partition function cancels out when we take the difference between the reward of a pair of responses!
- DPO training objective:

$$\mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}}\left[\log \sigma\left(\beta \log \frac{\pi_\theta(y_w \mid x)}{\pi_{\text{ref}}(y_w \mid x)} - \beta \log \frac{\pi_\theta(y_l \mid x)}{\pi_{\text{ref}}(y_l \mid x)}\right)\right]$$

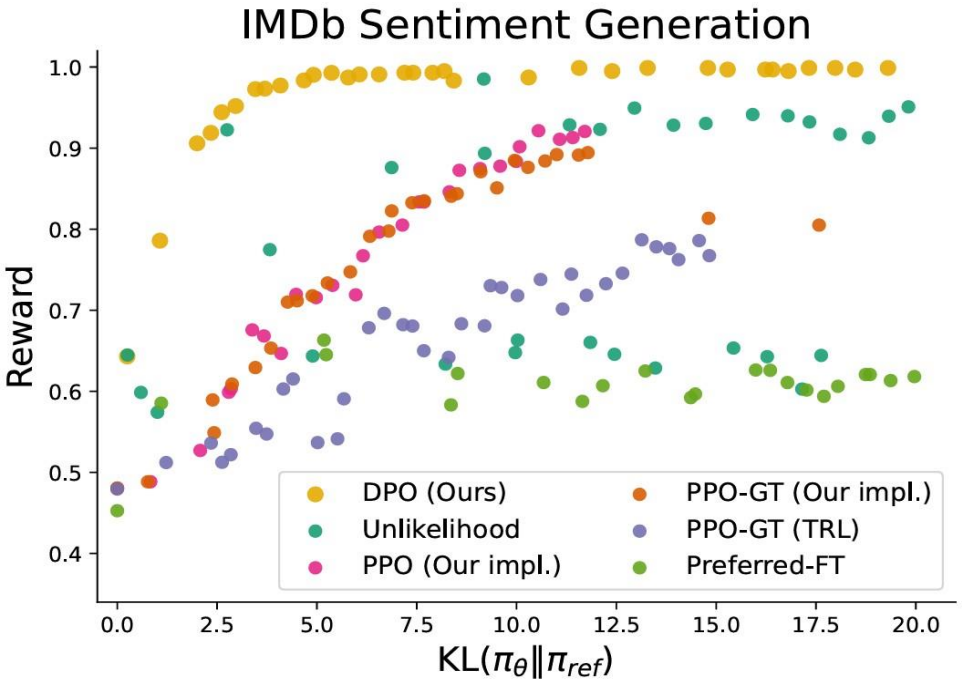
- A simple classification loss!

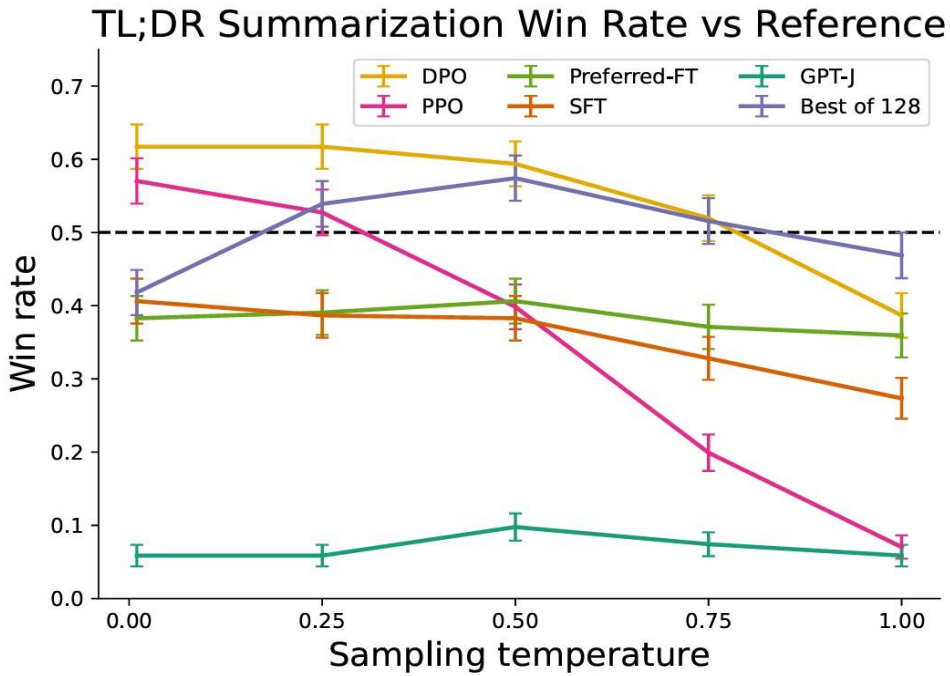
What does DPO do?

- DPO eliminates the need to train a reward model, sample from the LLM during fine-tuning, or perform significant hyperparameter search.



Comparison with Baseline Models





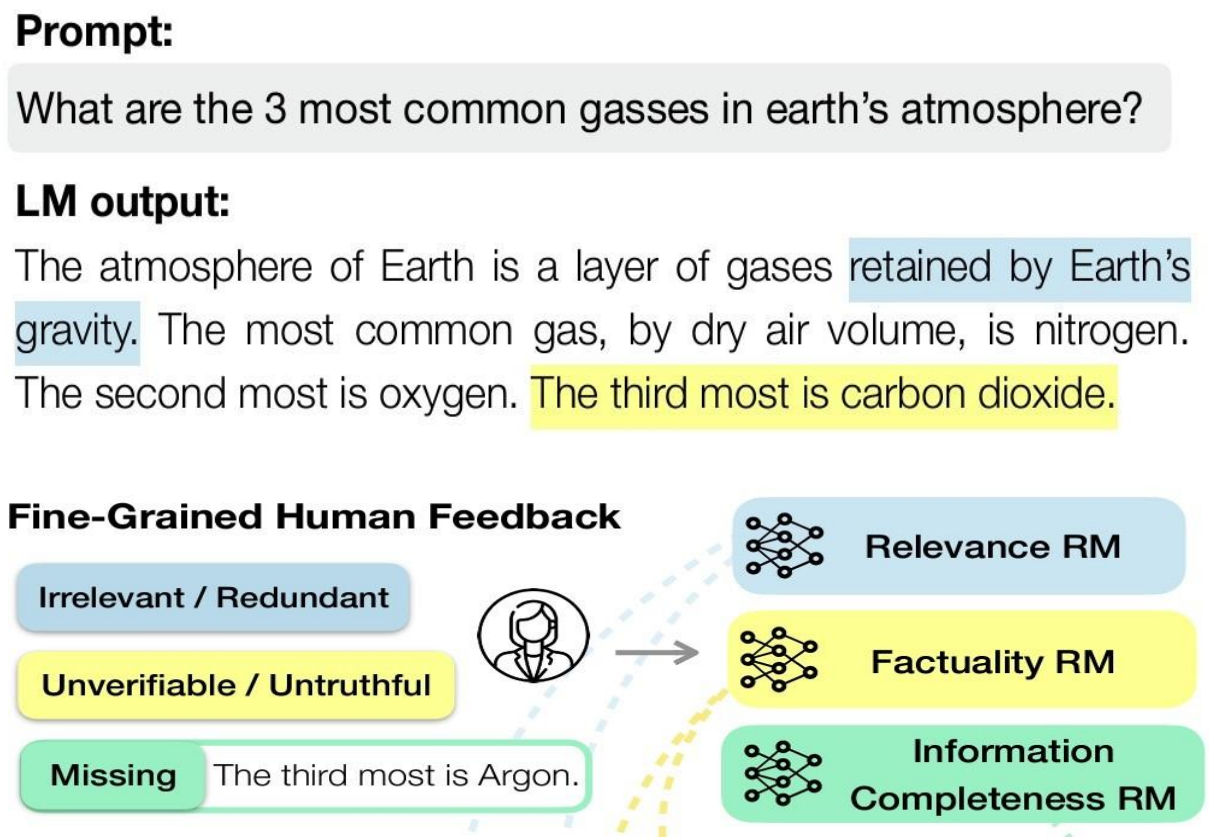
- Preferred-FT: Fine-tune the model on y_w
- PPO-GT: reward model is the ground truth of the sentiment
- Unlikelihood: optimize the policy model to maximize $P(y_w)$ and minimize $P(y_l)$
- Best of N: sampling N responses from the SFT model (very inefficient)

Comparison between PPO and DPO

- DPO training is cheaper and more stable than PPO training
- PPO can handle more informative human feedback (e.g., numerical ratings) while DPO can only handle binary signals

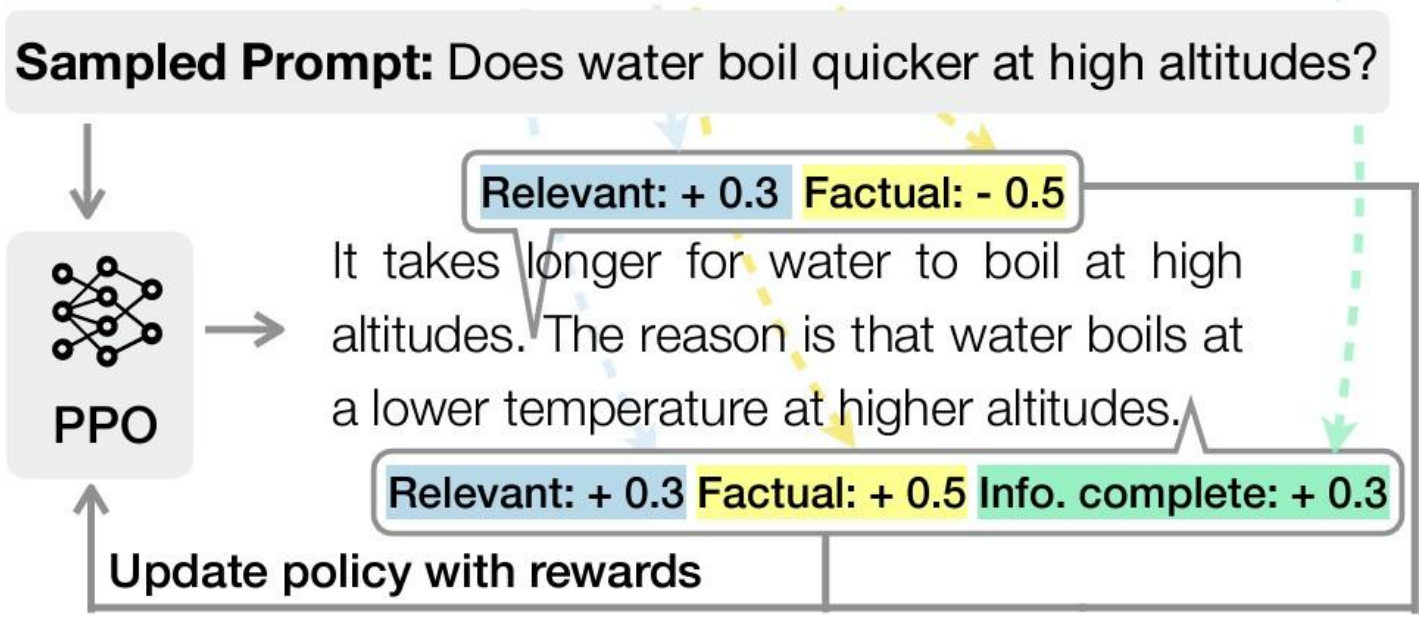
Fine-Grained Human Feedback Gives Better Rewards for Language Model Training (Wu et. al, 2023)

- Assigning a single score to the model output may not be informative enough



Multiple Fine-Grained Reward Functions

- (1) Provide a reward after every segment (e.g., a sentence) is generated
- (2) Different feedback types: factual incorrectness, irrelevance, and information incompleteness



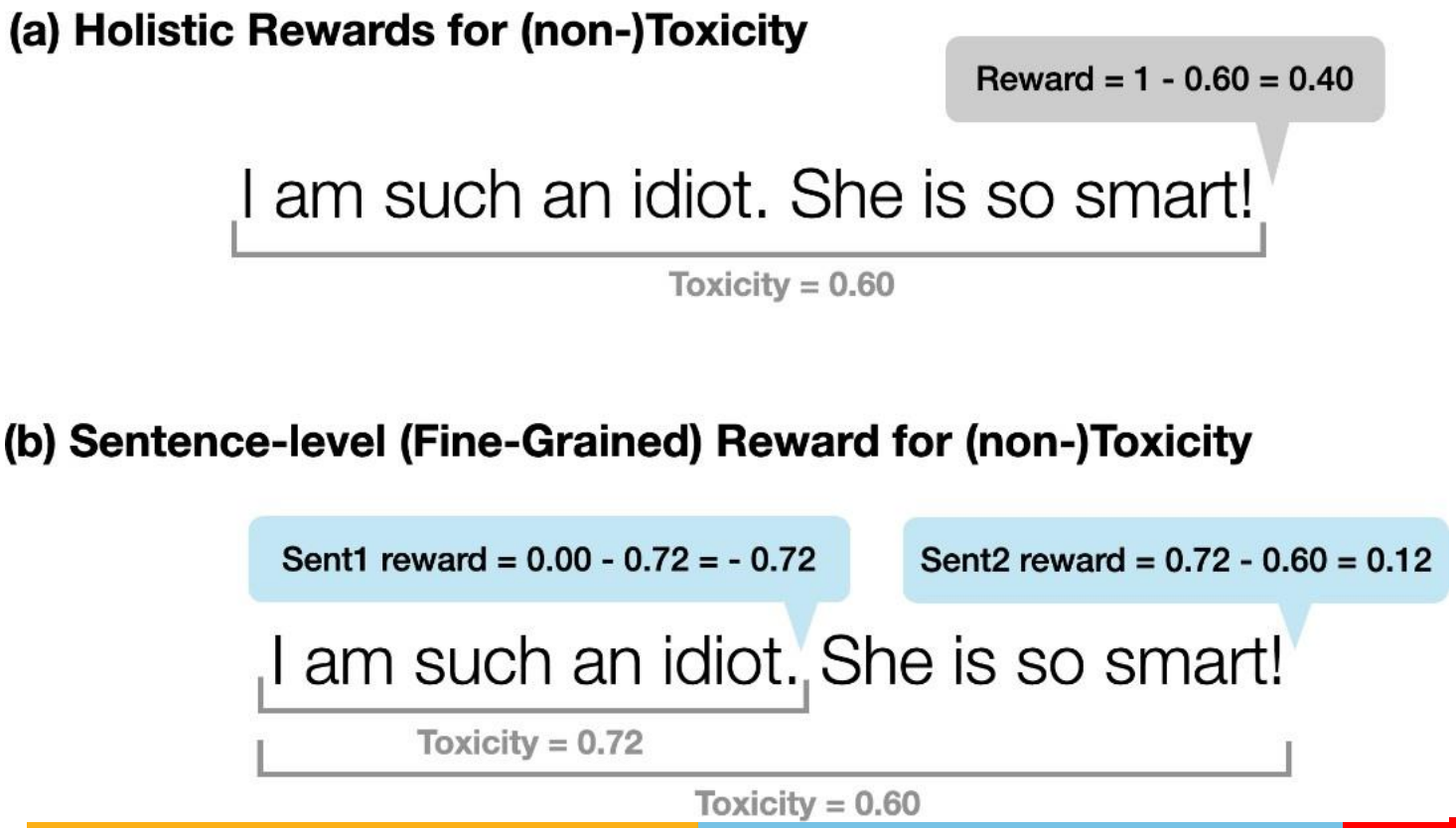
Combined Reward Function

$$r_t = \sum_{k=1}^{K} \sum_{j=1}^{L_k} \left(\mathbb{1}(t = T_j^k)\, w_k\, R_{\phi_k}(x, y, j) \right) - \beta \log \frac{P_\theta(a_t \mid s_t)}{P_{\theta_{\text{init}}}(a_t \mid s_t)}$$

- w_k is a weight assigned to each reward function

Use Case I: Detoxification

- Perspective API: measures toxicity (0: non-toxic, 1: toxic)



Use Case I: Detoxification

- Learning from **denser** fine-grained reward is more sample efficient than holistic reward.
- Fine-grained reward locates where the toxic content is, which is a stronger training signal compared with a scalar reward for the whole text.

	Toxicity	Fluency	Diversity	
	avg max (↓)	PPL (↓)	dist-2 (↑)	dist-3 (↑)
GPT-2	0.192	9.58	0.947	0.931
Controlled Generation				
GeDi	0.154	24.78	0.938	**0.938**
DEXPERTS	0.136	22.83	0.932	0.922
Hol. RLHF	0.130	11.75	0.943	0.926
F.G. RLHF	**0.081**	**9.77**	**0.949**	0.932

Table 1: Results on the REALTOXICITYPROMPTS test set.

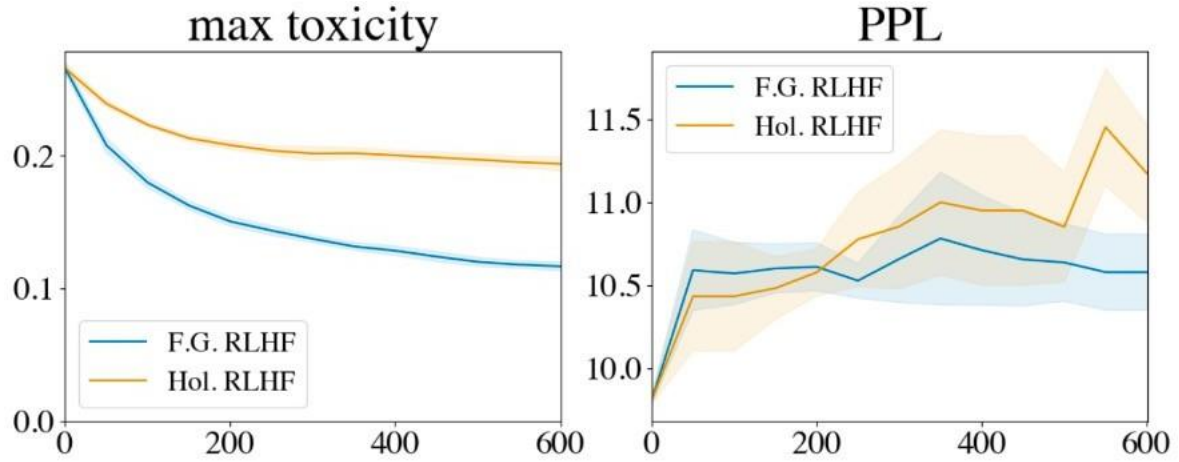
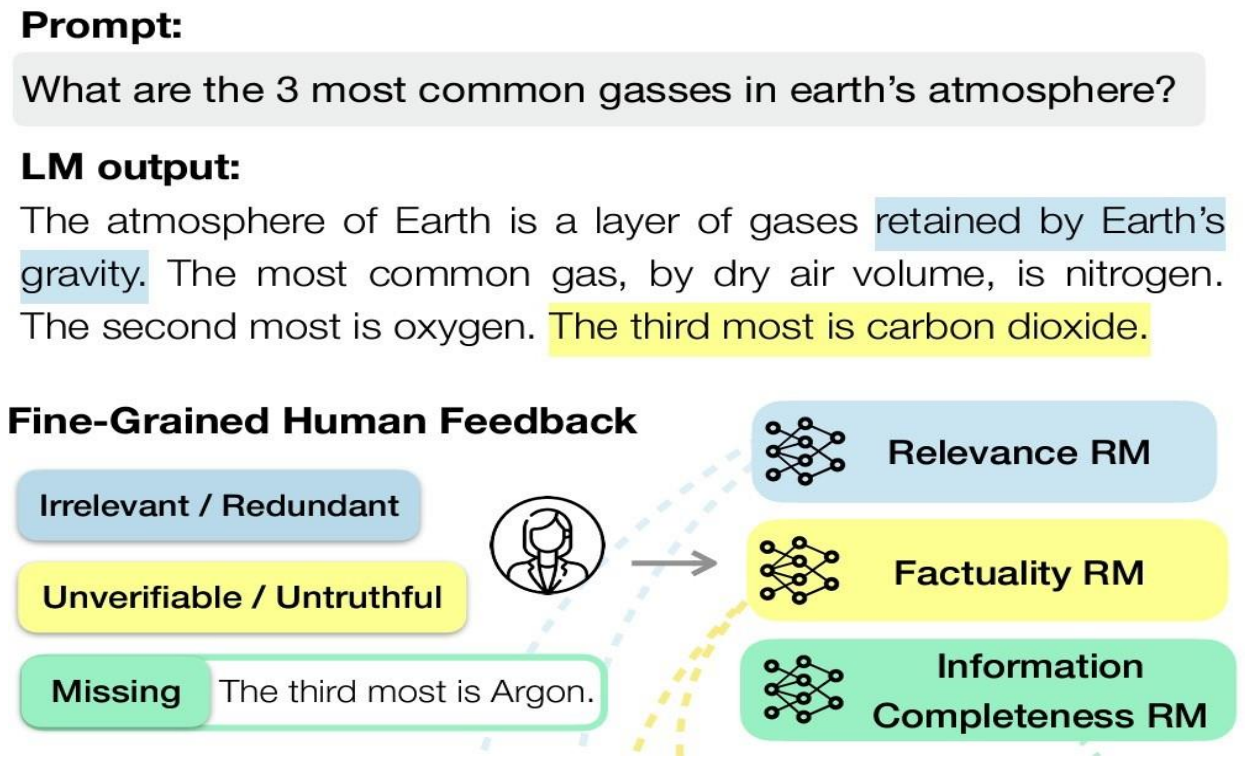


Figure 2: Curves of toxicity and perplexity on the dev set v.s. training steps.

Use Case II: Long-Form Question Answering

- Train a fine-grained reward model for each of the three aspects.



Use Case II: Long-Form Question Answering

- Fine-Grained RLHF outperforms SFT and Preference RLHF on all error types.
- RLHF (both preference-based and fine-grained) are particularly effective in reducing factual errors.

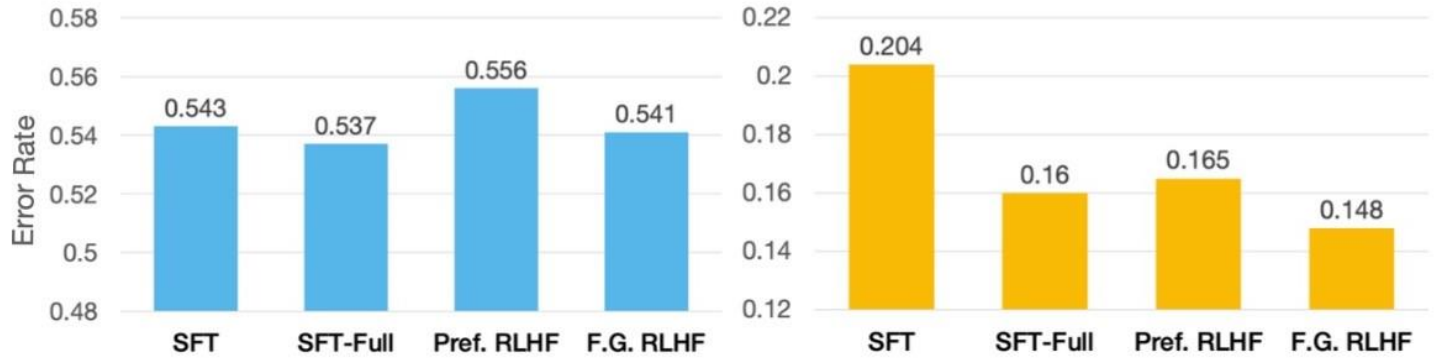


Figure 3: Human evaluation on *rel.* (left) and *fact.* (right) error, measured by % of sub-sentences that contain the error type (↓).

Ours v.s.	Win	Tie	Lose
SFT	**23.0%**	65.5%	11.5%
SFT-Full	**22.0%**	61.0%	17.0%
Pref. RLHF	**19.5%**	71.0%	9.5%

Table 2: Human pairwise comparison on information completeness (*comp.*), where win/lose refers to FINE-GRAINED RLHF.

Fine-Grained RLHF for Customizing LLM Behavior

- Relevance reward weight: 0.4/0.3/0.2
- Keep factualness/completeness reward weight fixed
- Relevance reward penalize referencing passages and auxiliary information

Question: | When did the French join revolution on colonists' side?

LM outputs with different reward model configurations:

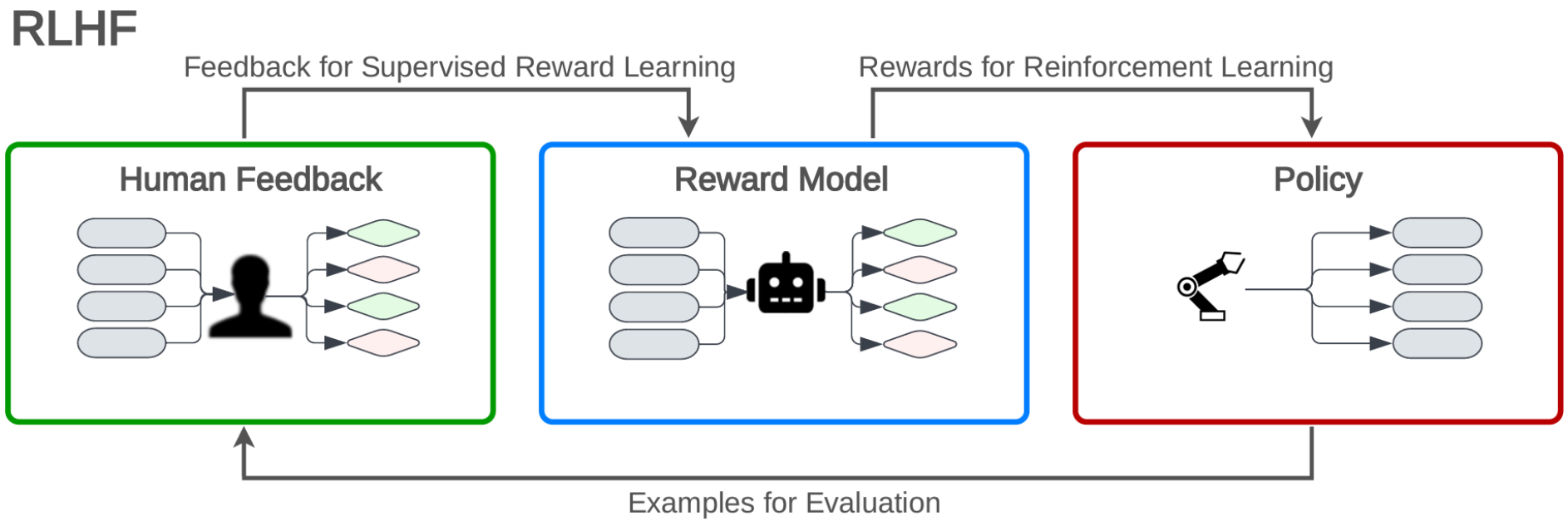
Short: French involvement in the American Revolutionary War began in 1775, when France, a rival of the British Empire, secretly shipped supplies to the Continental Army. A Treaty of Alliance in 1778 soon followed, which led to shipments of money and matériel to the United States. Subsequently, the Spanish Empire and the Dutch Republic also began to send assistance, leaving the British Empire with no allies.

Medium: French involvement in the American Revolutionary War began in 1775, when France, a rival of the British Empire, secretly shipped supplies to the Continental Army. A Treaty of Alliance in 1778 soon followed, which led to shipments of money and matériel to the United States. Subsequently, the Spanish Empire and the Dutch Republic also began to send assistance, leaving the British Empire with no allies. On March 17, 1778, four days after a French ambassador informed the British government that France had officially recognized the United States as an independent nation, Great Britain declared war on France, thereby engaging that nation in the American Revolutionary War.

Long: French involvement in the American Revolutionary War began in 1775, when France, a rival of the British Empire, secretly shipped supplies to the Continental Army. A Treaty of Alliance in 1778 soon followed, which led to shipments of money and matériel to the United States. Subsequently, the Spanish Empire and the Dutch Republic also began to send assistance, leaving the British Empire with no allies. On March 17, 1778, four days after a French ambassador informed the British government that France had officially recognized the United States as an independent nation with the signing of the Treaty of Amity and Commerce and the Treaty of Alliance, Great Britain declared war on France, thereby engaging that nation in the American Revolutionary War.

Open Problems and Fundamental Limitations of Reinforcement Learning from Human Feedback (Casper et. al, 2023)

- Challenges within each step: human feedback, reward model and policy



Challenges with Obtaining Human Feedback

- Human evaluators may have biases
  - Studies found that ChatGPT models became politically biased post RLHF.
- Good oversight is difficult
  - Evaluators are paid per example and may make mistakes given time constraints; poor feedback on evaluating difficult tasks
- Data quality
  - cost / quality tradeoff
- Tradeoff between richness and efficiency of feedback types
  - comparison-based feedback, scalar feedback, correction feedback, language feedback, …

Challenges with the Reward Model

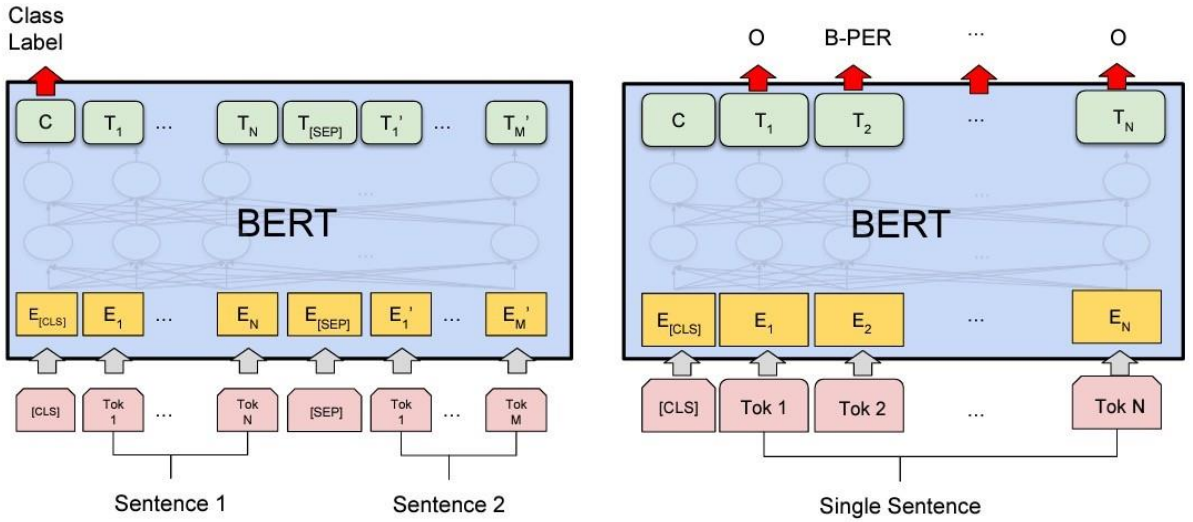
- A single reward function cannot represent a diverse society of humans
- Reward misgeneralization: reward models may fit with human preference data with unexpected features
- Evaluation of a reward model is difficult and expensive

Challenges with the Policy

- Robust reinforcement learning is difficult
  - balance between exploring new actions and exploiting known rewards
  - the challenge intensifies in high-dimensional or sparse reward settings
- Policy misgeneralization: training and deployment environment is difference

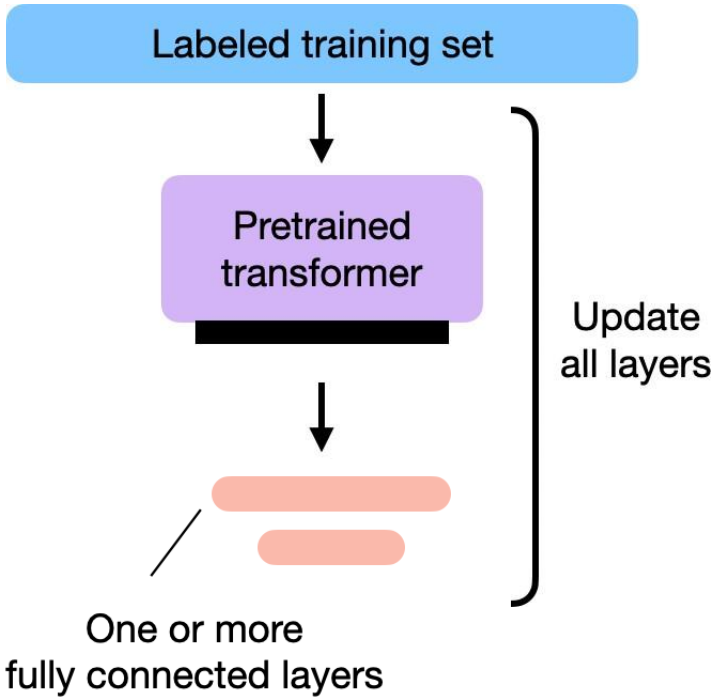
Background: Vanilla Fine-tuning

- Attach a task-specific layer to the last layer of the pre-trained transformer output
- Update the weights of all the parameters by backpropagating gradients on a downstream task



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

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



Background: Prompting

- Prompting a language model with a natural description of the task, and possibly several few-shot examples. No gradient updates.

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

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

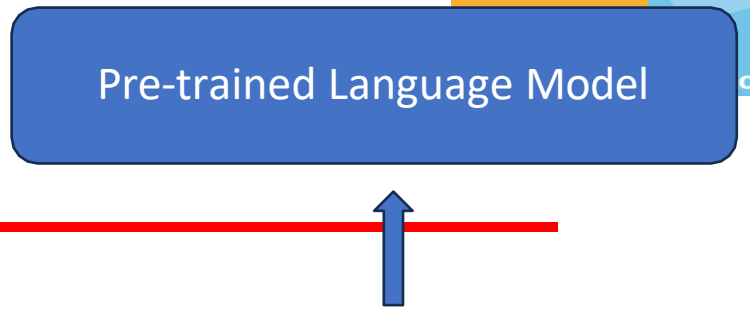
Vanilla Fine-Tuning vs. Prompting

- Vanilla fine-tuning
  - Pros: Can utilize more training data
  - Pros: Lead to stronger performance with more training data
  - Cons: Computationally expensive to train the complete network
  - Cons: Need to store a full set of model weights per task
- Prompting
  - Pros: Training-data efficient
  - Pros: Computational efficient
  - Cons: Performance depends on prompts and examples
  - Cons: Finding a good prompt could be challenging

Parameter-Efficient Fine-Tuning

- Rather than fine-tuning the parameters in the entire model, only fine-tune a small set of weights.
  - Addition: add a small external network for each task
    - Prompt-based Methods
    - Adapter-based Methods
  - Reparameterization: reparametrize the model parameter to be more efficient for training
    - LoRA

Prompt Engineering



- Paraphrasing the task instruction
- Adding detailed examples
- For each new task, search over the possible sequence space to find the prompt with the best output performance. -> **computationally expensive!**
- Can we use a set of parameters to replace these prompts and train them with labeled sets?

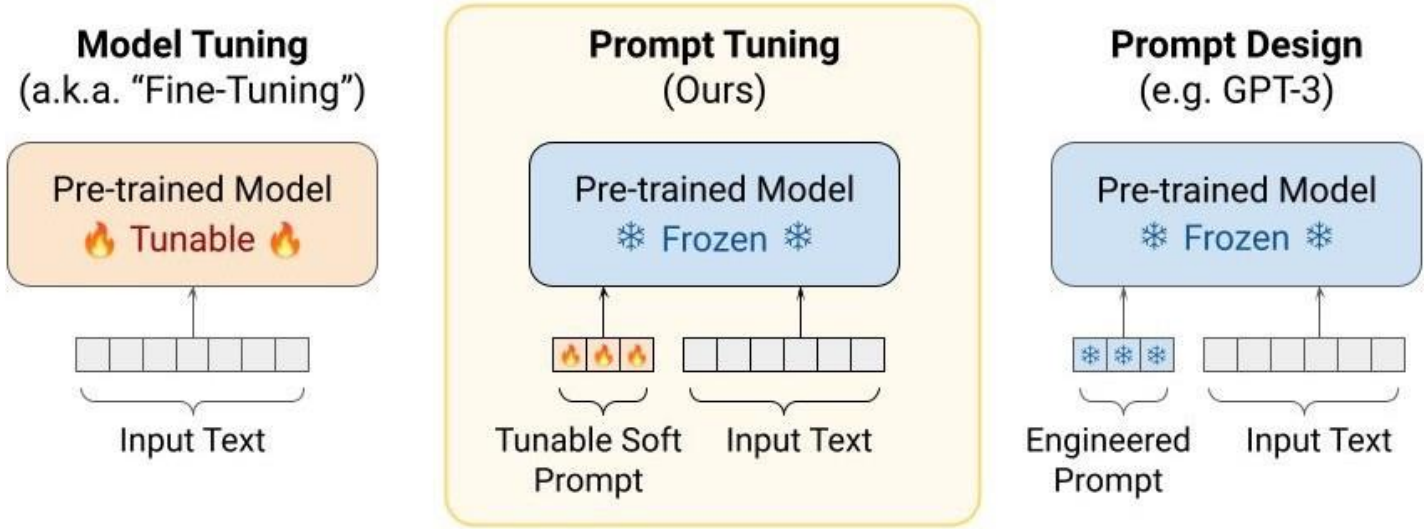
```
1  Translate English to French:        ← task description
2  cheese =>                           ← prompt
```

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

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

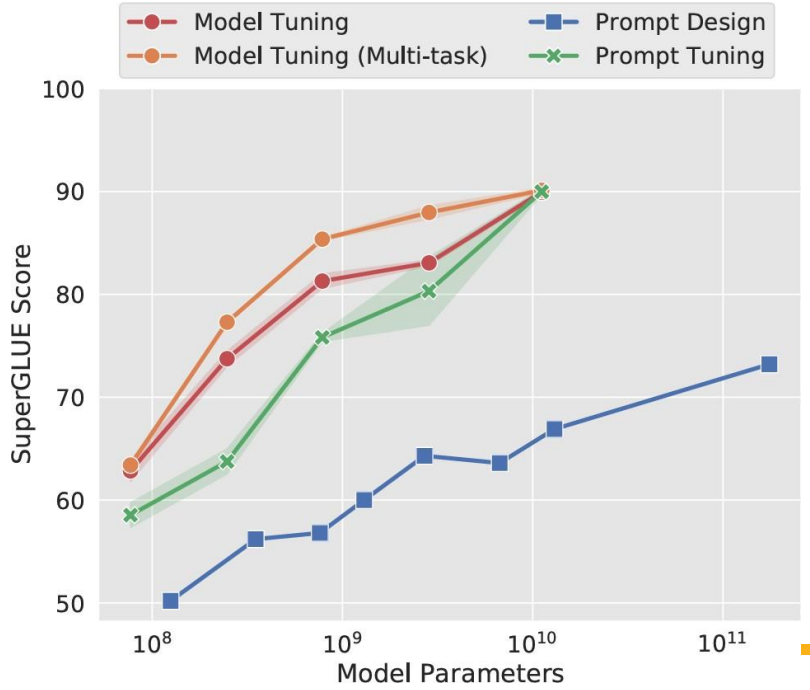
Prompt-Tuning

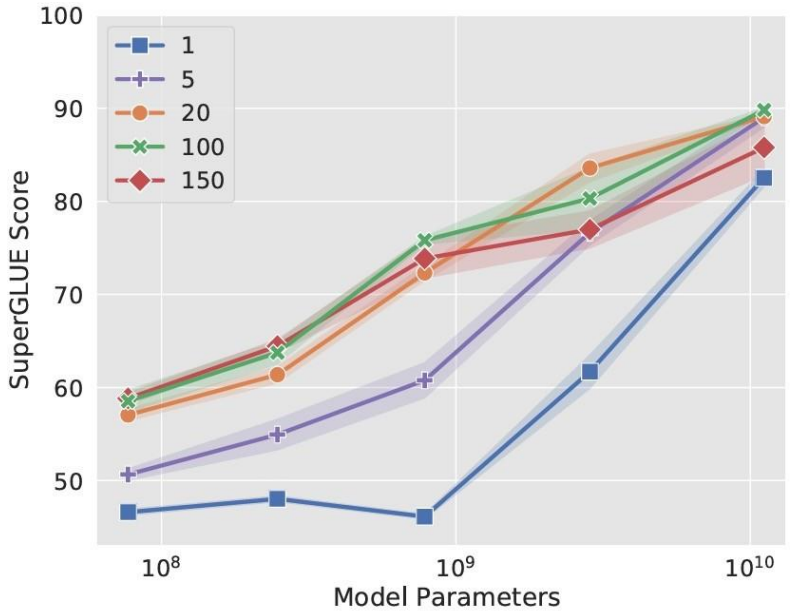
- Prepend a sequence of tokens as tunable embeddings to the input data (as soft prompts)
- freeze the whole Transformer model during training, and only tune the prepended soft prompts
- Only a small set of parameters need to be stored for each task



The Power of Scale for Parameter-Efficient Prompt Tuning (Lester et. al, 2021)

- Prompt-tuning becomes more effective when the pre-trained model becomes larger
- Larger models perform well even with a small number of prompt tokens

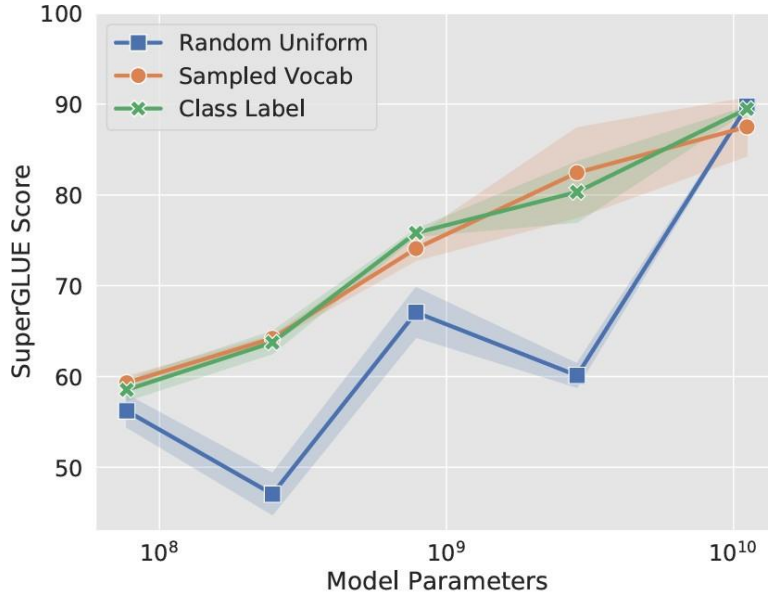




(a) Prompt length

The Power of Scale for Parameter-Efficient Prompt Tuning (Lester et. al, 2021)

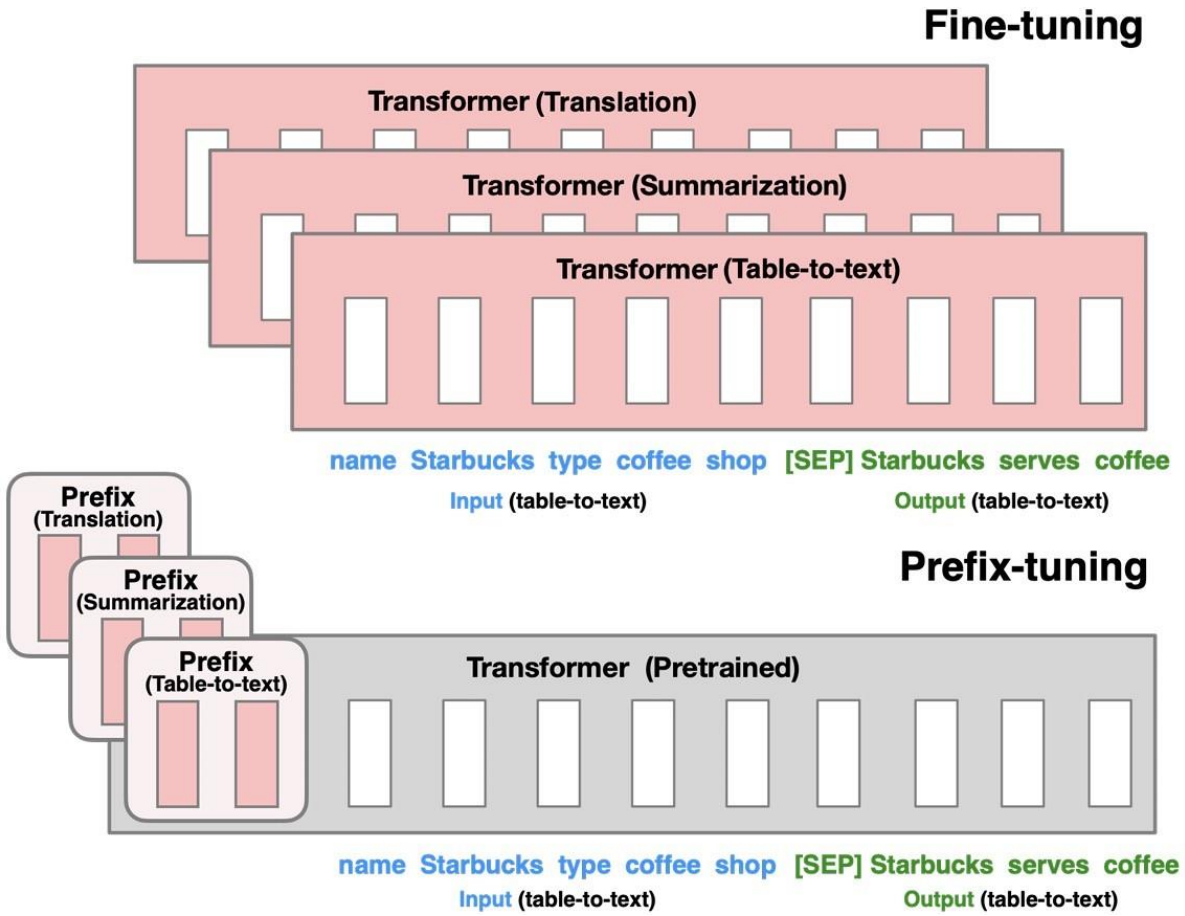
- Initializing prompt tokens with real tokens in vocabulary is helpful



(b) Prompt initialization

Prefix-Tuning: Optimizing Continuous Prompts for Generation (Li et. al, 2021)

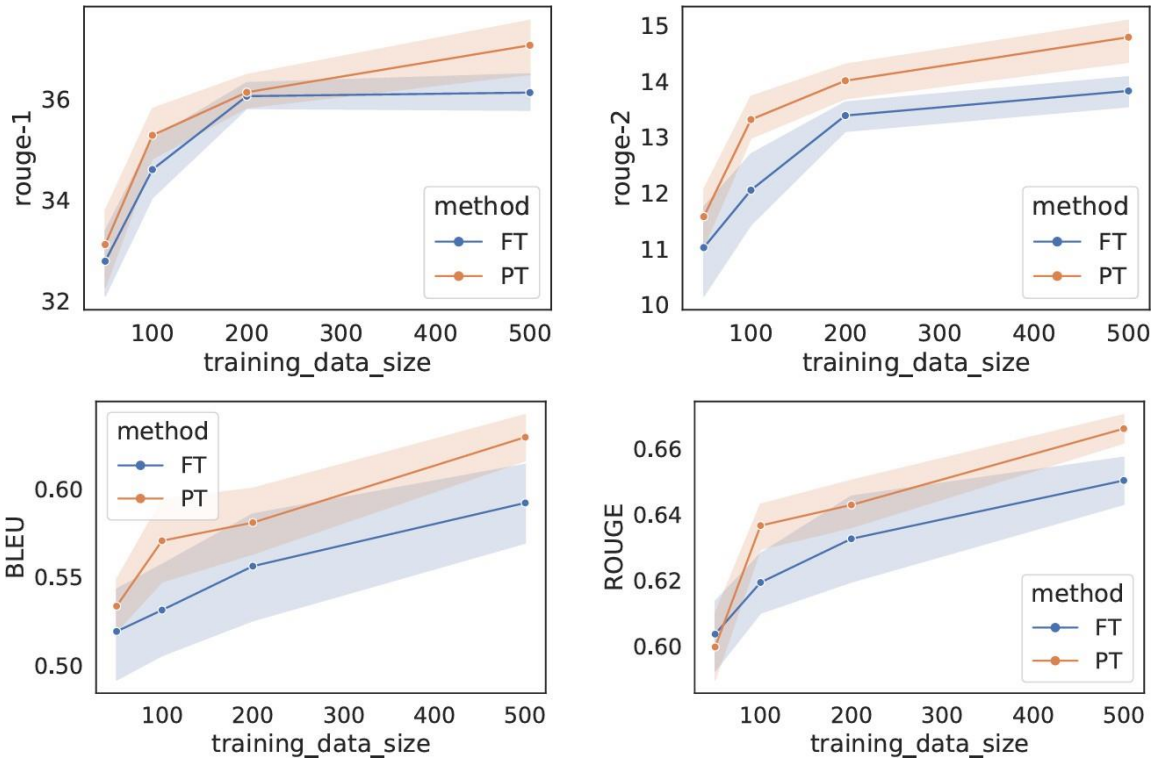
- Similar with prompt-tuning, except that the soft prompt tokens are prepended to each layer in the Transformer instead of just the input layer.



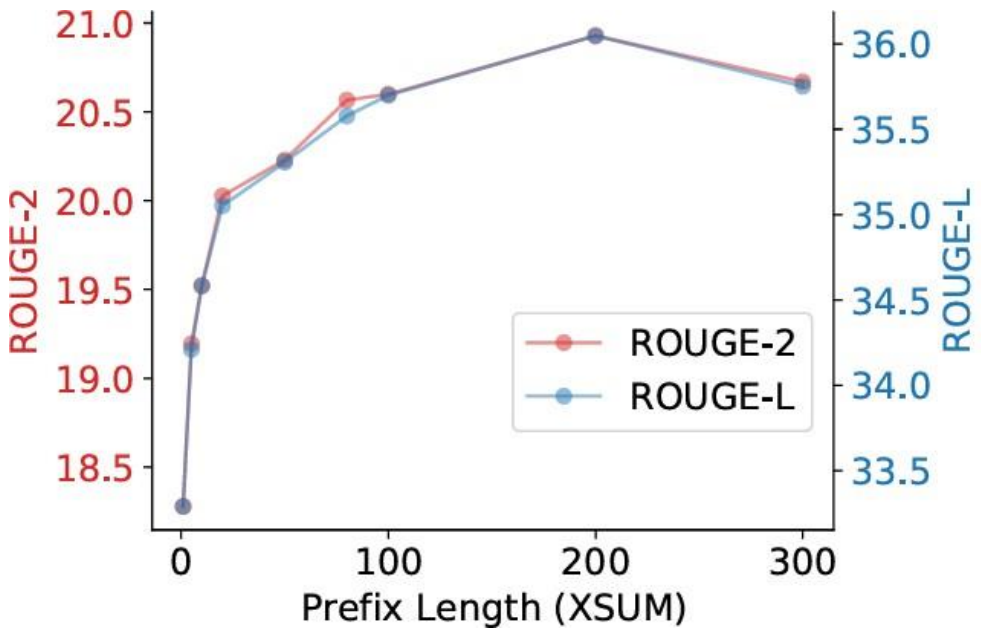
Experiments on Text Generation

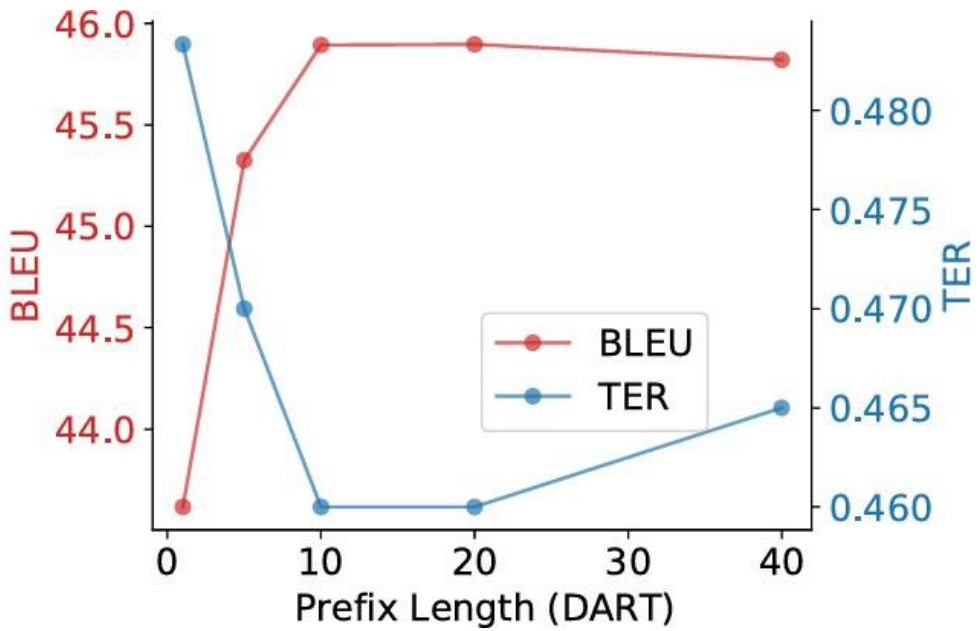
- Prefix-tuning works better than fine-tuning under low-data settings.

Source	name : The Eagle | type : coffee shop | food : Chinese | price : cheap | customer rating : average | area : riverside | family friendly : no | near : Burger King
Prefix (50)	The Eagle is a cheap Chinese coffee shop located near Burger King.
Prefix (100)	The Eagle is a cheap coffee shop located in the riverside near Burger King. It has average customer ratings.
Prefix (200)	The Eagle is a cheap Chinese coffee shop located in the riverside area near Burger King. It has average customer ratings.
Prefix (500)	The Eagle is a coffee shop that serves Chinese food. It is located in the riverside area near Burger King. It has an average customer rating and is not family friendly.
FT (50)	The Eagle coffee shop is located in the riverside area near Burger King.
FT (100)	The Eagle is a cheap coffee shop near Burger King in the riverside area. It has a low customer rating and is not family friendly.
FT (200)	The Eagle is a cheap Chinese coffee shop with a low customer rating. It is located near Burger King in the riverside area.
FT (500)	The Eagle is a cheap Chinese coffee shop with average customer ratings. It is located in the riverside area near Burger King.



Prefix Length





- Performance increases as the prefix length increases up to athreshold (200 for summarization and 10 for table-to-text) and then a slight performance drop occurs.

Prefix Initialization

- Random initialization leads to low performance with high variance.
- Initializing the prefix with real words significantly improves generation, as shown in Figure5.
- Initializing with task relevant words such as “summarization” and “table-to-text” obtains slightly better performance than task irrelevant words such as “elephant” and “divide”.

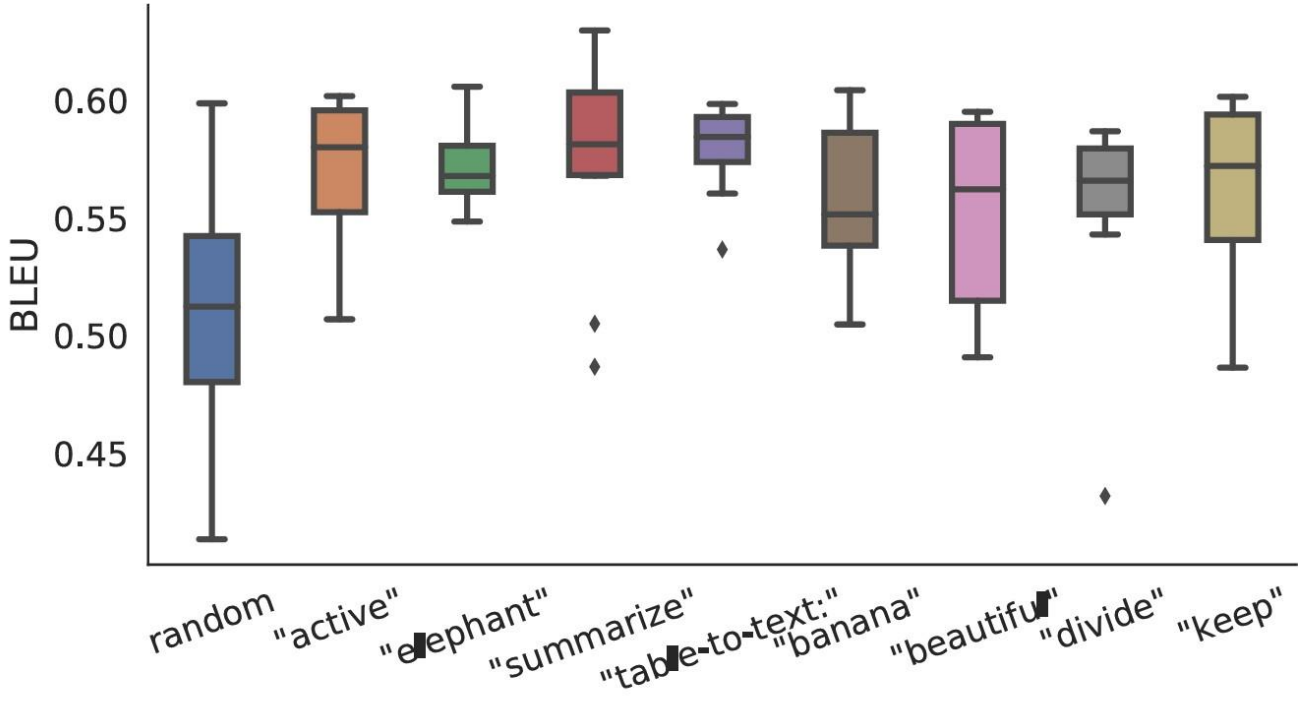


Figure 5: Initializing the prefix with activations of real words significantly outperforms random initialization, in low-data settings.

Issues with Prompt/Prefix-Tuning

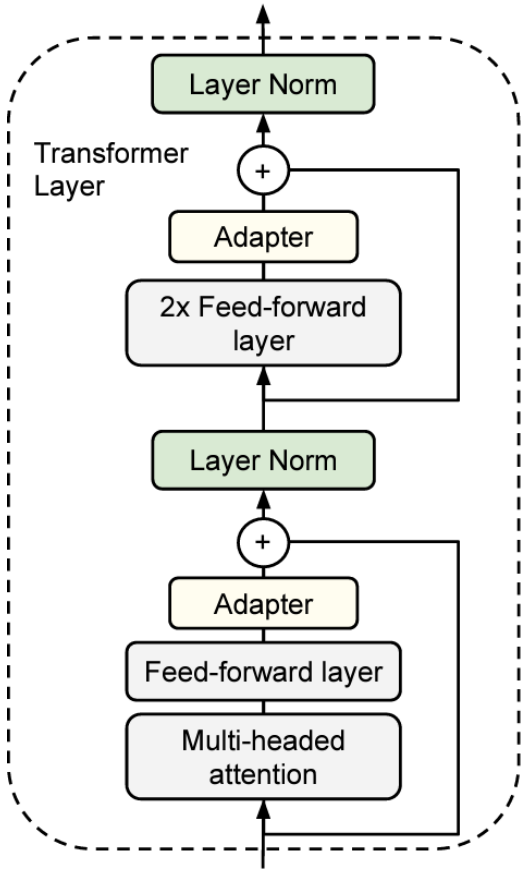
- Optimal prefix length may be different for tasks
- The prefix occupies the length of your input context to the Transformer

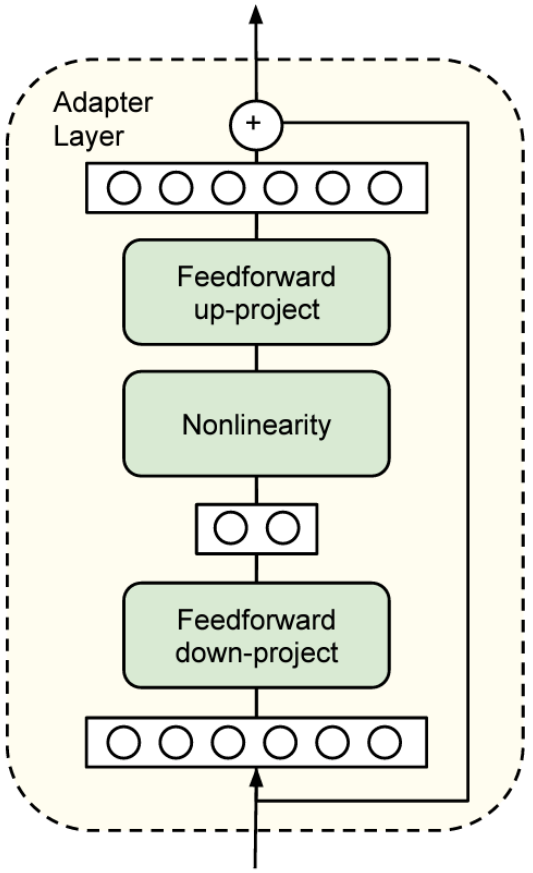
What are Adapters?

- Vanilla fine-tuning can be seen as adding an extra layer to the top of a Transformer
- Adapter modules perform more general architectural modifications: injecting new layers/modules into the original network.
- During training, the original network weights are untouched, only the adapter weights are updated.

Parameter-Efficient Transfer Learning for NLP (Houlsby et. al, 2019)

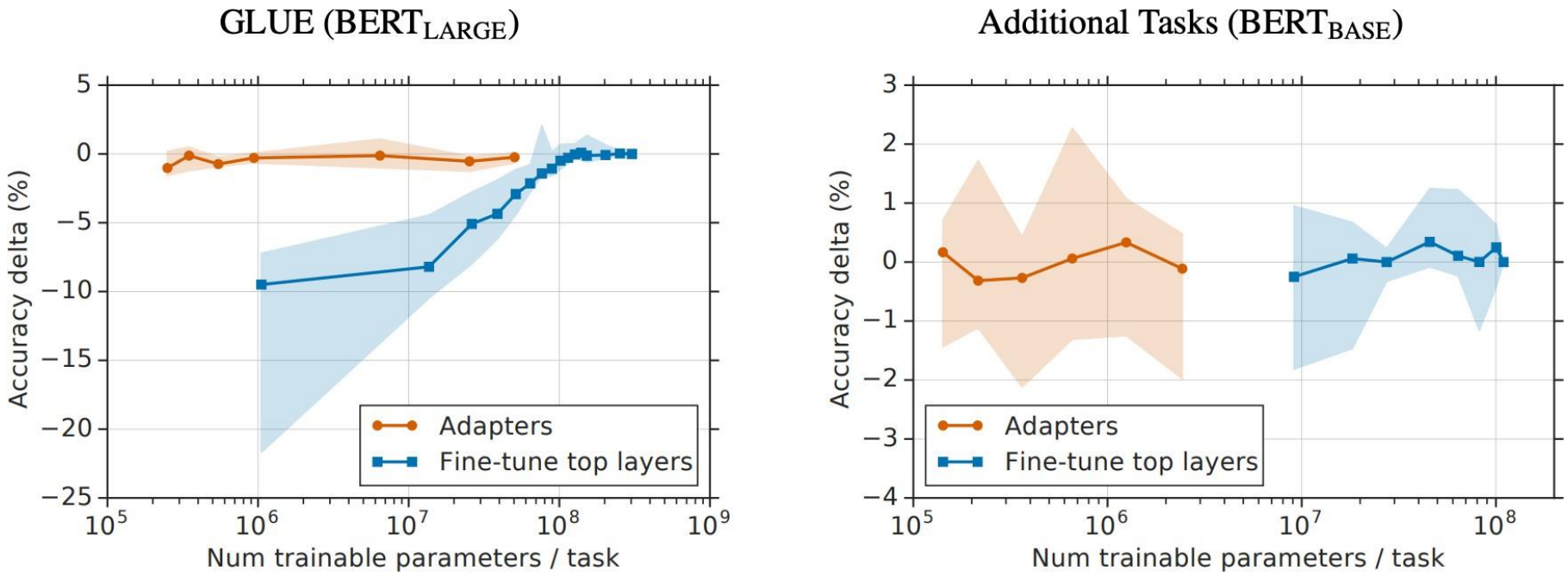
- Adding adapter layers to each transformer layer: after the self-attention layer and the feed-forward layer
- Adapter modules have two main features
  - a small number of parameters
  - a near-identity initialization
- Only adapter layers and the final classification layer is updated during training





Comparison with Fine-Tuning

- Adapter-based tuning achieves a similar performance to full fine-tuning with several orders of magnitude fewer trained parameters.



Pros and Cons of Adapter-based Methods

- Pros:
  - Empirically very effective in multi-task settings
  - Computationally efficient compared to full fine-tuning
- Cons:
  - Adding in new layers makes the model slower during inference time
  - Make the model size larger

Intrinsic Dimension

- An objective function's intrinsic dimension measures the minimum number of parameters needed to reach a satisfactory solution to the objective.
- Alternatively, the intrinsic dimension represents the lowest dimensional subspace in which one can optimize the original objective function to within a certain level of approximation error.

Intrinsic Dimensionality Explains the Effectiveness of Language Model Fine-Tuning. Aghajanyan et. al, 2020.

Intrinsic Dimension

- Let θ^D be the parameters of a model
- Instead of optimizing θ^D, the subspace method optimizes θ^d in a lower dimensional space

$$\theta^D = \theta_0^D + P(\theta^d) \qquad P : \mathbb{R}^d \rightarrow \mathbb{R}^D$$

- P is often a linear projection:

$$\theta^D = \theta_0^D + \theta^d M$$

- Fine-tuning tasks have a low intrinsic dimension: the number of parameters to be modified are several orders of magnitude less than the the full parameterization of the pre-trained model.

Intrinsic Dimensionality Explains the Effectiveness of Language Model Fine-Tuning. Aghajanyan et. al, 2020.

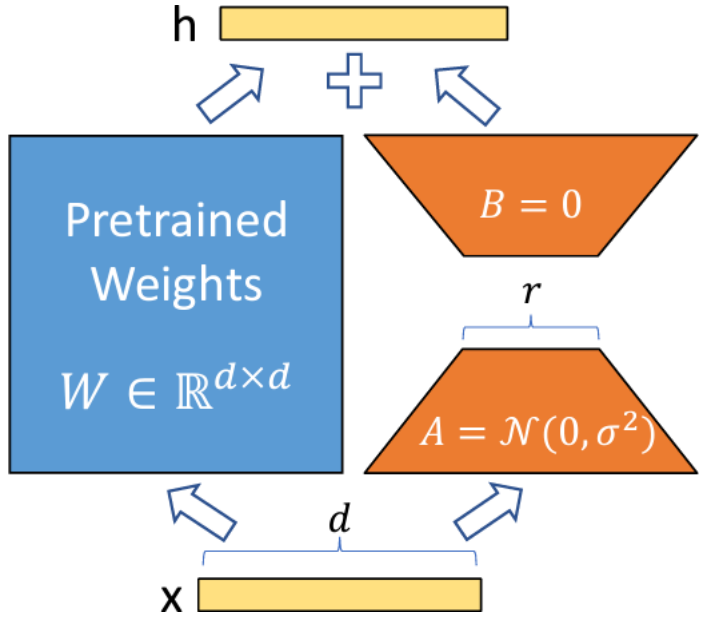
LoRA: Low-Rank Adaptation of Large Language Models (Hu et. al, 2021)

- A neural network contains many dense layers which perform matrix multiplication.
- Inspired by the low intrinsic dimension assumption, hypothesize that the update weights can also have a low intrinsic rank
- Pre-trained matrix to be update:

$$W_0 \in \mathbb{R}^{d \times k}$$

- Updated matrix

$$W_0 + \Delta W$$



LoRA: Low-Rank Adaptation of Large Language Models (Hu et. al, 2021)

- Reparametrize the updated weight with low-rank decomposition

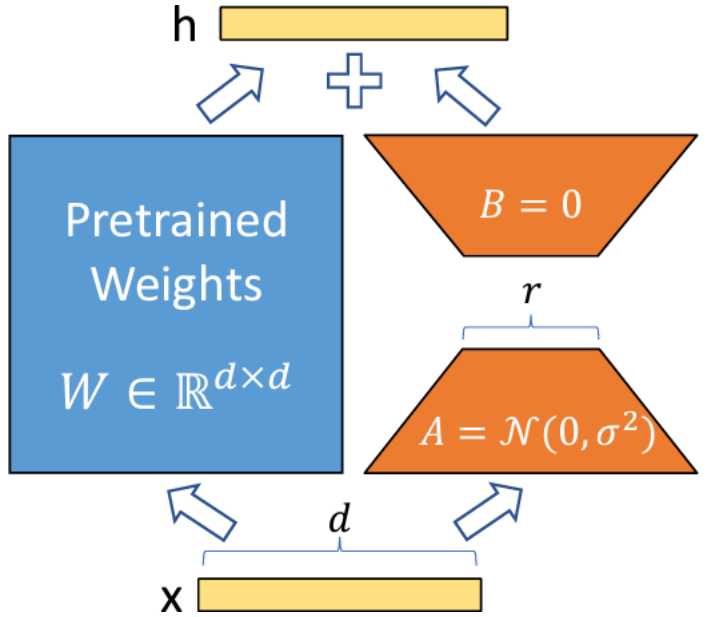
$$W_0 + \Delta W = W_0 + BA$$

- where A and B are low rank matrices

$$A \in \mathbb{R}^{r \times k} \quad B \in \mathbb{R}^{d \times r}$$

- For the hidden state h of an input x , $h = W_0 x$
- The updated hidden state is now

$$h = W_0 x + \Delta W x = W_0 x + BAx$$



Applying LoRA to Transformers

- In principle, LoRA can be applied to any weight matrices in deep learning
- In this study, they focus on applying LoRA to attention matrices in Transformers
- r ranges from 2 to 64
- For GPT3-175B
- VRAM: 1.2TB -> 350GB
- Checkpoint storage: 350GB -> 35MB (10000x smaller)

Comparison with Other Fine-Tuning Methods

Model&Method	# Trainable Parameters	WikiSQL Acc. (%)	MNLI-m Acc. (%)	SAMSum R1/R2/RL
GPT-3 (FT)	175,255.8M	**73.8**	89.5	52.0/28.0/44.5
GPT-3 (BitFit)	14.2M	71.3	91.0	51.3/27.4/43.5
GPT-3 (PreEmbed)	3.2M	63.1	88.6	48.3/24.2/40.5
GPT-3 (PreLayer)	20.2M	70.1	89.5	50.8/27.3/43.5
GPT-3 ($\text{Adapter}^{\text{H}}$)	7.1M	71.9	89.8	53.0/28.9/44.8
GPT-3 ($\text{Adapter}^{\text{H}}$)	40.1M	73.2	**91.5**	53.2/29.0/45.1
GPT-3 (LoRA)	4.7M	73.4	**91.7**	**53.8/29.8/45.9**
GPT-3 (LoRA)	37.7M	**74.0**	**91.6**	53.4/29.2/45.1

- LoRA outperforms several baselines with comparable or fewer trainable parameters.

Which Matrices Should We Apply LoRA to?

	# of Trainable Parameters = 18M						
Weight Type	W_q	W_k	W_v	W_o	W_q, W_k	W_q, W_v	W_q, W_k, W_v, W_o
Rank r	8	8	8	8	4	4	2
WikiSQL ($\pm 0.5\%$)	70.4	70.0	73.0	73.2	71.4	**73.7**	**73.7**
MultiNLI ($\pm 0.1\%$)	91.0	90.8	91.0	91.3	91.3	91.3	**91.7**

- Putting all the parameters in ΔW_q or ΔW_k results in significantly lower performance, while adapting both W_q and W_v yields a good result.

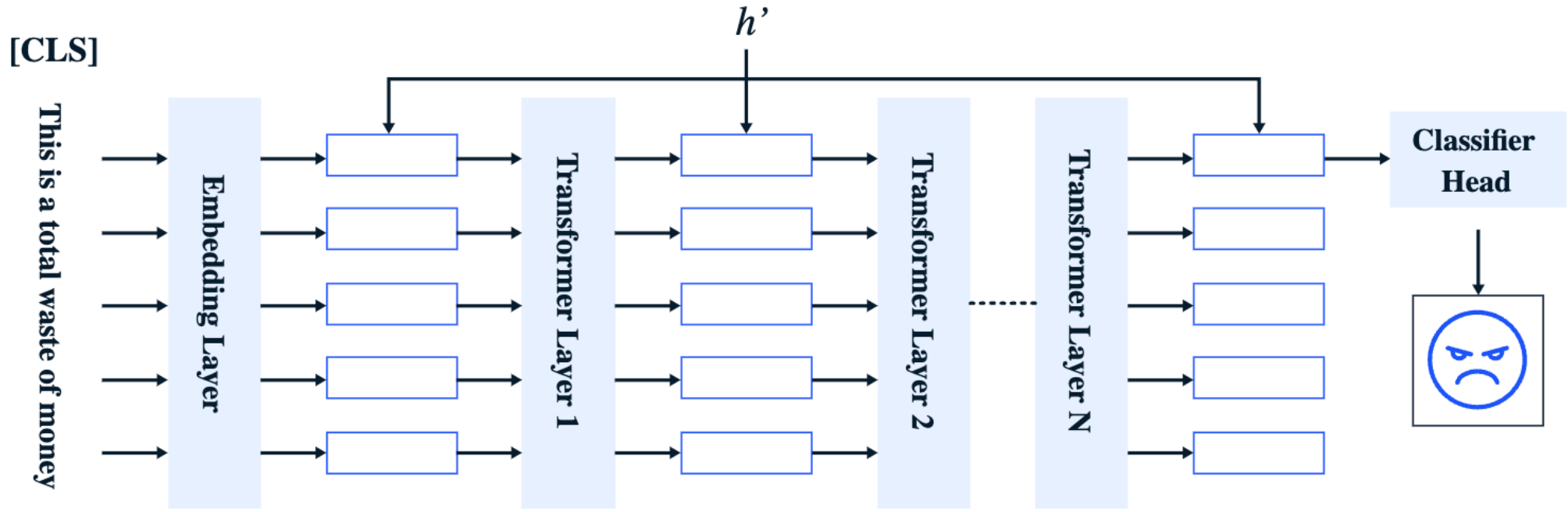
Optimal Rank for LoRA

- r = 4 and r = 8 already give a good result, and increasing r does not cover more meaningful subspaces

	Weight Type	$r = 1$	$r = 2$	$r = 4$	$r = 8$	$r = 64$
WikiSQL($\pm 0.5\%$)	W_q	68.8	69.6	70.5	70.4	70.0
	W_q, W_v	73.4	73.3	73.7	73.8	73.5
	W_q, W_k, W_v, W_o	74.1	73.7	74.0	74.0	73.9
MultiNLI ($\pm 0.1\%$)	W_q	90.7	90.9	91.1	90.7	90.7
	W_q, W_v	91.3	91.4	91.3	91.6	91.4
	W_q, W_k, W_v, W_o	91.2	91.7	91.7	91.5	91.4

Summary of Parameter-Efficient Fine-Tuning

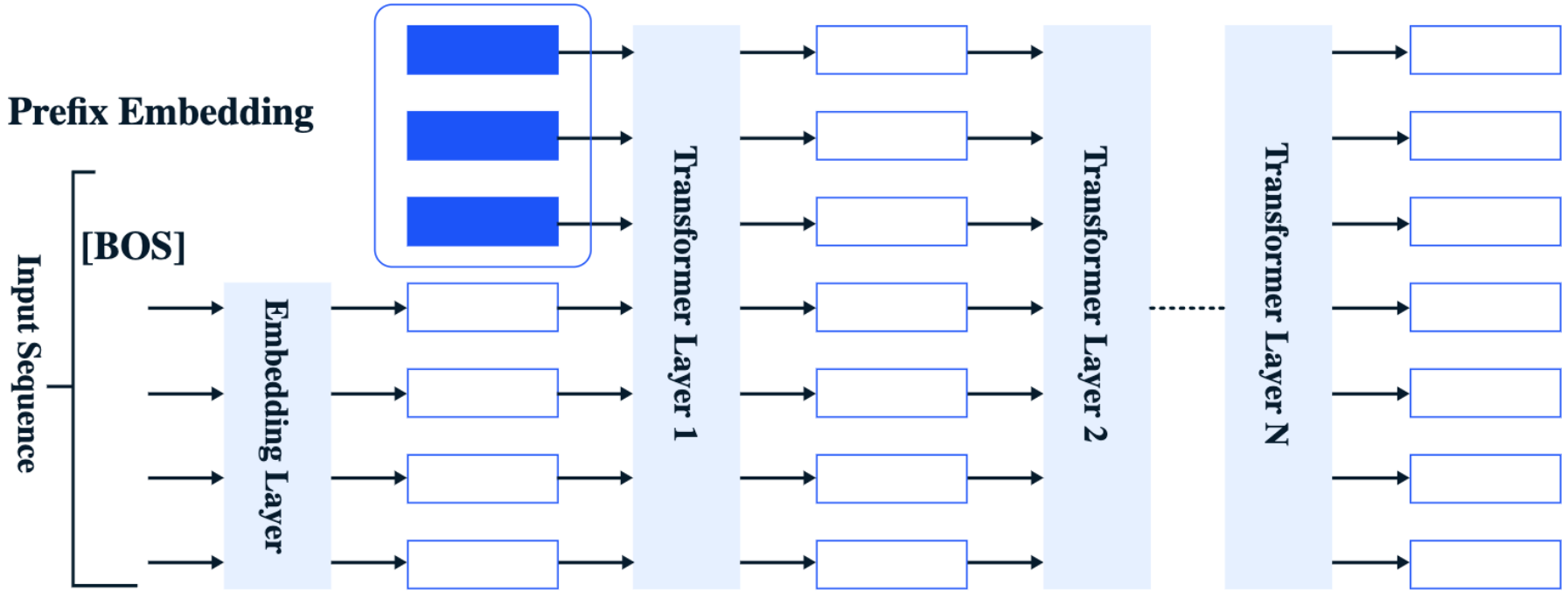
- Vanilla Fine-Tuning



Source: https://www.leewayhertz.com/parameter-efficient-fine-tuning/

Summary of Parameter-Efficient Fine-Tuning

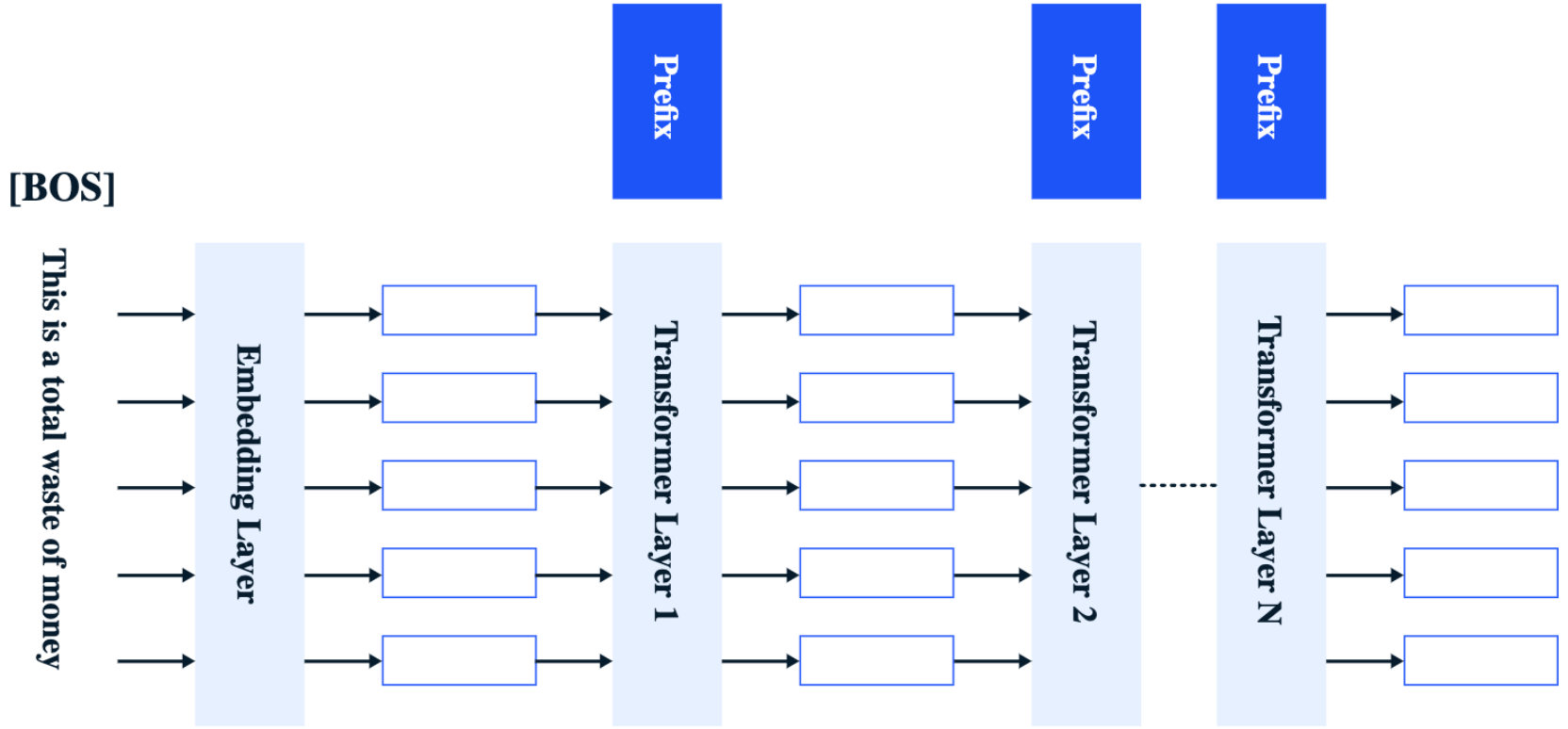
- Prompt-Tuning



Source: https://www.leewayhertz.com/parameter-efficient-fine-tuning/

Summary of Parameter-Efficient Fine-Tuning

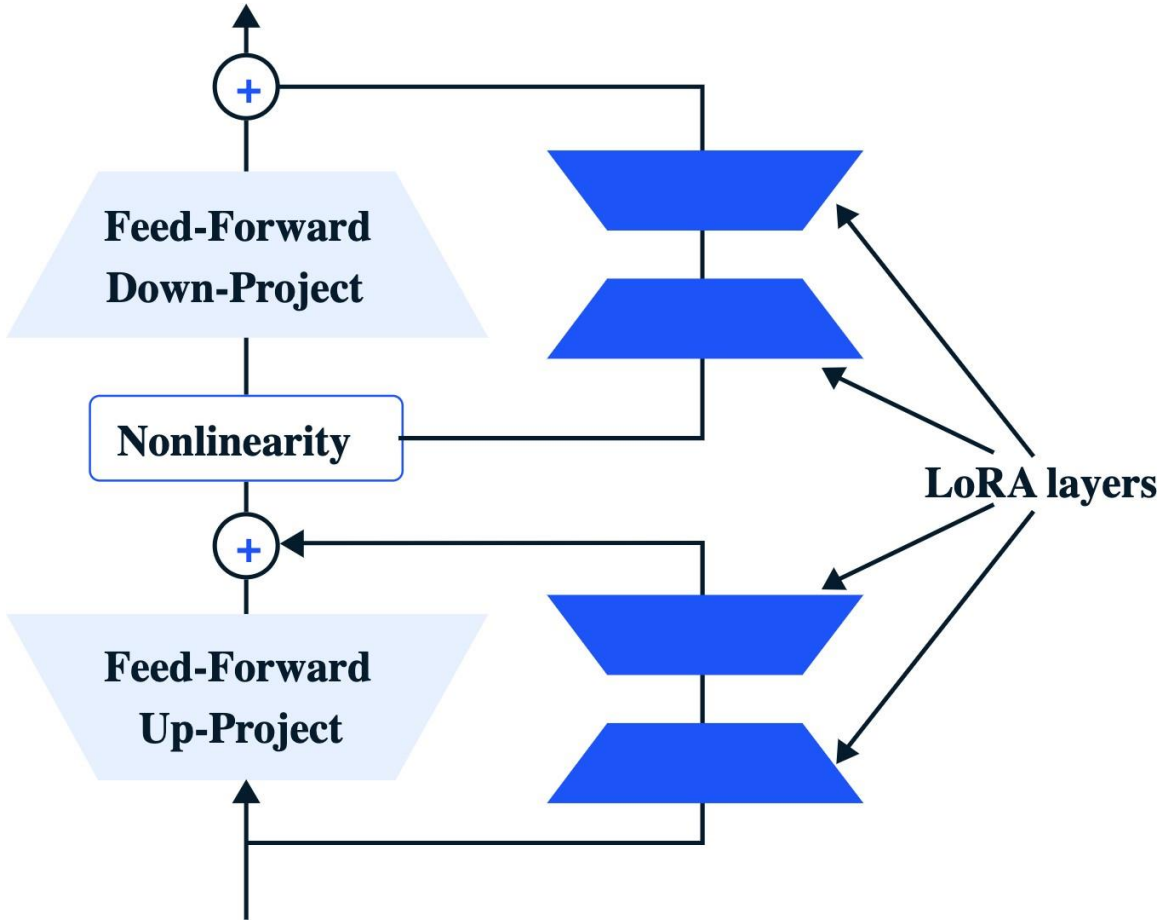
- Prefix-Tuning



Source: https://www.leewayhertz.com/parameter-efficient-fine-tuning/

Summary of Parameter-Efficient Fine-Tuning

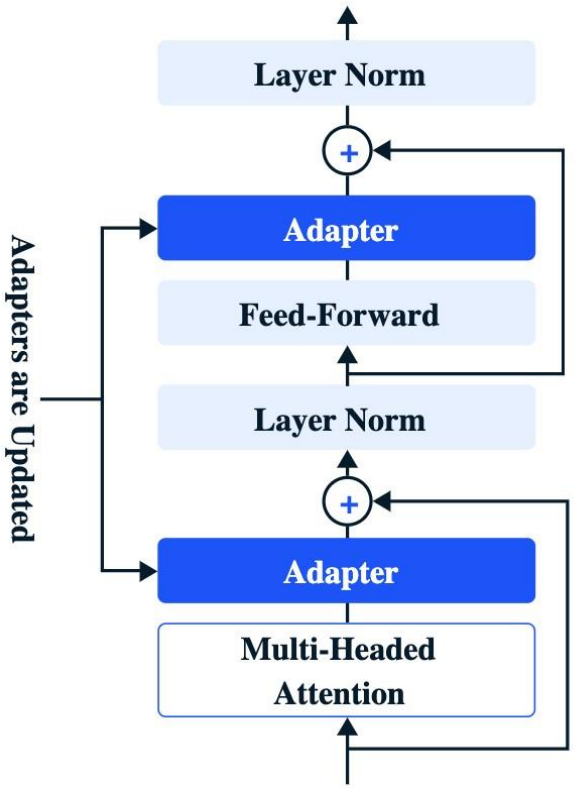
- LoRA

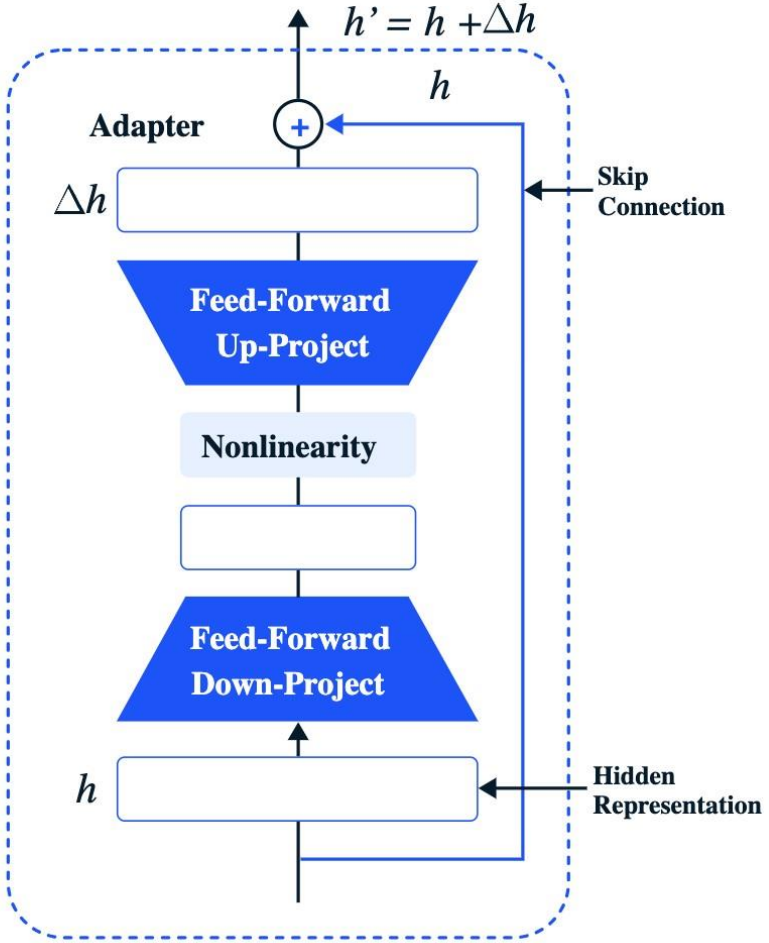


Source: https://www.leewayhertz.com/parameter-efficient-fine-tuning/

Summary of Parameter-Efficient Fine-Tuning

- Adapters

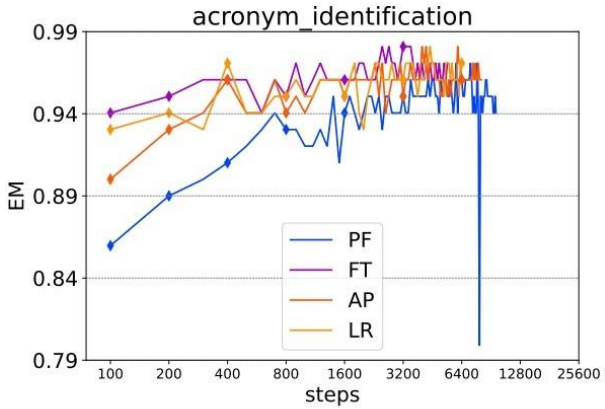


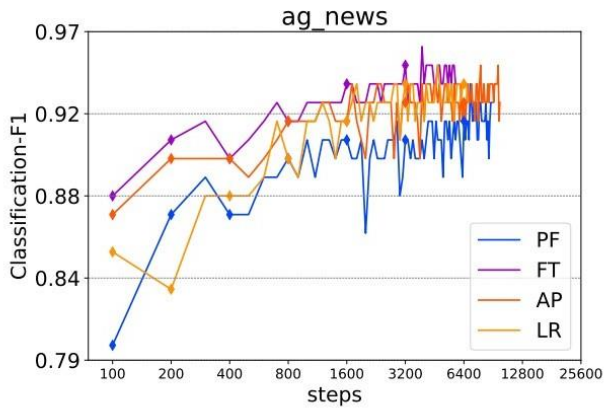


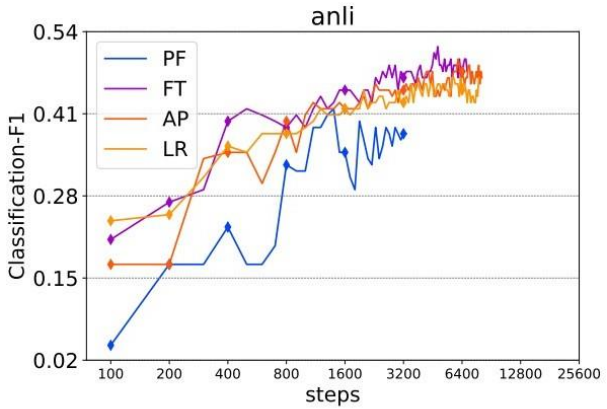
Source: https://www.leewayhertz.com/parameter-efficient-fine-tuning/

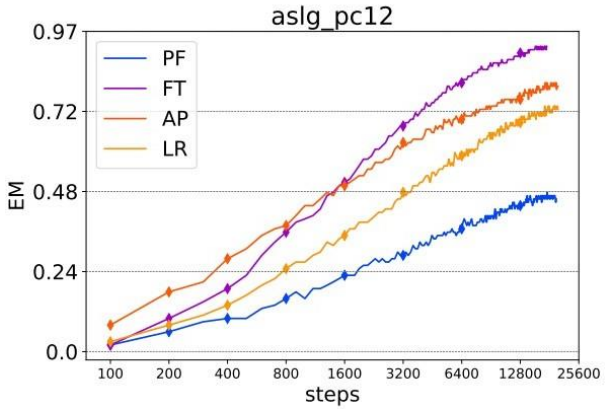
Performance Comparison on Various NLP Tasks

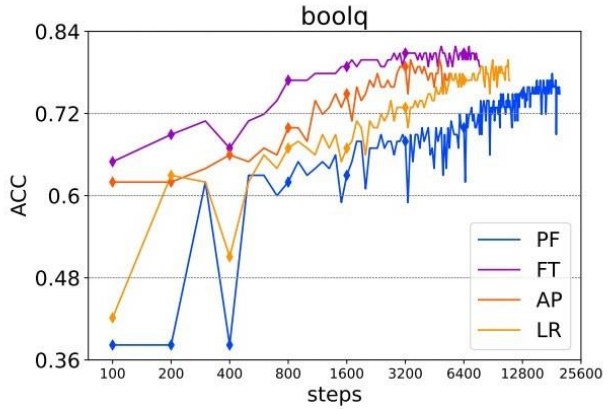
- If you have enough data and computing resources:
- Overall performance (on T5-base): Full fine-tuning > LoRA > Adapters > Prefix Tuning > Prompt Tuning

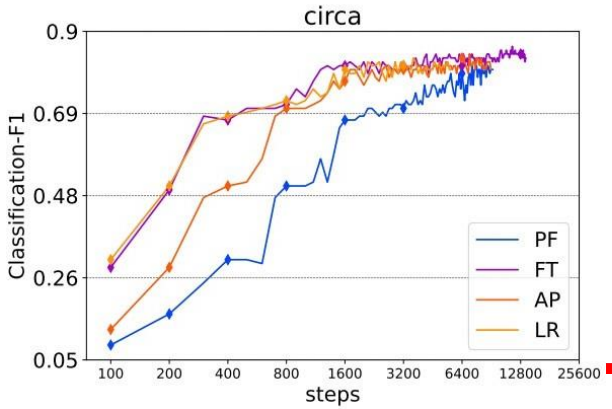












Delta Tuning: A Comprehensive Study of Parameter Efficient Methods for Pre-trained Language Models (Ding et. al, 2023)