Pretrained Language Models

SLP 3 CH 11

Training vs. Pretraining

- Previous paradigm in NLP
  - For each specific task…
  - Collect annotated data
  - Train a model using the data
- Problems
  - Insufficient annotated data
    - Hence small models
  - Separate models for different tasks
    - Intuitively, they all model languages and should have shared linguistic knowledge

Training vs. Pretraining

- Pretraining: the new paradigm
  - Train a general-purpose model
    - It should be a word-level (or subword-level) model to accommodate various NLP tasks
  - On what data?
    - Task-specific annotated data ✗
    - Unannotated text ✓
      - Much more abundant than annotated text
  - With what learning objective?
    - There is no annotation…
    - Language modeling (and its variants)!

🏆 Pretrained Language Models (dominant in NLP since 2018)

Training vs. Pretraining

- How shall we use a pretrained language model?
  - Language modeling
    - Ex: text generation
  - Text representation
    - Word, span, sentence, …
  - Strong initial models
    - …for continued training on task-specific data
    - Aka. fine-tuning

Text representation

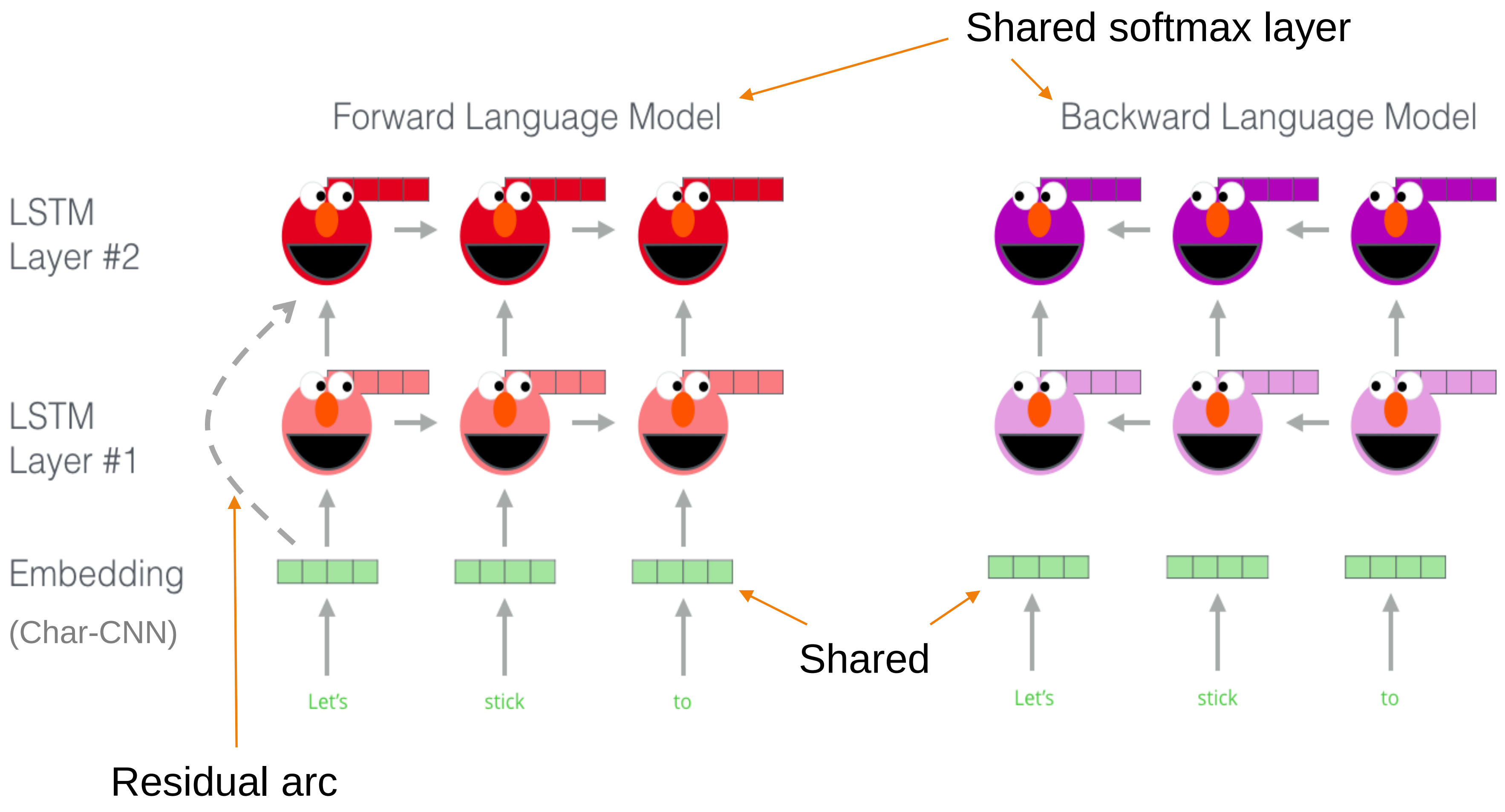
- We've learned static word representation
  - LSA, Word2vec
  - Each word has exactly one vector representation
- But a word may have meanings and syntactic behaviors in different contexts!
  - "book a flight" vs. "read a book"
  - 我骑车差点摔倒，好在我一把把把把住了
- Wouldn't it be nice to have representations of words that change over contexts?
  - This is called Contextualized Word Embeddings
  - They can be generated with PLMs

ELMo: Embeddings from Language Models

ELMo: Embeddings from Language Models

- Train a forward LSTM LM and a backward LSTM LM (bi-LSTM)
  - Use 2-layer LSTMs
  - Use character CNN to build initial word representation
  - Use a residual connection
  - Share the parameters for token representation and softmax word prediction layer in the two LSTMs

ELMo: Embeddings from Language Models



https://jalammar.github.io/illustrated-bert/

ELMo: Embeddings from Language Models

- ELMo learns to combine BiLSTM representations in different layers.

$$ELMo_k = \sum_{j=0}^{L} s_j \cdot h_{k,j}^{LM}$$

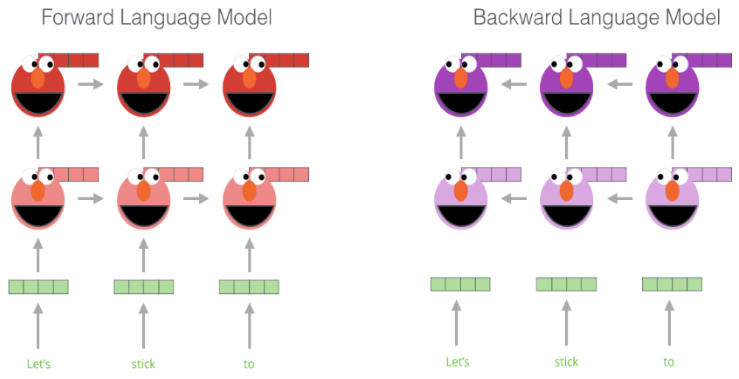
  - s_j is a learnable weight for layer j
- The two BiLSTM layers have differentiated uses
  - Lower layer is better for lower-level syntax, etc.
    - Part-of-speech tagging, syntactic dependencies, NER
  - Higher layer is better for higher-level semantics
    - Sentiment, Semantic role labeling, question answering, SNLI

Use ELMo with a task

- Run ELMo on input sentences to get word representations
- Then let (whatever) end-task model use them
- When training the end-task model:
  - Freeze weights of ELMo
  - Or: also train weights of ELMo together with the end-task model, which typically leads to better performance
- This is the pretrain+finetune paradigm!
  - Pretrain: train the word embedding model on a general task with lots of data (e.g., LM)
  - Finetune: for a specific task, connect the word embedding model to the task model and continue its training together with the task model

BERT: Bidirectional Encoder Representations from Transformers

From ELMo to BERT

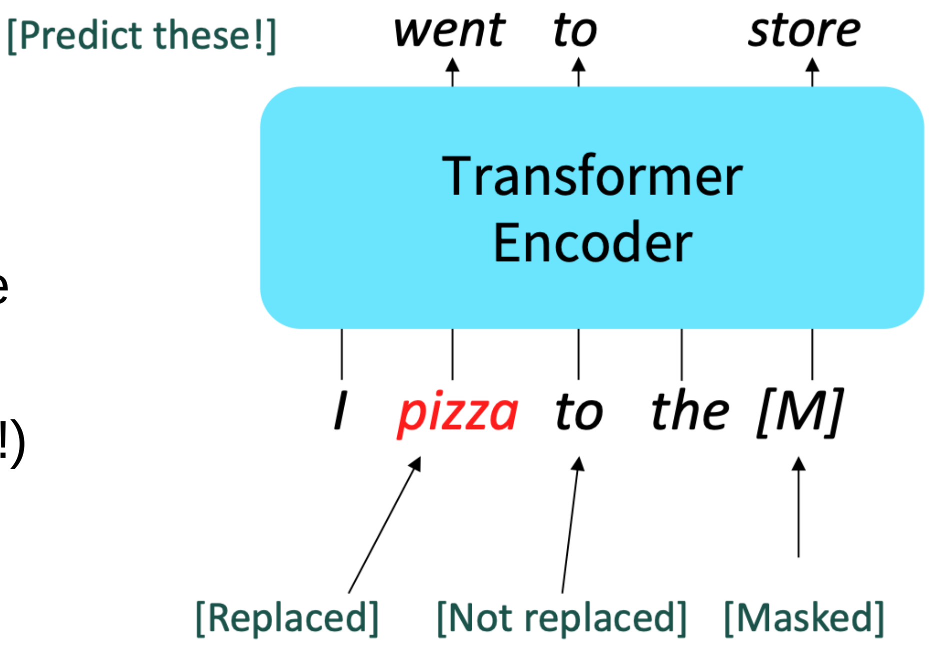


- ELMo: separate representations for the left context and right context
- Why not a unified representation for the whole context?
  - Because LM is unidirectional
- Solution: masked language model (MLM)
  - Mask out *k*% of the input words, and then predict the masked words

$$\begin{array}{c} \text{I went to the } \overset{\overset{\text{store}}{\uparrow}}{\text{[MASK]}} \text{ to buy a } \overset{\overset{\text{gallon}}{\uparrow}}{\text{[MASK]}} \text{ of milk} \end{array}$$

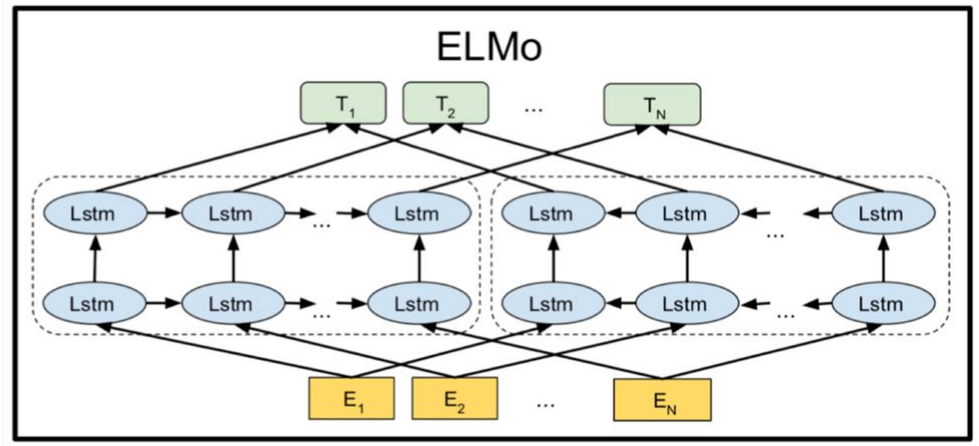
Masked Language Models in BERT

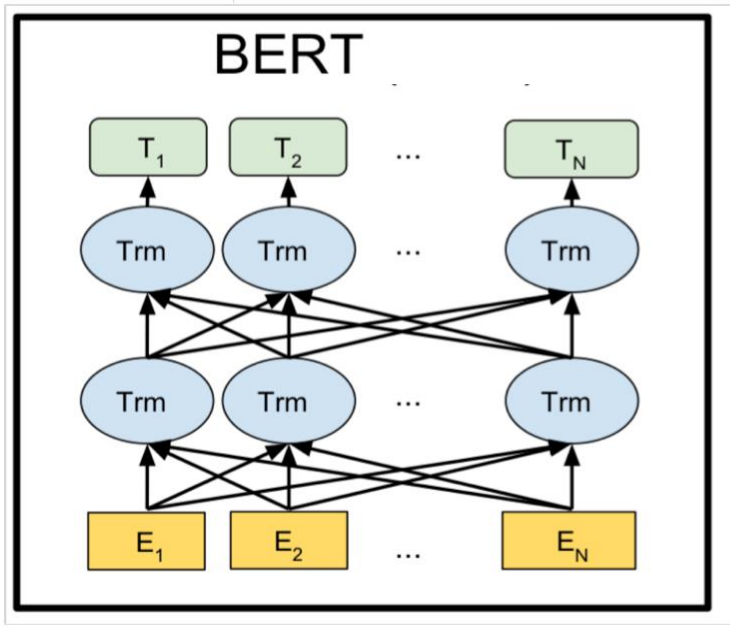
- Predict a random 15% of (sub)word tokens.
  - Replace input word with [MASK] 80% of the time
  - Replace input word with a random token 10% of the time
  - Leave input word unchanged 10% of the time (still predict it!)
- Why the last part?
  - No masking during fine-tuning and prediction
  - The model must learn to build strong representations of non-masked words



From ELMo to BERT

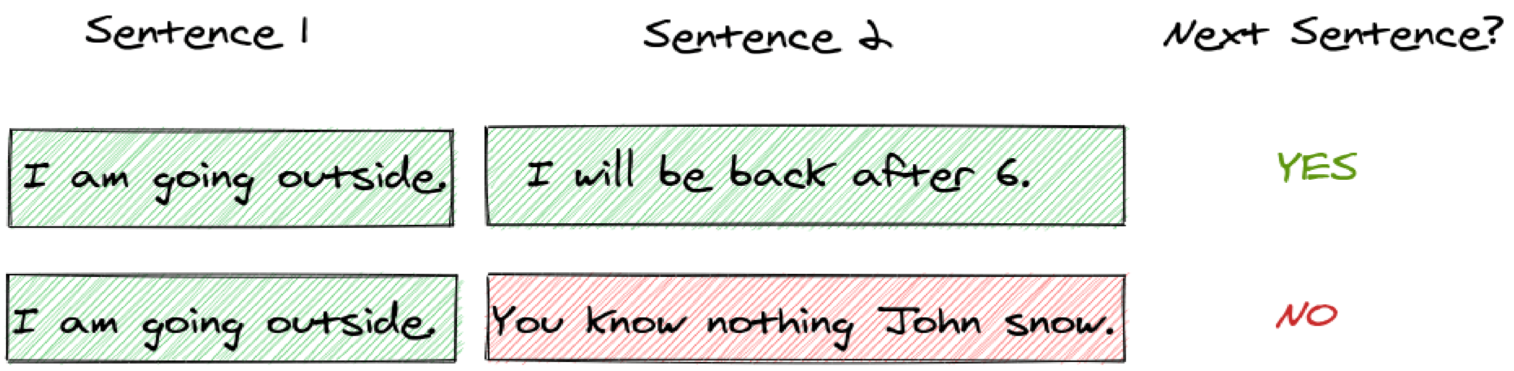
- Replace LSTM with Transformer
  - Long-range dependency & parallelizability
- Use subword tokenization (WordPiece)
  - Similar to BPE





BERT complication: next sentence prediction (NSP)

- To learn relationships between sentences, predict whether Sentence B is a sentence that proceeds Sentence A



[CLS] I am going outside [SEP] I will be back after 6 [SEP]

↓

FFN+Softmax → IsNext: 99%
NotNext: 1%

Later work argued that NSP is not necessary.

Details about BERT

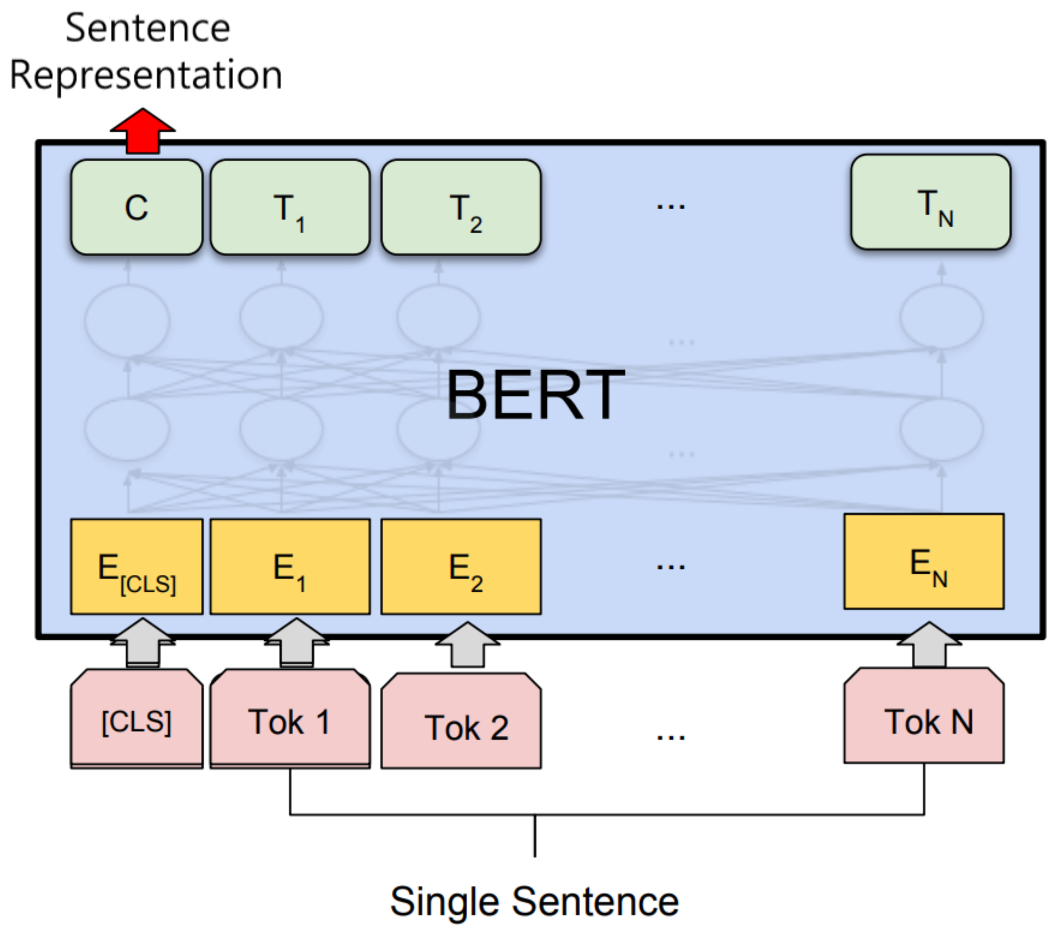
- Two released models:
  - BERT-base: 12 layers, 768-dim hidden states, 12 attention heads, 110 million params.
  - BERT-large: 24 layers, 1024-dim hidden states, 16 attention heads, 340 million params.
- Trained on:
  - BooksCorpus (800 million words)
  - English Wikipedia (2,500 million words)
- Pretraining is expensive and impractical on a single GPU.
  - BERT was pretrained with 64 TPU chips for a total of 4 days. (TPUs are special tensor operation acceleration hardware)

BERT Output

- Produce contextual representations from:
  - Last layer
  - Second-to-last layer
  - Mixture of (top) layers
- If a word is split to multiple subwords:
  - Mean pooling
  - Embedding of the first subword

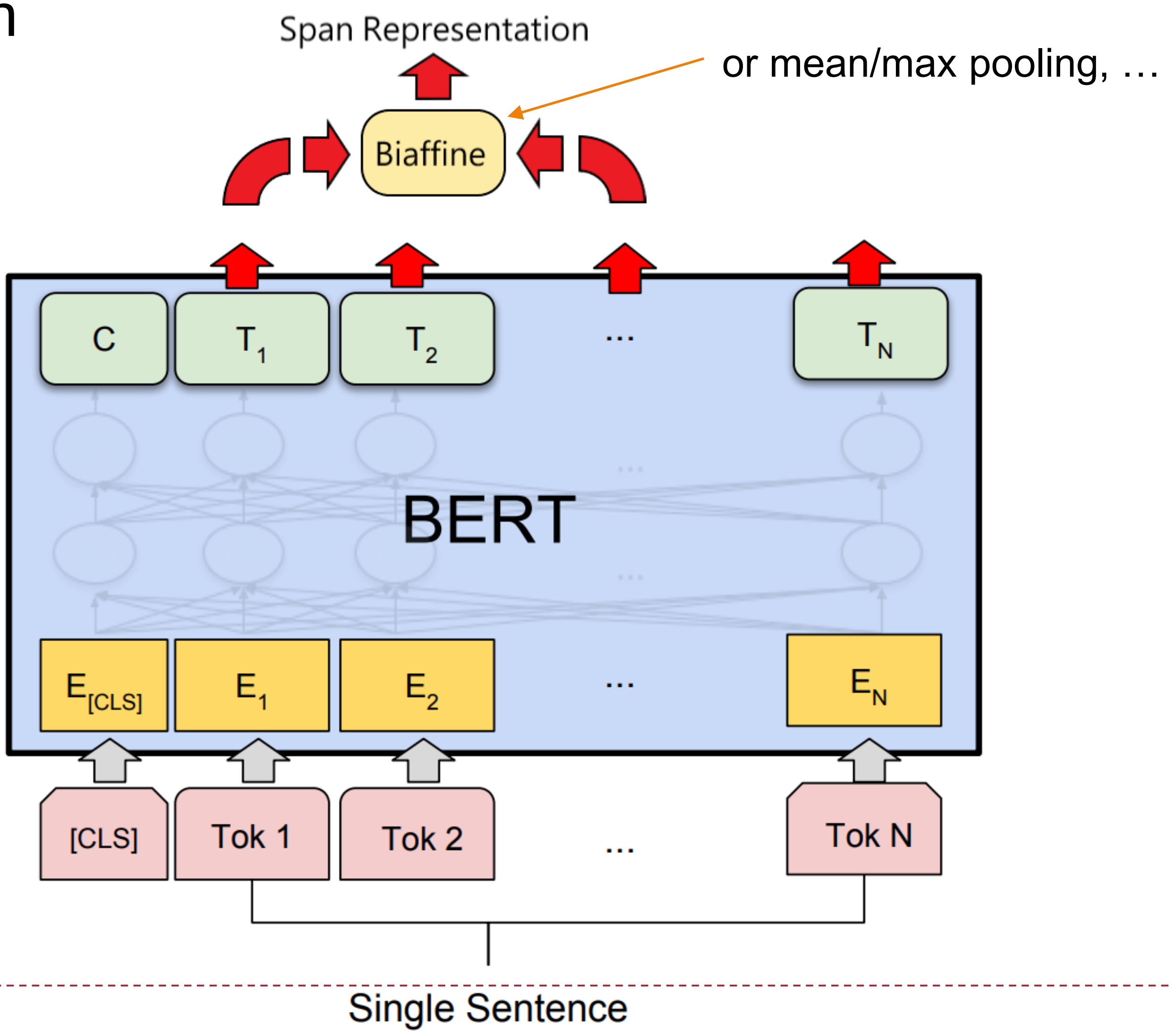
Beyond word representations

- Sentence



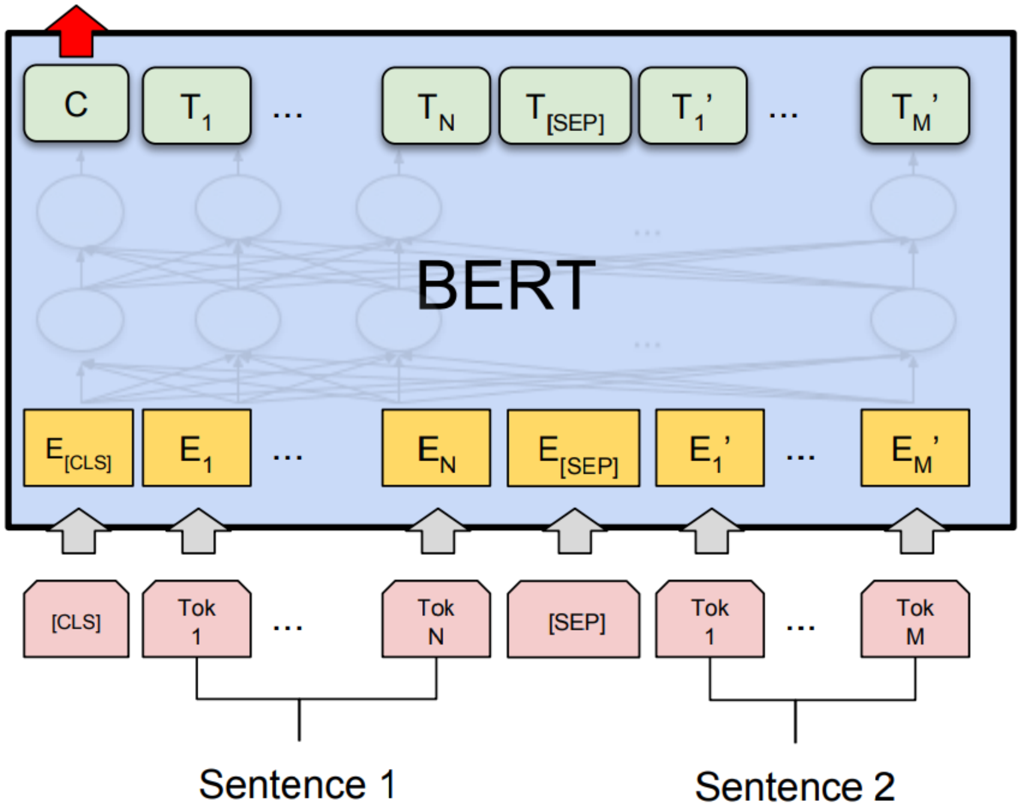
Beyond word representations

- Text Span



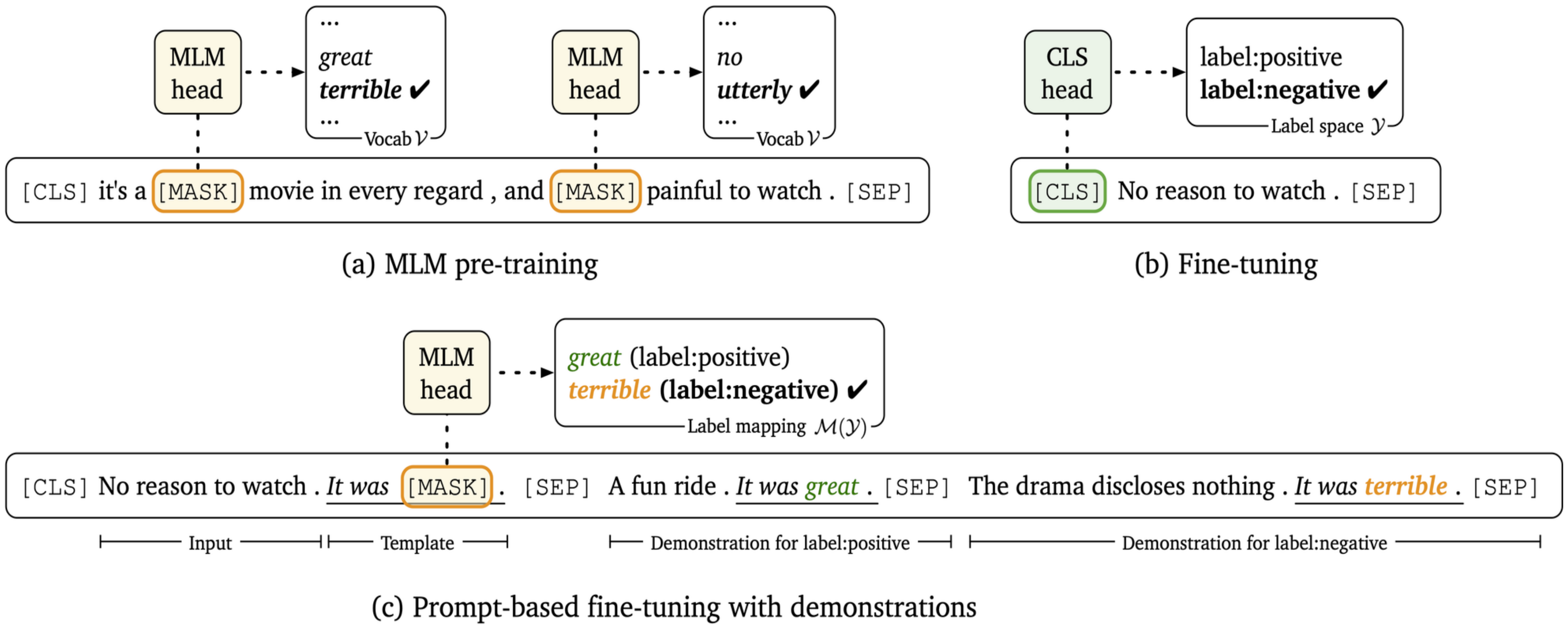
Beyond word representations

- Sentence pair

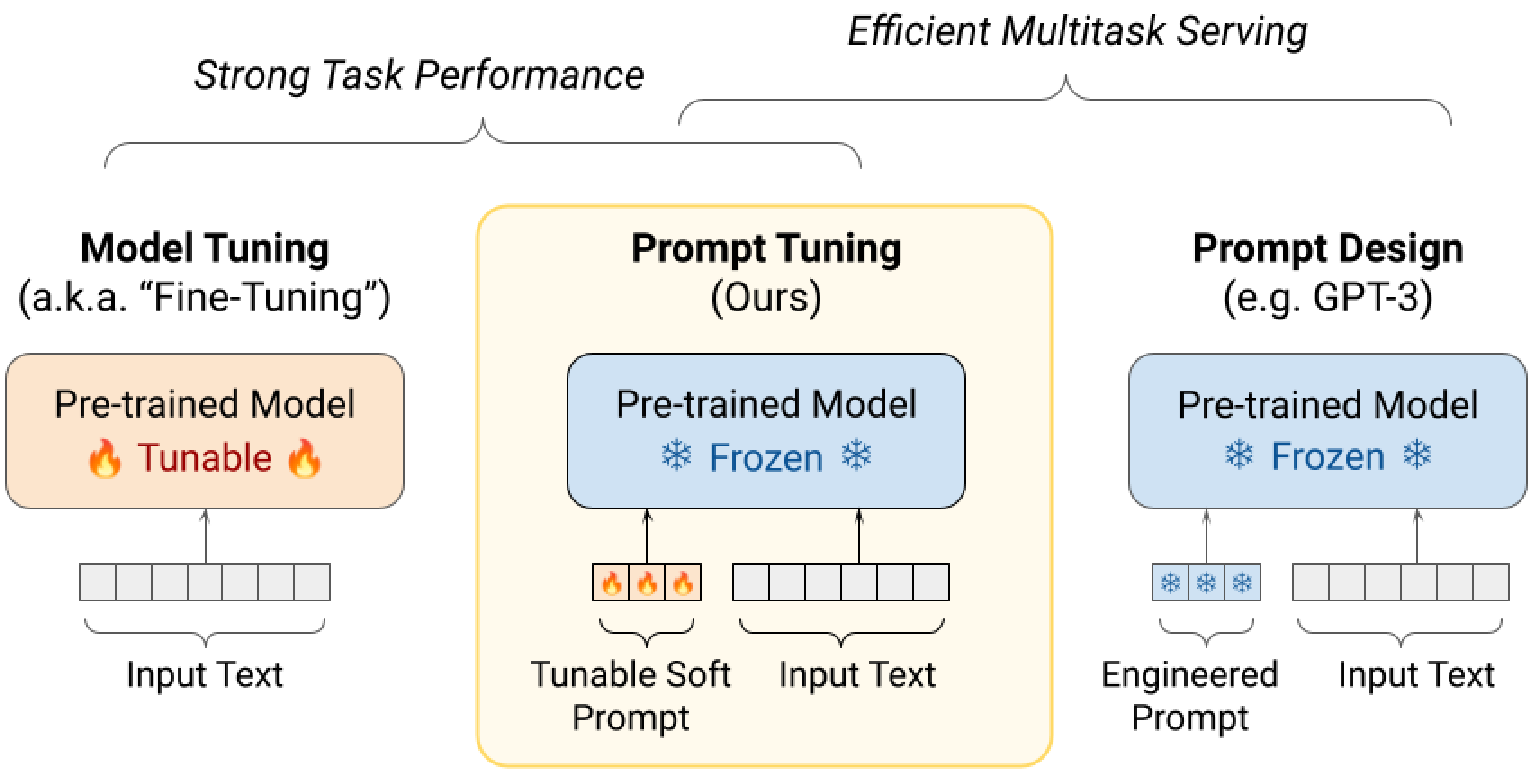


Utilizing BERT in downstream tasks

- Finetuning: just like in ELMo
- Prompting: insert a piece of text (prompt) in the input to formulate a task as MLM.


MLM head
...
great
terrible ✔
...
Vocab $\mathcal{V}$
MLM head
...
no
utterly ✔
...
Vocab $\mathcal{V}$
[CLS] it's a [MASK] movie in every regard , and [MASK] painful to watch . [SEP]
(a) MLM pre-training
CLS head
label:positive
label:negative ✔
Label space $\mathcal{Y}$
[CLS] No reason to watch . [SEP]
(b) Fine-tuning
MLM head
great (label:positive)
terrible (label:negative) ✔
Label mapping $\mathcal{M}(\mathcal{Y})$
[CLS] No reason to watch . It was [MASK] . [SEP] A fun ride . It was great . [SEP] The drama discloses nothing . It was terrible . [SEP]
Input
Template
Demonstration for label:positive
Demonstration for label:negative
(c) Prompt-based fine-tuning with demonstrations

Utilizing BERT in downstream tasks



https://ai.googleblog.com/2022/02/guiding-frozen-language-models-with.html

Extensions of BERT

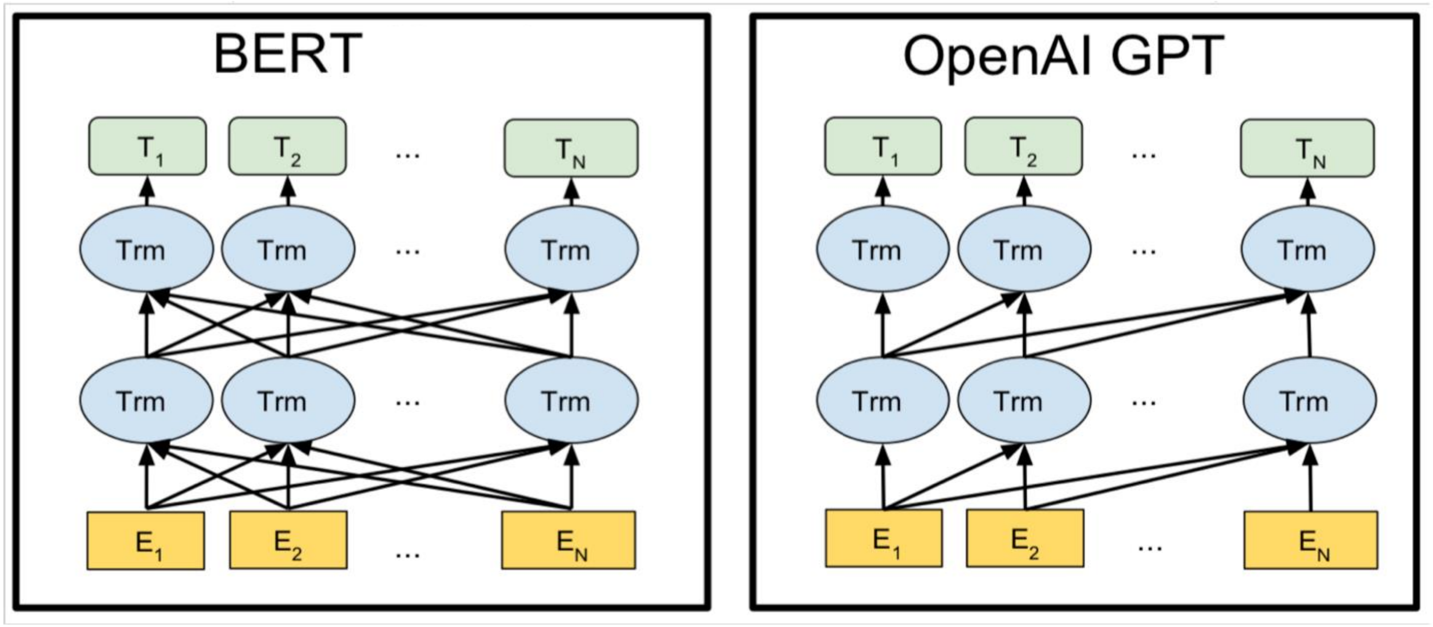
- A lot of variants
  - RoBERTa: mainly just train BERT for longer and remove next sentence prediction!
  - mBERT: trained on multilingual data
  - SpanBERT: masking contiguous spans of words makes a harder, more useful pretraining task
  - ERNIE: introduce entity/relation knowledge into pretraining
  - ALBERT: apply parameter saving strategy to build deeper and parameter-efficient models
  - Many more…

GPT: Generative Pre-trained Transformer

GPT

- A unidirectional language model
  - Transformer with attention mask
- Pre-training loss
  - $-\sum_i \log P(w_i | w_{i-k}, \ldots, w_{i-1}; \theta)$

$-\infty$	$-\infty$	$-\infty$	$-\infty$
	$-\infty$	$-\infty$	$-\infty$
		$-\infty$	$-\infty$
			$-\infty$



Utilizing GPT

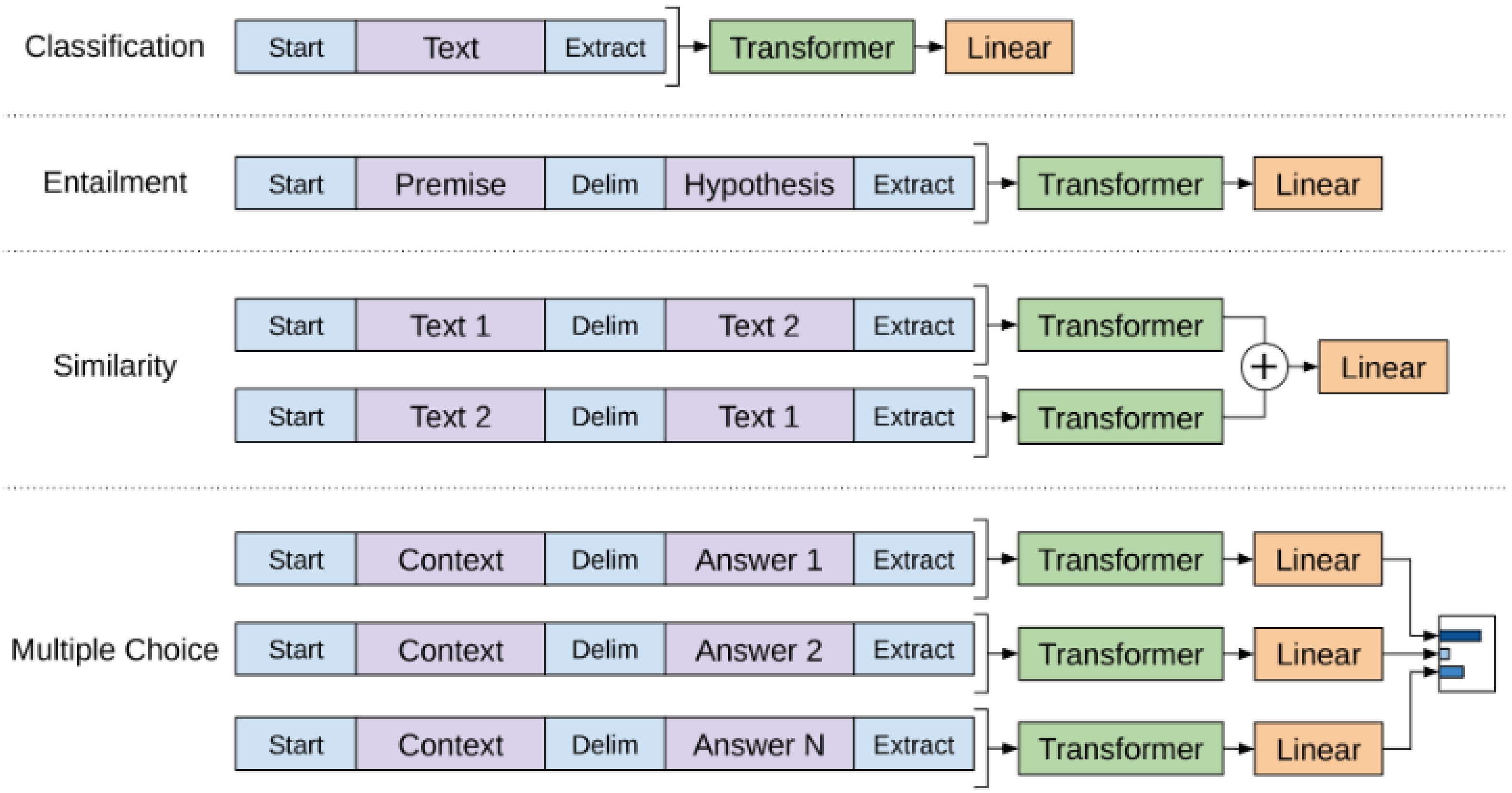
- Generation
  - Human prompt
    - In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.
  - Model completion
    - The scientist named the population, after their distinctive horn, Ovid's Unicorn. These four-horned, silver-white unicorns were previously unknown to science.

      Now, after almost two centuries, the mystery of what sparked this odd phenomenon is finally solved.

      Dr. Jorge Pérez, an evolutionary biologist from the University of La Paz, and several companions, were exploring the Andes Mountains when they found a small valley, with no other animals or humans. Pérez noticed that the valley had what appeared to be a natural fountain, surrounded by two peaks of rock and silver snow.

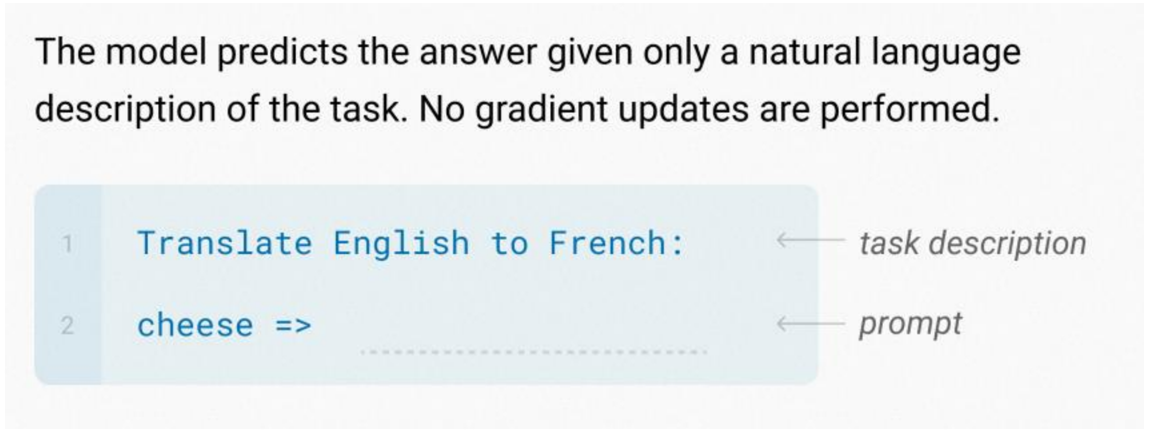
Utilizing GPT

- Finetuning
  - Predicting from output of the last token



Utilizing GPT

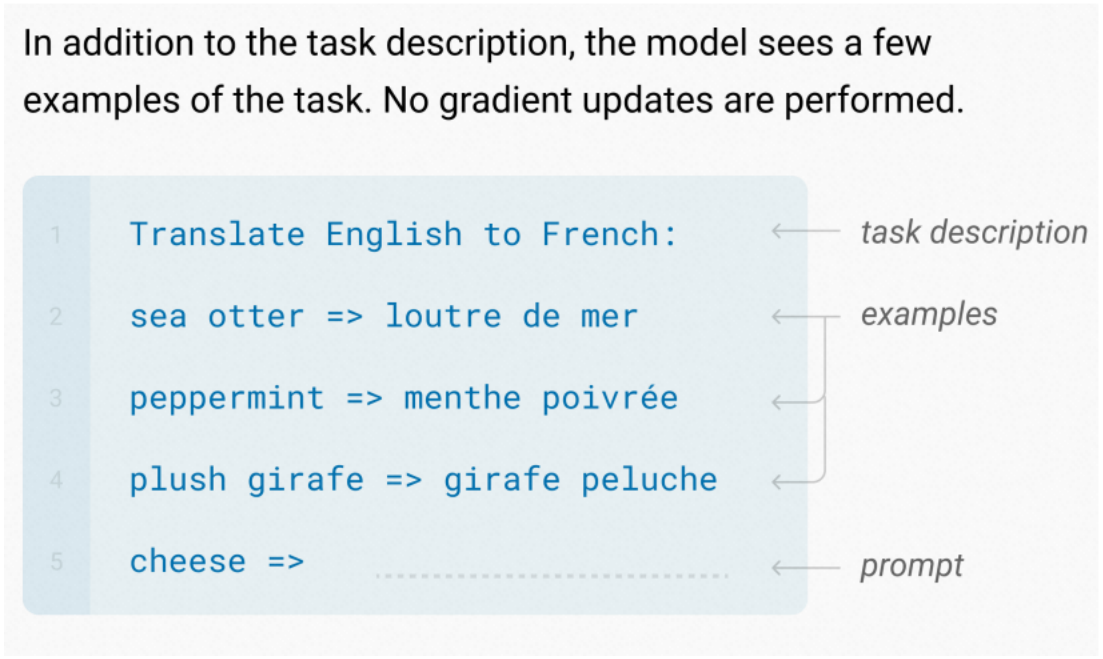
- Prompting


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

```
Translate English to French:
cheese =>
```

Utilizing GPT

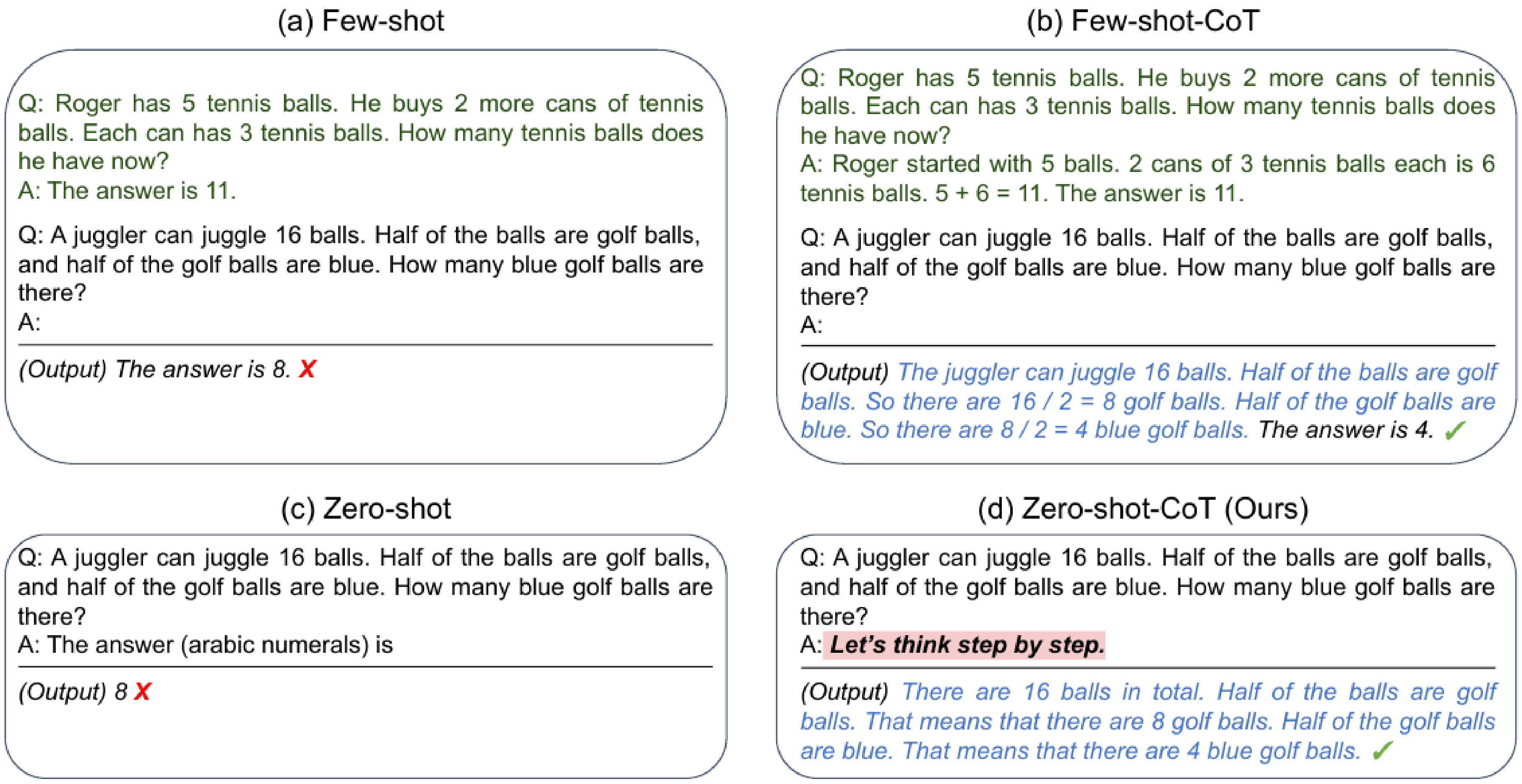
- Prompting (in-context learning)


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

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

Utilizing GPT

- Chain-of-thought prompting



(a) Few-shot

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: The answer is 11.

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

(Output) The answer is 8. ***X***

(b) Few-shot-CoT

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?
A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. 5 + 6 = 11. The answer is 11.

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

(Output) The juggler can juggle 16 balls. Half of the balls are golf balls. So there are 16 / 2 = 8 golf balls. Half of the golf balls are blue. So there are 8 / 2 = 4 blue golf balls. The answer is 4. ✓

(c) Zero-shot

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?
A: The answer (arabic numerals) is

(Output) 8 ***X***

(d) Zero-shot-CoT (Ours)

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?
A: ***Let's think step by step.***

(Output) There are 16 balls in total. Half of the balls are golf balls. That means that there are 8 golf balls. Half of the golf balls are blue. That means that there are 4 blue golf balls. ✓

Kojima et al. NIPS2022

GPT History

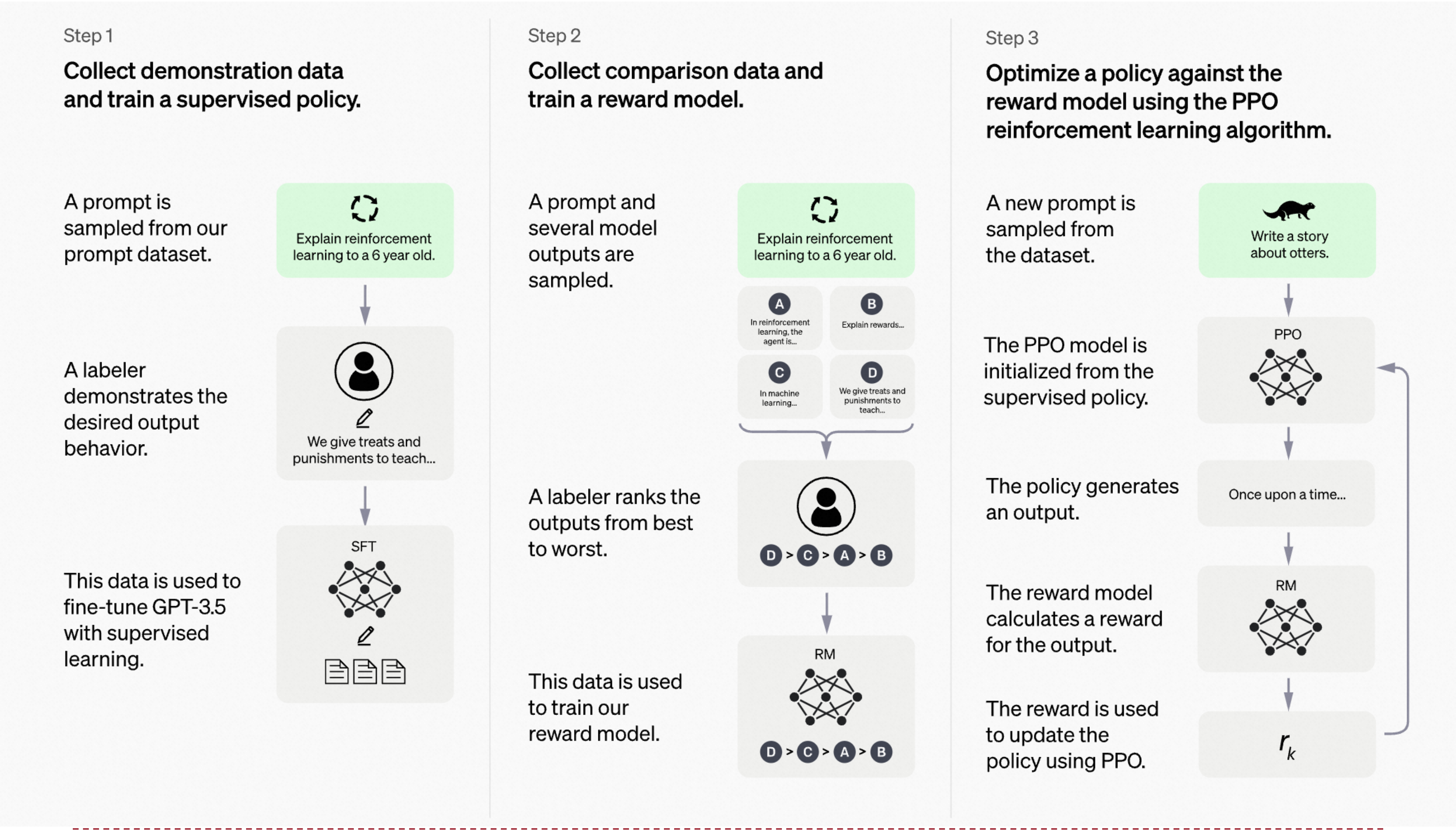
- GPT-1 (2018)
  - 117 million parameters
  - 5GB training data
- GPT-2 (2019)
  - 1.5 billion parameters
  - 40GB training data
- GPT-3 (2020)
  - 175 billion parameters!
    - 96 layers, 96 attention heads, 12888 word embedding size, 2048 context window size
  - 45TB Training data (300 billion tokens)
  - Cost ≈ 355 V100 GPU-years and $4.6M

Dataset	# tokens	Proportion
Common Crawl	410 billion	60%
WebText2	19 billion	22%
Books1	12 billion	8%
Books2	55 billion	8%
Wikipedia	3 billion	3%

InstructGPT

- Based on GPT-3
- Additionally trained with humans in the loop to better follow user intensions
  - Reinforcement Learning with Human Feedback (RLHF)

InstructGPT



Step 1

Collect demonstration data and train a supervised policy.

A prompt is sampled from our prompt dataset.

A labeler demonstrates the desired output behavior.

This data is used to fine-tune GPT-3.5 with supervised learning.

Step 2

Collect comparison data and train a reward model.

A prompt and several model outputs are sampled.

A labeler ranks the outputs from best to worst.

This data is used to train our reward model.

Step 3

Optimize a policy against the reward model using the PPO reinforcement learning algorithm.

A new prompt is sampled from the dataset.

The PPO model is initialized from the supervised policy.

The policy generates an output.

The reward model calculates a reward for the output.

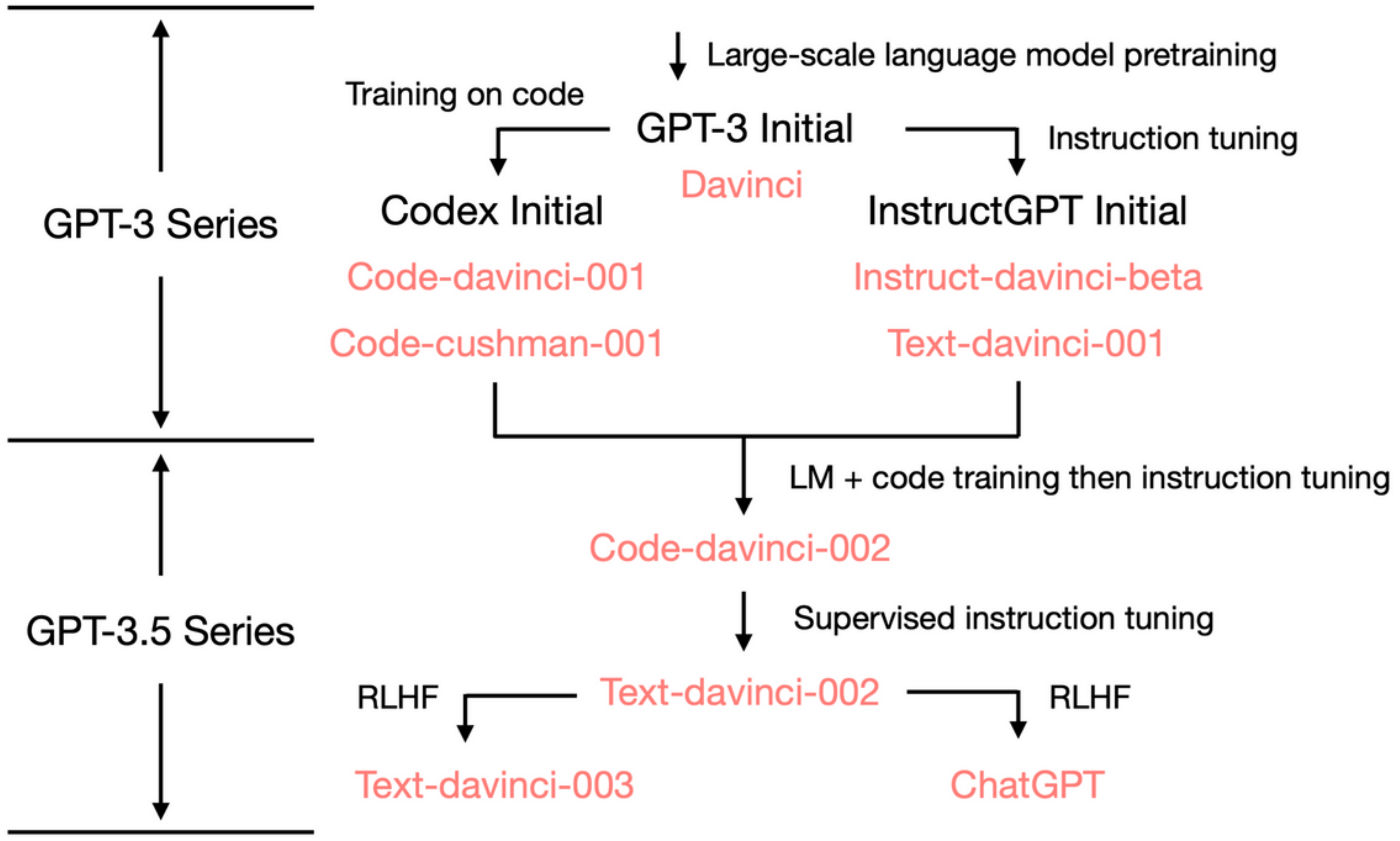
The reward is used to update the policy using PPO.

Explain reinforcement learning to a 6 year old.
We give treats and punishments to teach...
SFT
Explain reinforcement learning to a 6 year old.
A
In reinforcement learning, the agent is...
B
Explain rewards...
C
In machine learning...
D
We give treats and punishments to teach...
D > C > A > B
RM
D > C > A > B
Write a story about otters.
PPO
Once upon a time...
RM
r_k

ChatGPT

- ChatGPT
  - Based on InstructGPT
  - Possibly also trained on programming code

GPT-3 Family


Large-scale language model pretraining
Training on code
GPT-3 Initial
Davinci
Instruction tuning
GPT-3 Series
Codex Initial
InstructGPT Initial
Code-davinci-001
Instruct-davinci-beta
Code-cushman-001
Text-davinci-001
LM + code training then instruction tuning
Code-davinci-002
GPT-3.5 Series
Supervised instruction tuning
RLHF
Text-davinci-002
RLHF
Text-davinci-003
ChatGPT

Encoder-Decoder PLMs

Transformer PLMs

- Encoder-only
  - BERT, …
- Decoder-only (auto-regressive)
  - GPT, …
- Encoder-Decoder
  - BART
    - Pre-trained by corrupting text with an arbitrary noising function, and learning a model to reconstruct the original text.
  - T5: Text-to-Text Transfer Transformer
    - Pre-trained on a mixture of unsupervised and supervised tasks converted into a text-to-text format

Summary

Pretrained Language Models

- ELMo
  - BiLSTM + LM
- BERT
  - Transformer + MLM
- GPT
  - Transformer + LM
- Utilizing PLMs
  - Finetuning
  - Prompting