AI VIETNAM
All-in-One Course

NLP Course

Large Language Models
Prompting Techniques

Nguyen Quoc Thai

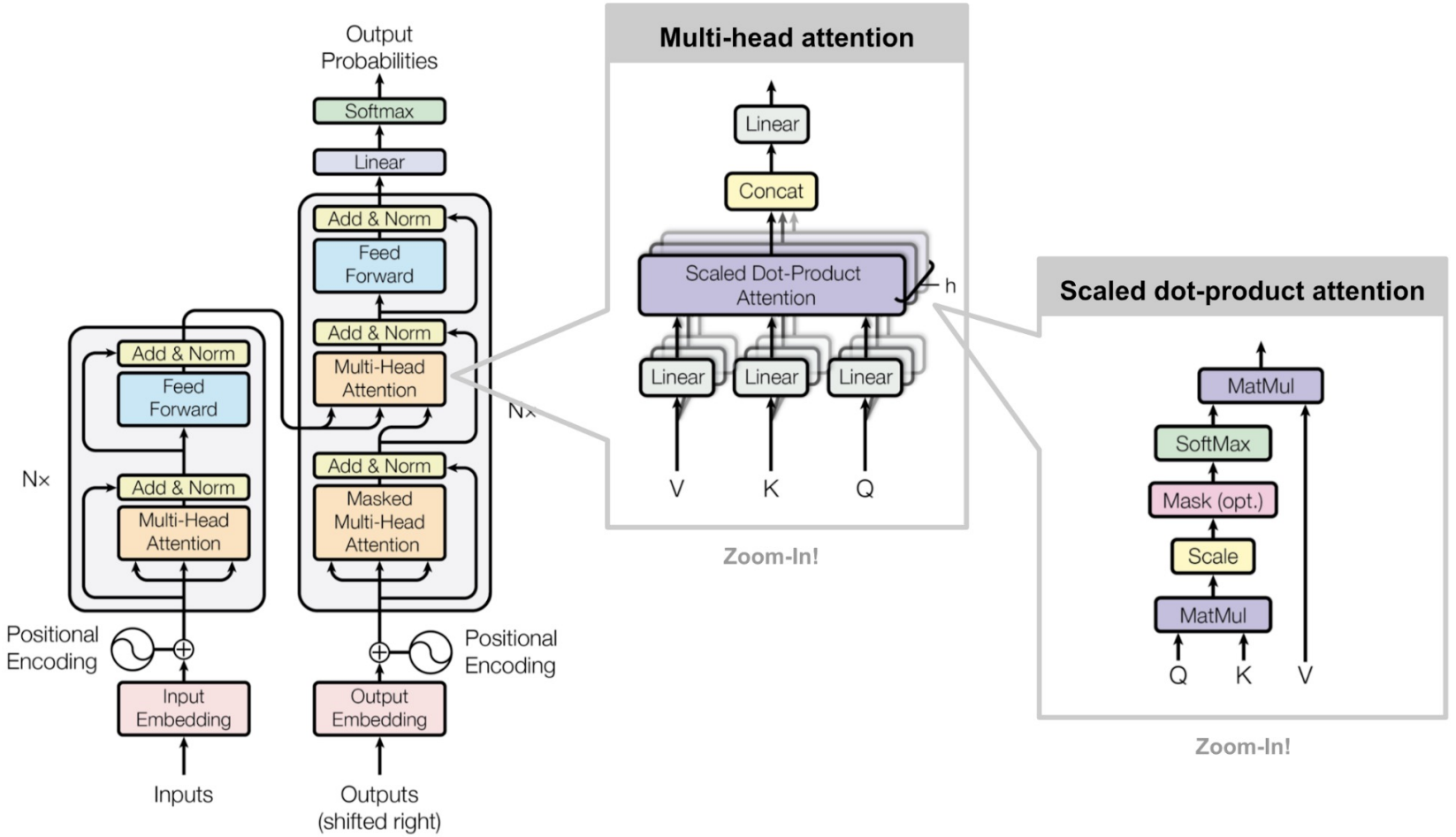
Year 2023

CONTENT

1 – Large Language Models

Review

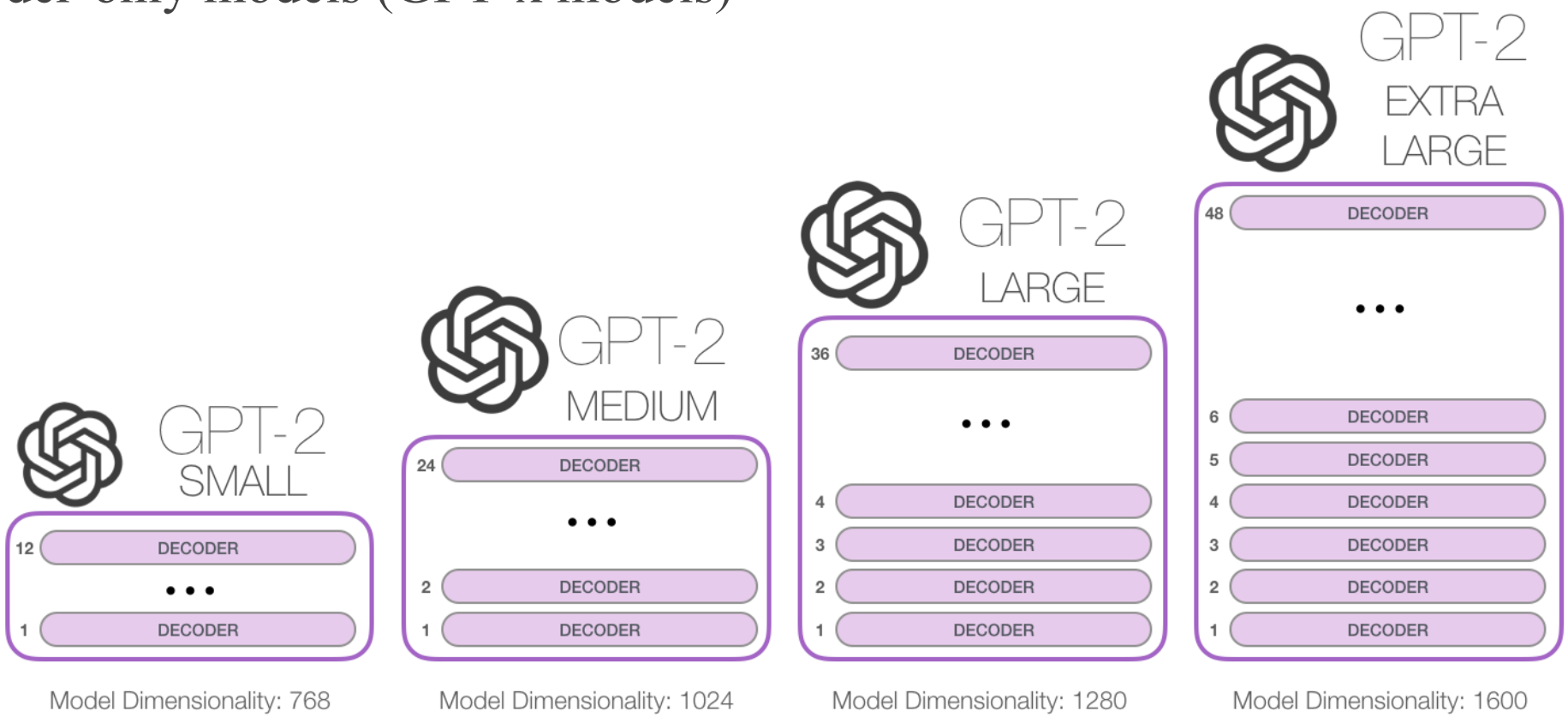
- Transformer



1 – Large Language Models

Review

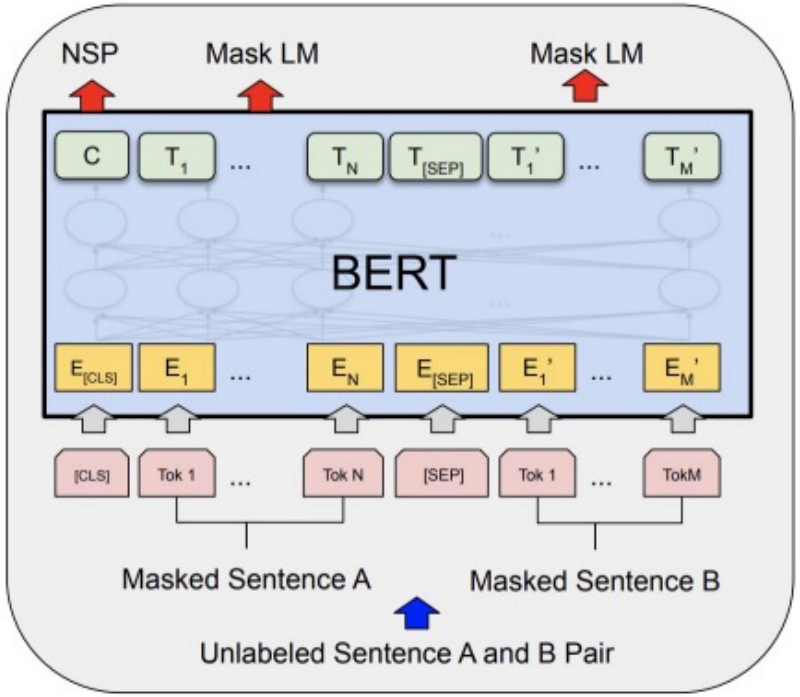
- Decoder-only models (GPT-x models)

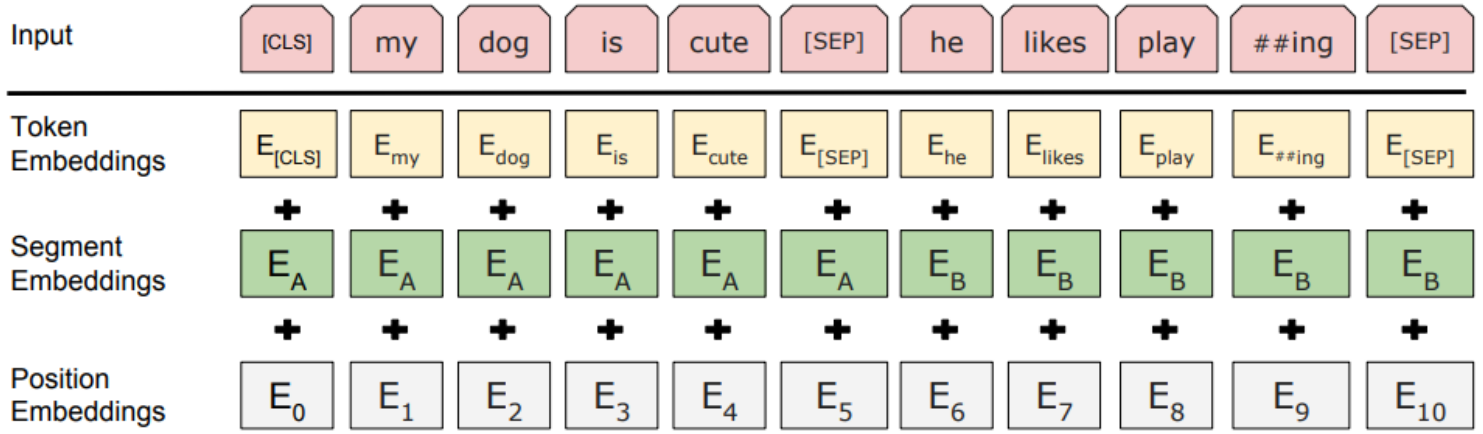


1 – Large Language Models

Review

- Encoder-only models (BERT, RoBERTa, ELECTRA)



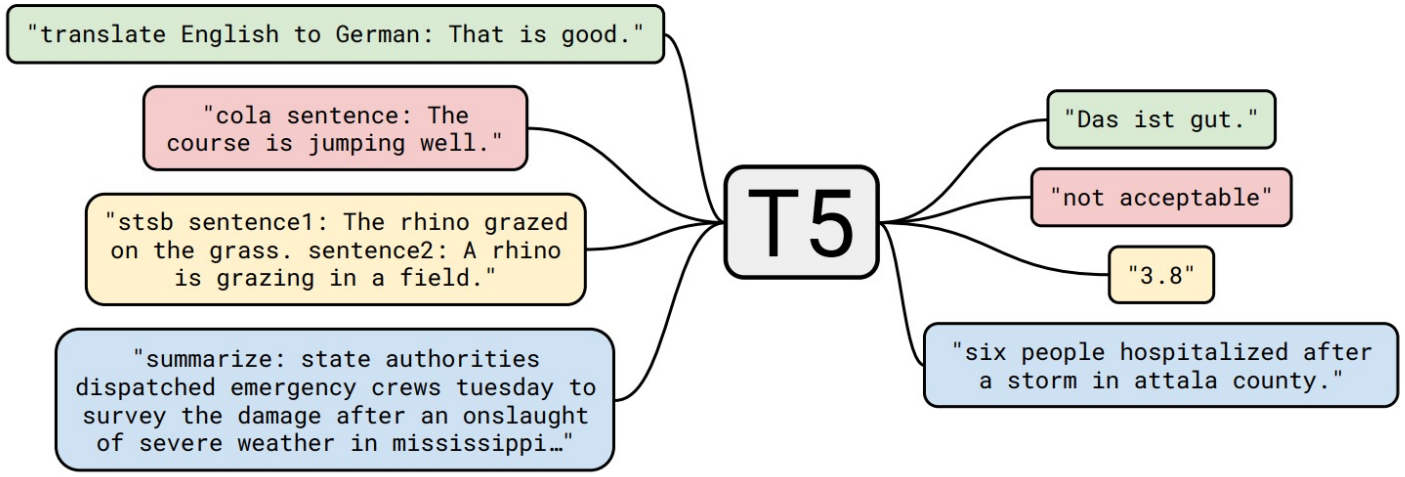


Input BERT

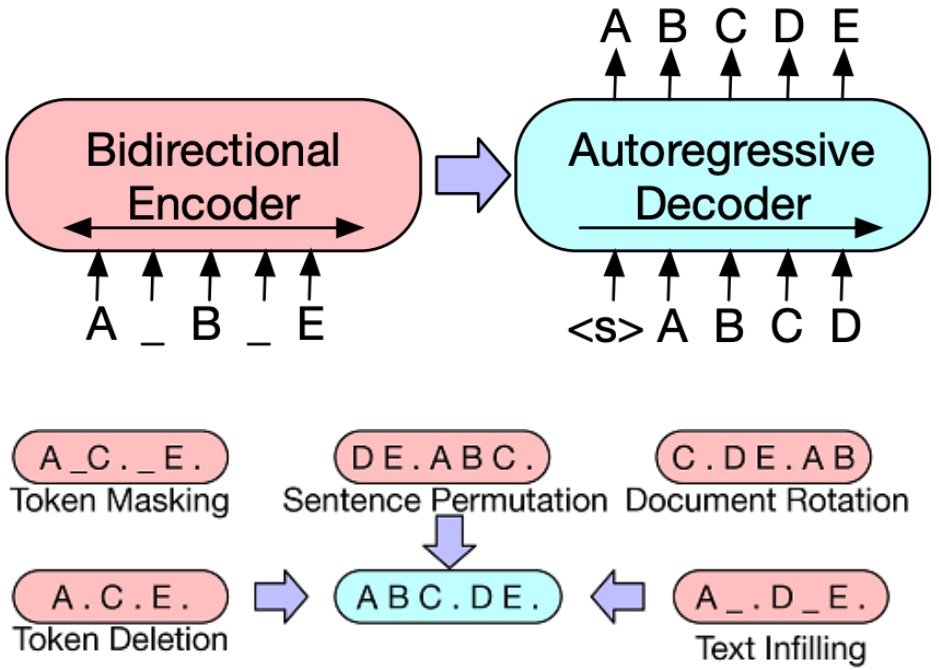
1 – Large Language Models

Review

- Encoder-Decoder models (T5, BART)



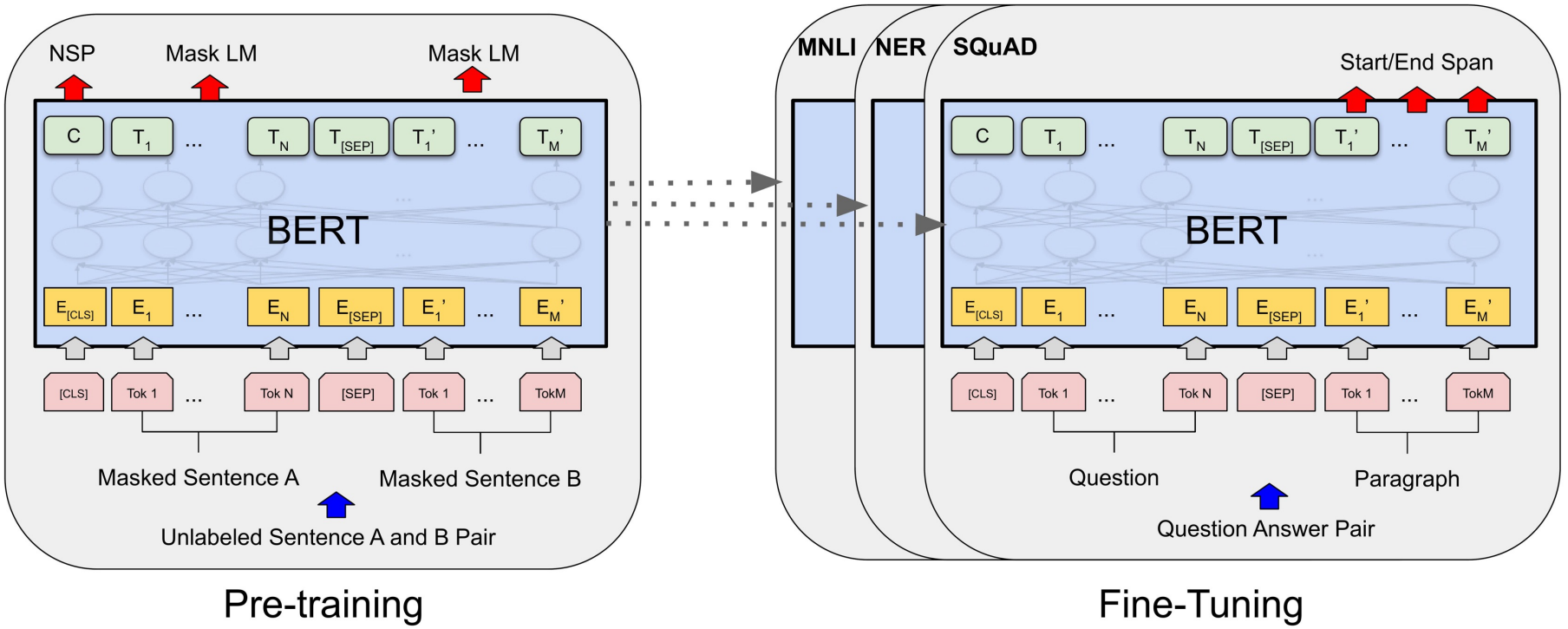
BART model



1 – Large Language Models

Why need Large Language Models?

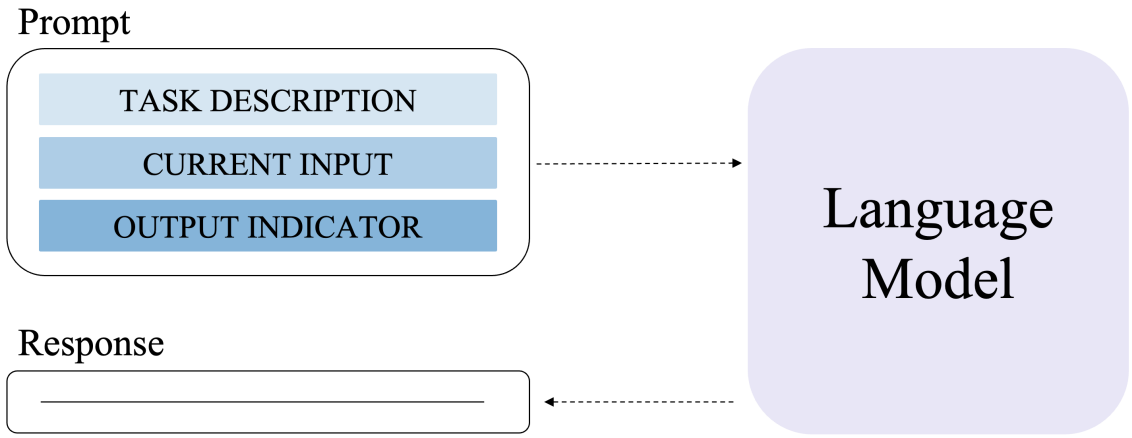
- Pre-training: trained on huge amounts of unlabeled text using "self-supervised" training objective
- Adaptation: how to use a pre-trained model for downstream task?



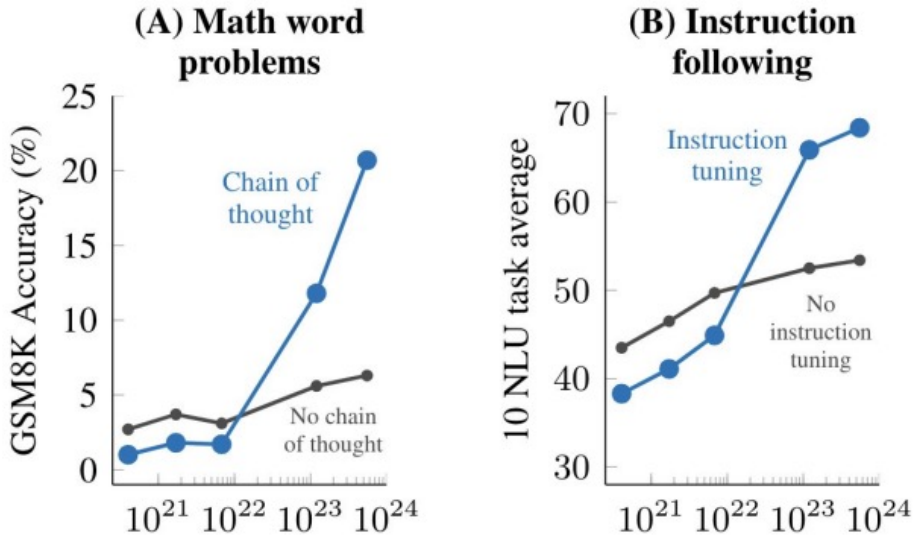
1 – Large Language Models

Why need Large Language Models?

- The promise: one single model to solve many NLP tasks



- Emergent properties in LLMs



1 – Large Language Models

Prompt

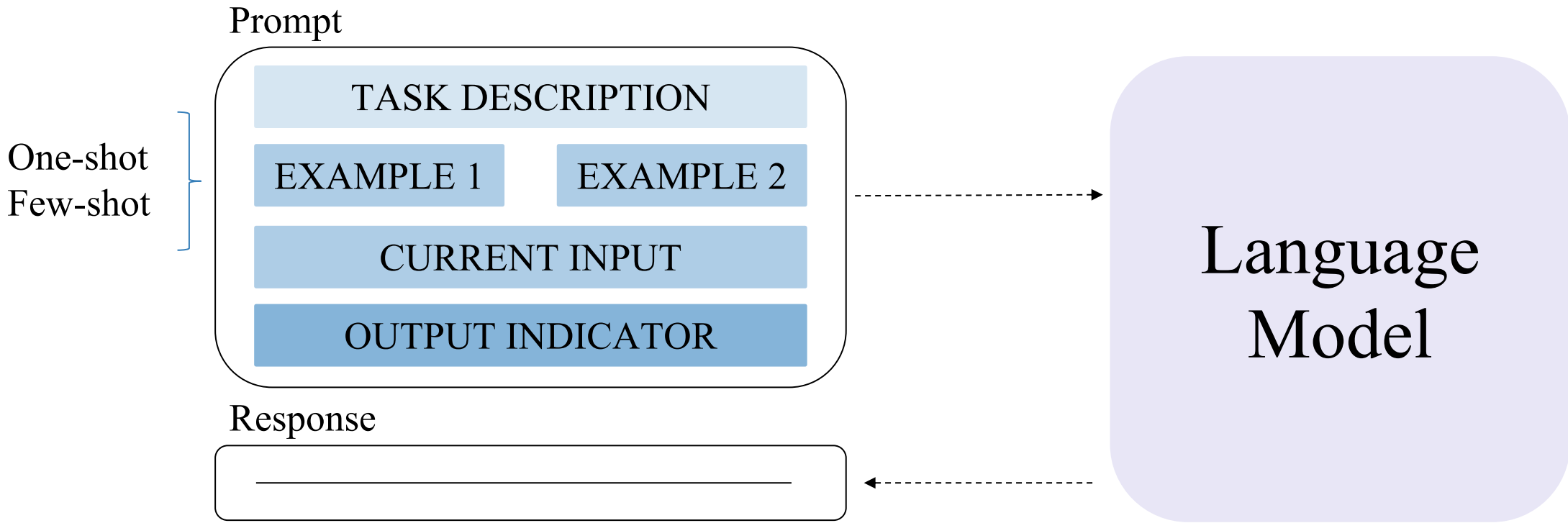
- Prompts involve instructions and context passed to a language model to achieve a desired task
- Prompt engineering is the practice of developing and optimizing prompts to efficiently use language models (LMs) for a variety of applications

```
What is prompt engineering?

Prompt engineering is a process of creating a set of prompts,
or questions, that are used to guide the user toward a desired
outcome. It is an effective tool for designers to create user
experiences that are easy to use and intuitive. This method is
often used in interactive design and software development, as
it allows users to easily understand how to interact with a
system or product..
```

1 – Large Language Models

Elements of a Prompt



1 – Large Language Models

Prompt Examples

Three setting for In-Content-Learning

Few-shot

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

```
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
```

Zero-shot

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

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

One-shot

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

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

1 – Large Language Models

Parameter

- Medium-sized models: BERT/RoBERTa models (100M or 300M), T5 models (220M, 700M, 3B, 11B)
- "Very" large LMs: models of 100+ billion parameters

  GPT3 (175B), BLOOM (176B), PaLM (540B), GLaM (1200B)…
- Larger model sizes => Larger compute, more expensive during inference

1 – Large Language Models

Dataset

- Data scale: usually in the order of trillions of tokens

  GPT3 (0.5 trillion tokens), LLaMA (1.4 trillion tokens), …

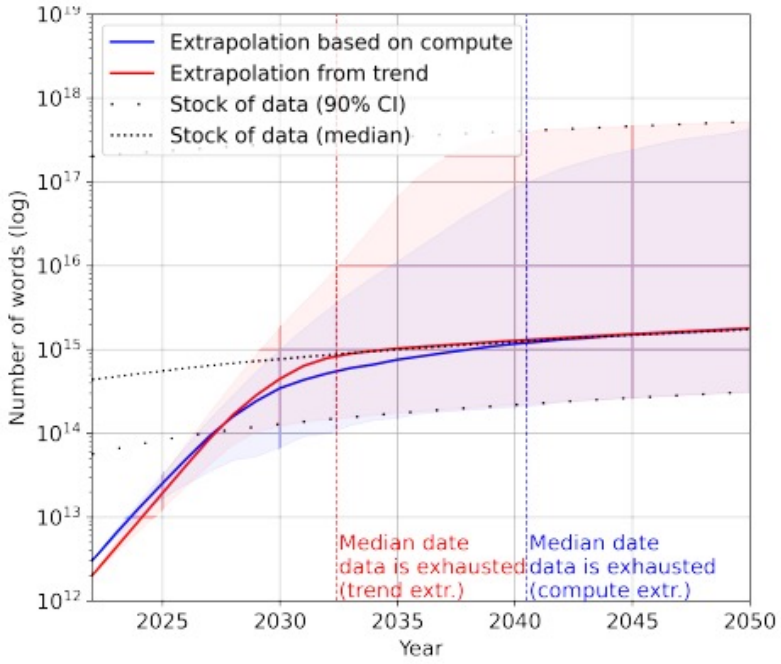
- Training data: Low-quality data

Corpora	Size	Source	Latest Update Time
BookCorpus [122]	5GB	Books	Dec-2015
Gutenberg [123]	-	Books	Dec-2021
C4 [73]	800GB	CommonCrawl	Apr-2019
CC-Stories-R [124]	31GB	CommonCrawl	Sep-2019
CC-NEWS [27]	78GB	CommonCrawl	Feb-2019
REALNEWs [125]	120GB	CommonCrawl	Apr-2019
OpenWebText [126]	38GB	Reddit links	Mar-2023
Pushift.io [127]	2TB	Reddit links	Mar-2023
Wikipedia [128]	21GB	Wikipedia	Mar-2023
BigQuery [129]	-	Codes	Mar-2023
the Pile [130]	800GB	Other	Dec-2020
ROOTS [131]	1.6TB	Other	Jun-2022

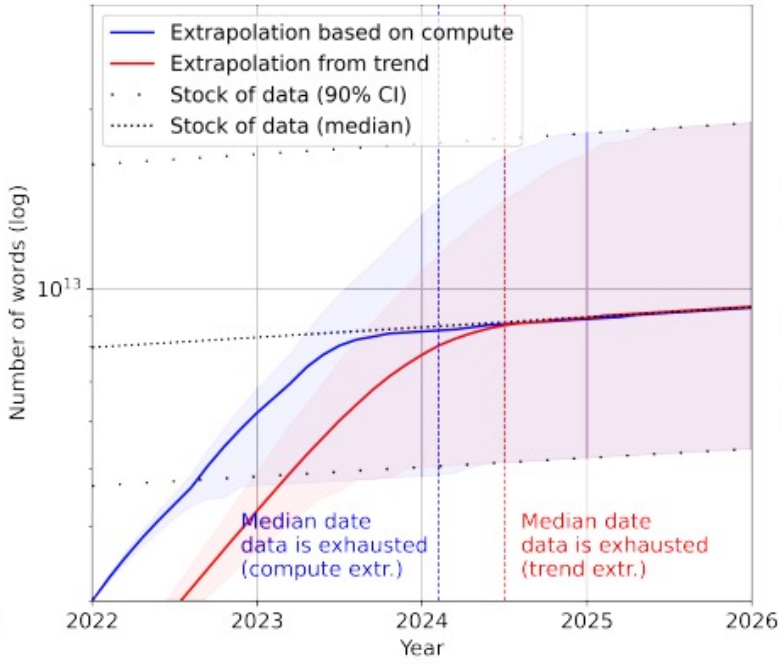
1 – Large Language Models

Dataset

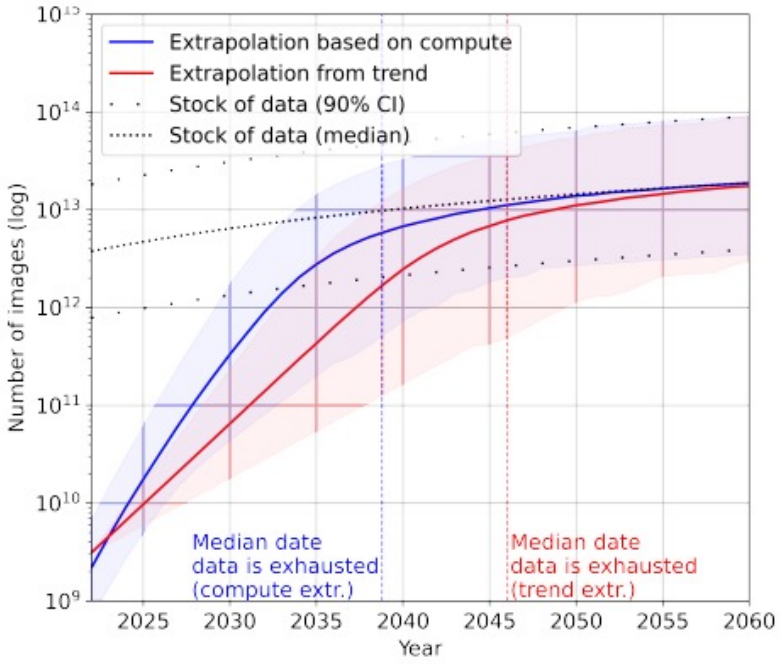
- Data bottleneck for training



(a) Projections for low-quality language data



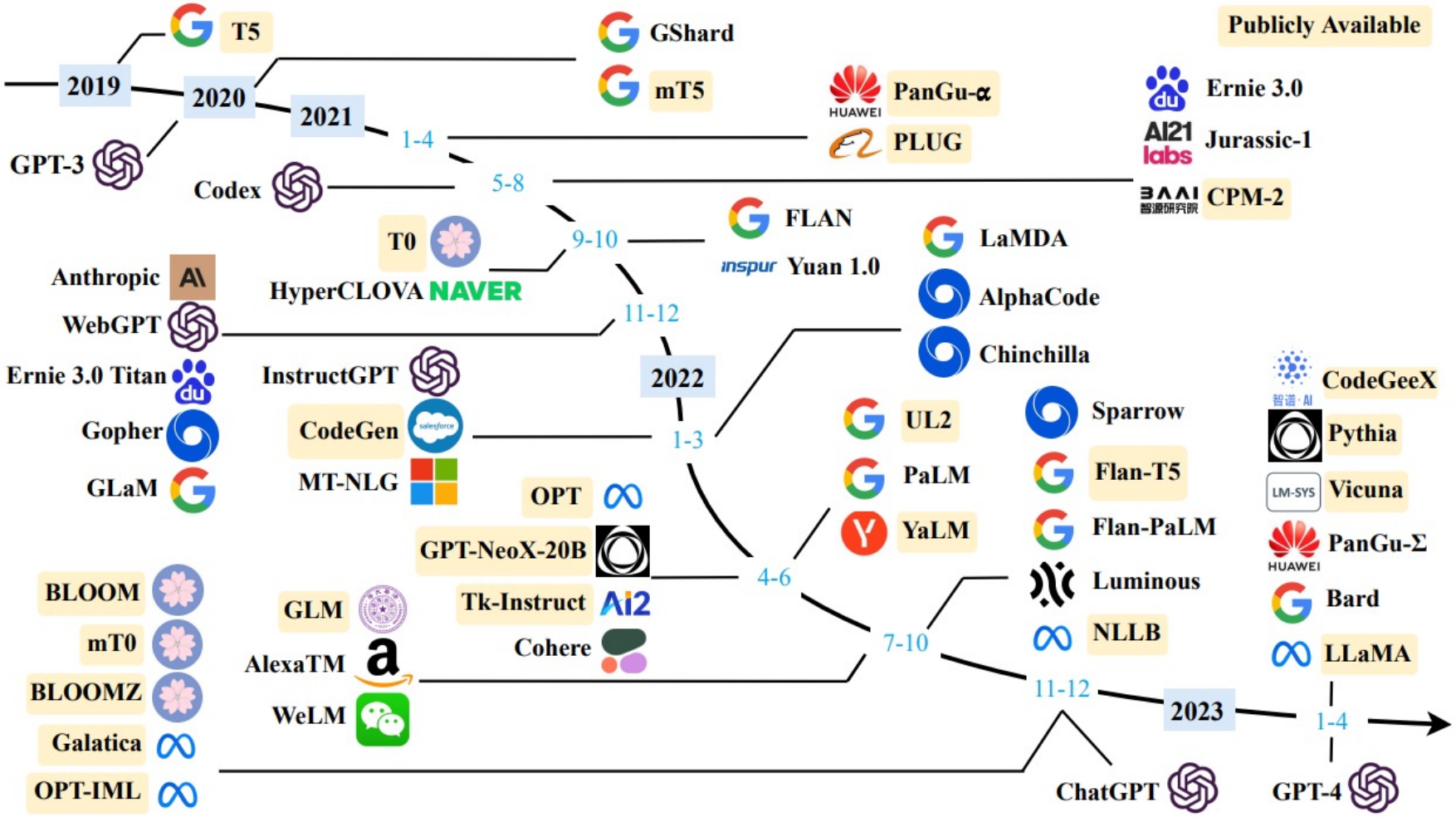
(b) Projections for high-quality language data



(c) Projections for vision data

1 – Large Language Models

A timeline of existing large language models



1 – Large Language Models

T5 Model

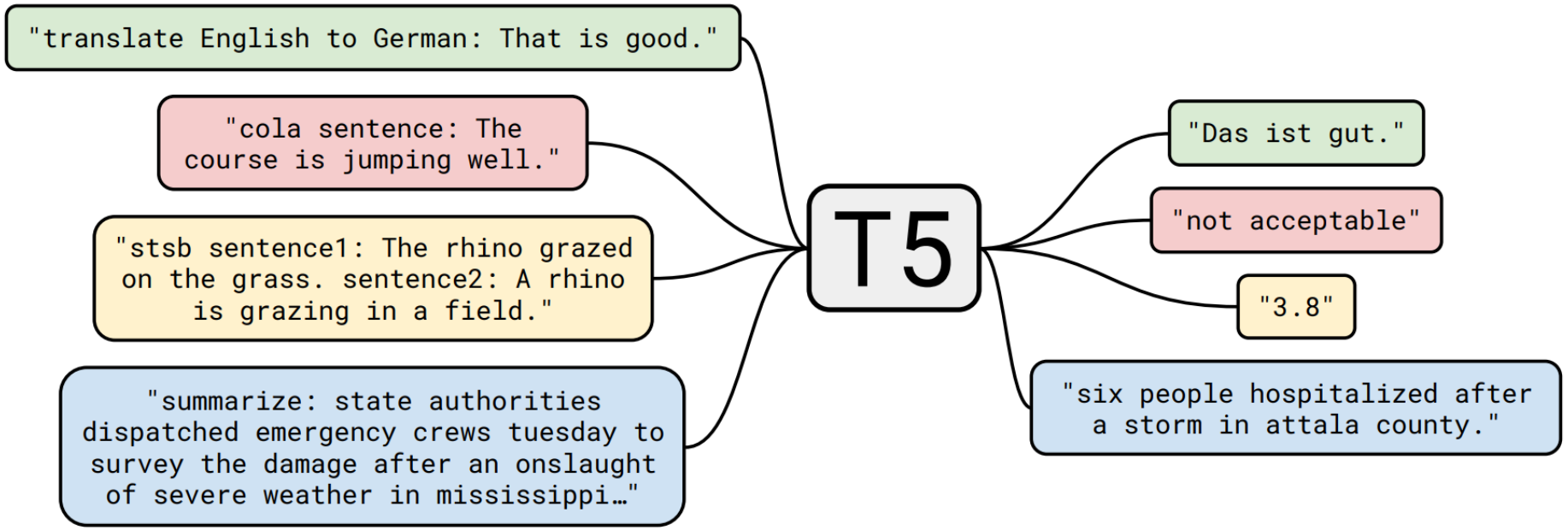
- Text-to-Text Transfer Transformer

Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer

1 – Large Language Models

T5 Model

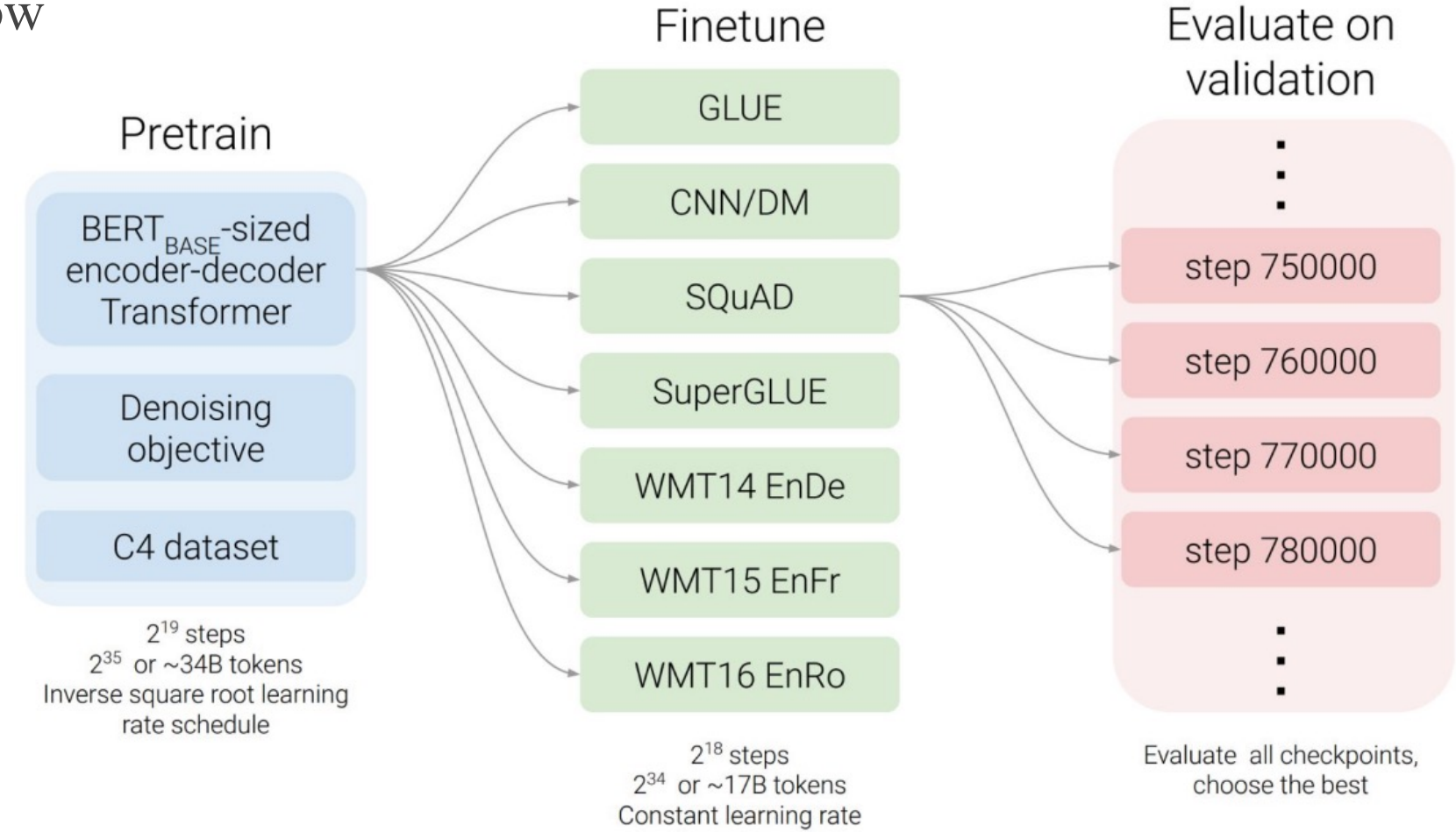
- Every task, one format!
- [“Task-specific prefix]: [Input text]” => “[Output text]”



1 – Large Language Models

T5 Model

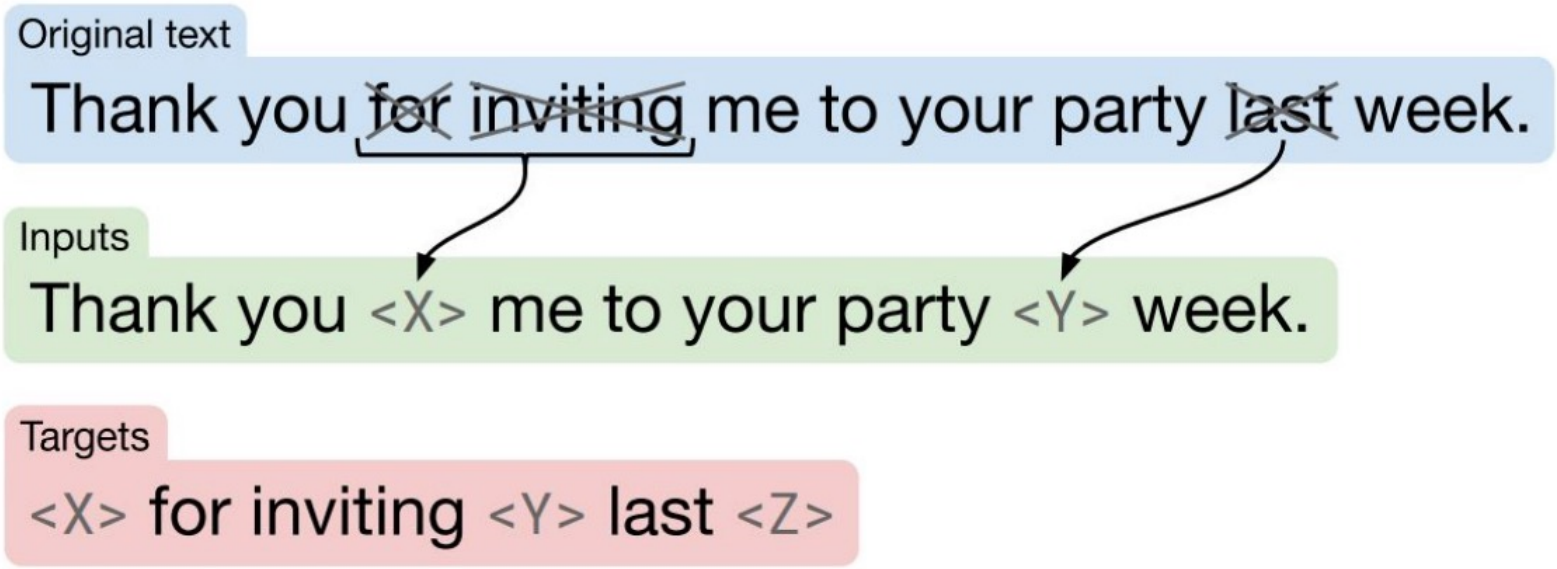
- Workflow



1 – Large Language Models

T5 Model

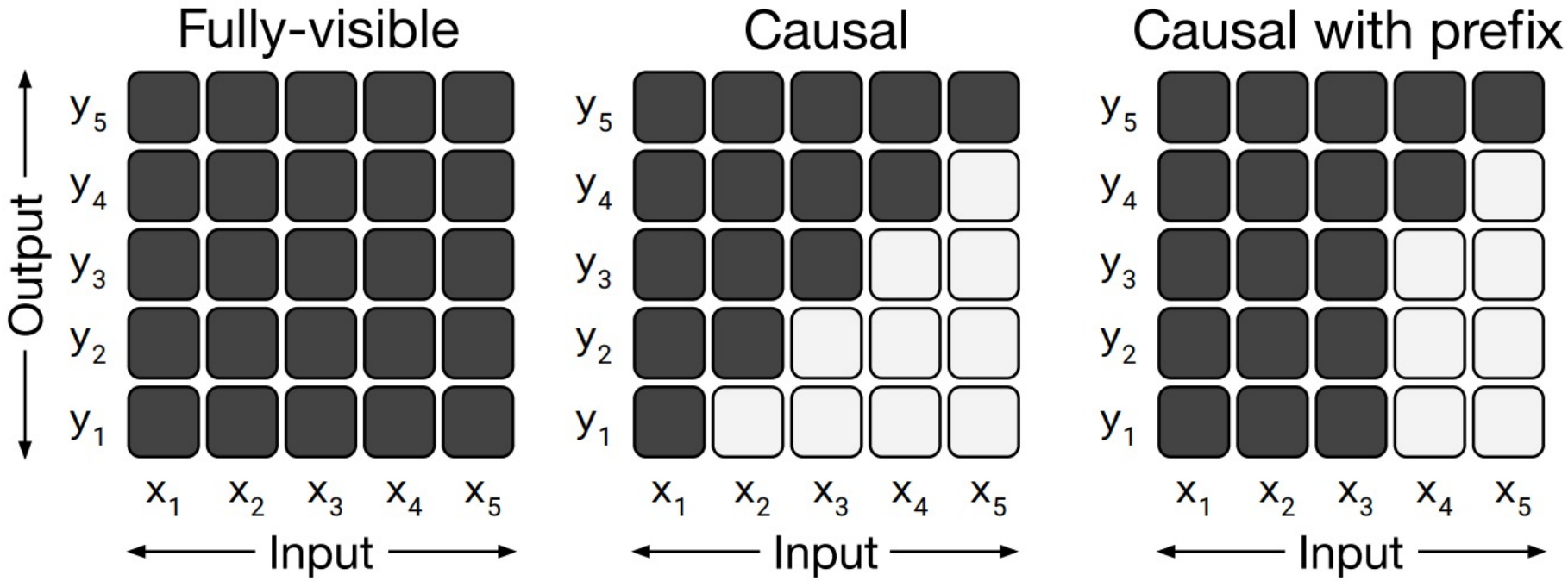
- Baseline Objective



1 – Large Language Models

T5 Model

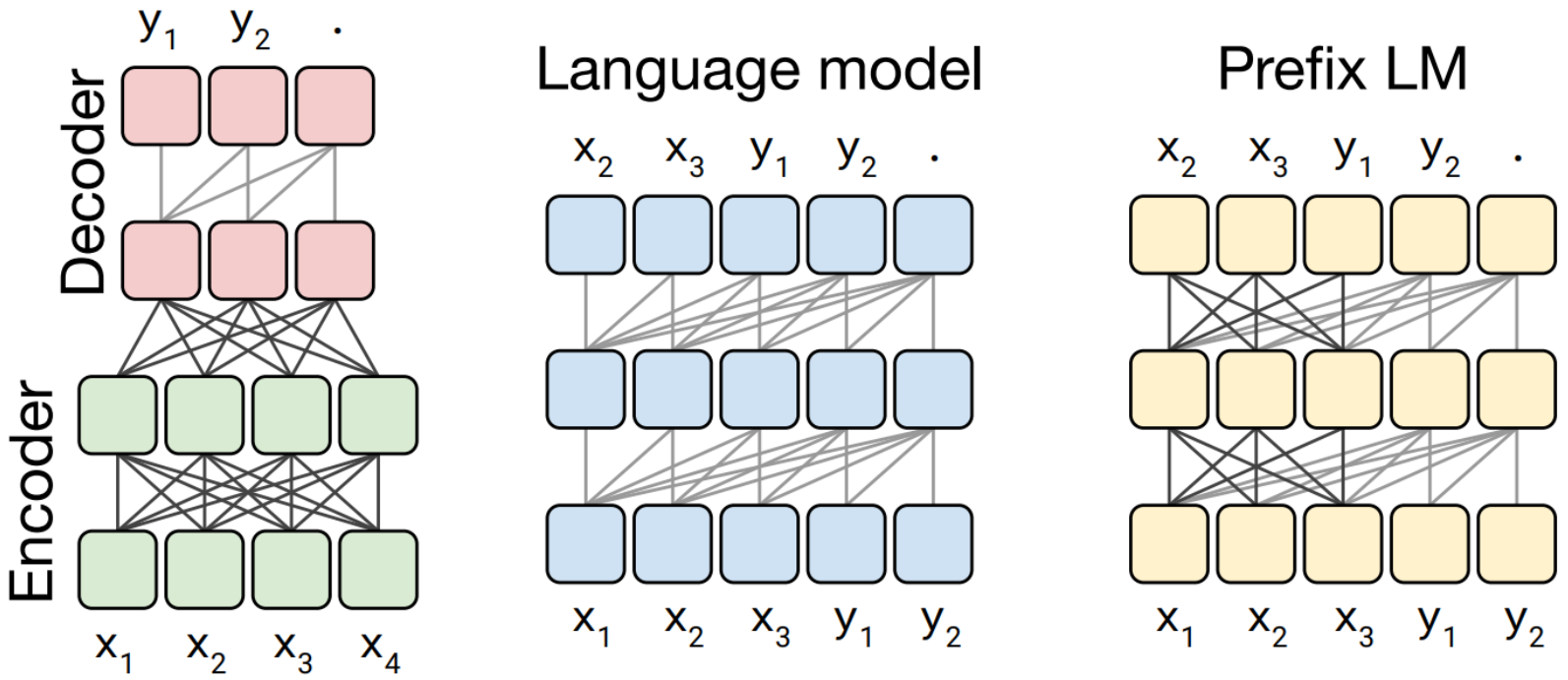
- Different Attention Mask Patterns



1 – Large Language Models

T5 Model

- Transformer architecture variants



1 – Large Language Models

T5 Model

- Transformer architecture variants

Architecture	Objective	Params	Cost	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Encoder-decoder	Denoising	$2P$	M	**83.28**	**19.24**	**80.88**	**71.36**	**26.98**	**39.82**	**27.65**
Enc-dec, shared	Denoising	P	M	82.81	18.78	**80.63**	**70.73**	26.72	39.03	**27.46**
Enc-dec, 6 layers	Denoising	P	$M/2$	80.88	18.97	77.59	68.42	26.38	38.40	26.95
Language model	Denoising	P	M	74.70	17.93	61.14	55.02	25.09	35.28	25.86
Prefix LM	Denoising	P	M	81.82	18.61	78.94	68.11	26.43	37.98	27.39
Encoder-decoder	LM	$2P$	M	79.56	18.59	76.02	64.29	26.27	39.17	26.86
Enc-dec, shared	LM	P	M	79.60	18.13	76.35	63.50	26.62	39.17	27.05
Enc-dec, 6 layers	LM	P	$M/2$	78.67	18.26	75.32	64.06	26.13	38.42	26.89
Language model	LM	P	M	73.78	17.54	53.81	56.51	25.23	34.31	25.38
Prefix LM	LM	P	M	79.68	17.84	76.87	64.86	26.28	37.51	26.76

1 – Large Language Models

T5 Model

- Different Unsupervised Objectives

Objective	Inputs	Targets
Prefix language modeling	Thank you for inviting	me to your party last week .
BERT-style Devlin et al. (2018)	Thank you <M> <M> me to your party apple week .	*(original text)*
Deshuffling	party me for your to . last fun you inviting week Thank	*(original text)*

High-level approaches

- Language modeling
- BERT-style
- Deshuffling

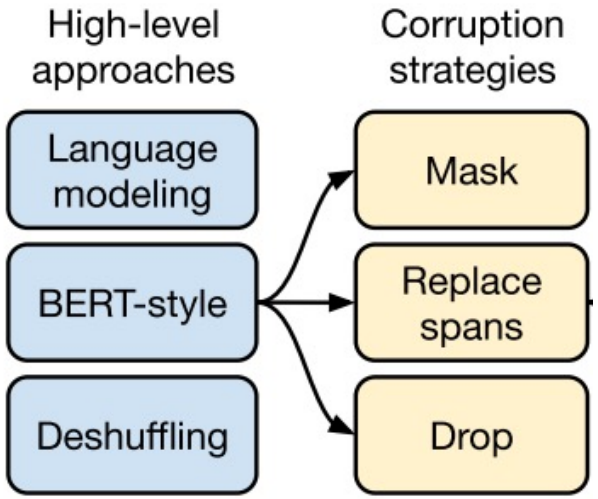
Objective	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
Prefix language modeling	80.69	18.94	77.99	65.27	**26.86**	39.73	**27.49**
BERT-style (Devlin et al., 2018)	**82.96**	**19.17**	**80.65**	**69.85**	**26.78**	**40.03**	**27.41**
Deshuffling	73.17	18.59	67.61	58.47	26.11	39.30	25.62

1 – Large Language Models

T5 Model

- Different Unsupervised Objectives

Objective	Inputs	Targets
MASS-style Song et al. (2019)	Thank you <M> <M> me to your party <M> week .	*(original text)*
I.i.d. noise, replace spans	Thank you <X> me to your party <Y> week .	<X> for inviting <Y> last <Z>
I.i.d. noise, drop tokens	Thank you me to your party week .	for inviting last
Random spans	Thank you <X> to <Y> week .	<X> for inviting me <Y> your party last <Z>

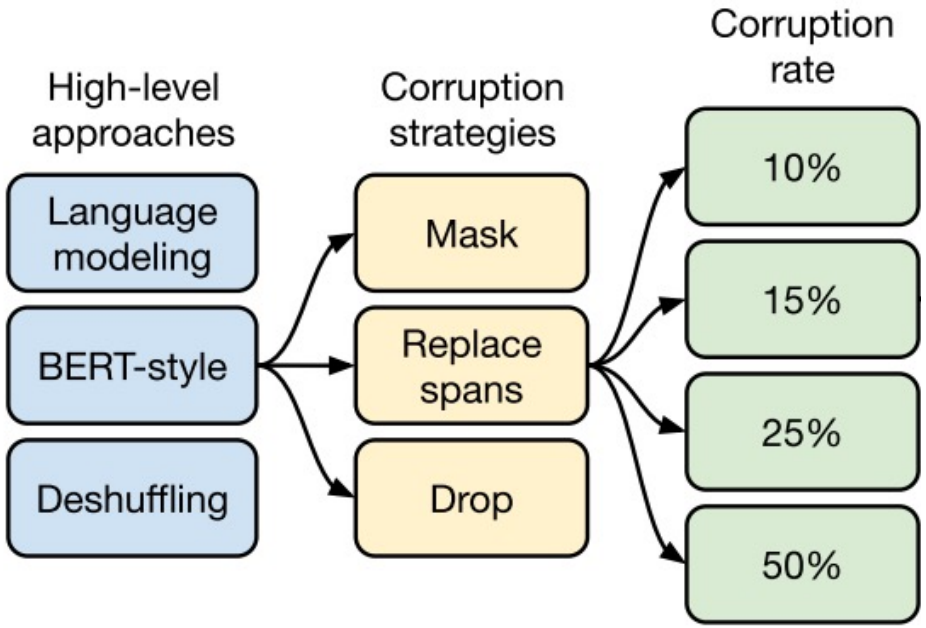


Objective	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
BERT-style (Devlin et al., 2018)	82.96	19.17	**80.65**	69.85	26.78	**40.03**	27.41
MASS-style (Song et al., 2019)	82.32	19.16	80.10	69.28	26.79	**39.89**	27.55
★ Replace corrupted spans	83.28	**19.24**	**80.88**	**71.36**	**26.98**	39.82	**27.65**
Drop corrupted tokens	**84.44**	**19.31**	**80.52**	68.67	**27.07**	39.76	**27.82**

1 – Large Language Models

T5 Model

- Different Unsupervised Objectives

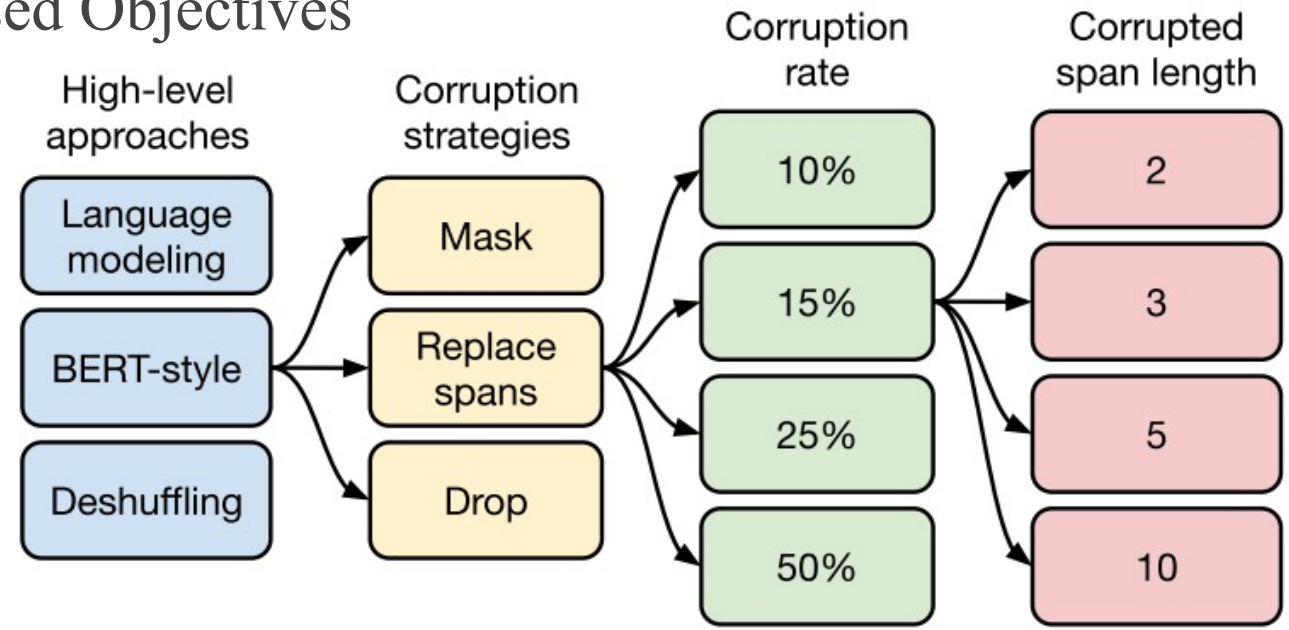


Corruption rate	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
10%	**82.82**	19.00	**80.38**	69.55	**26.87**	39.28	**27.44**
★ 15%	**83.28**	19.24	**80.88**	**71.36**	**26.98**	**39.82**	**27.65**
25%	**83.00**	**19.54**	**80.96**	70.48	**27.04**	**39.83**	**27.47**
50%	81.27	19.32	79.80	70.33	**27.01**	**39.90**	**27.49**

1 – Large Language Models

T5 Model

- Different Unsupervised Objectives

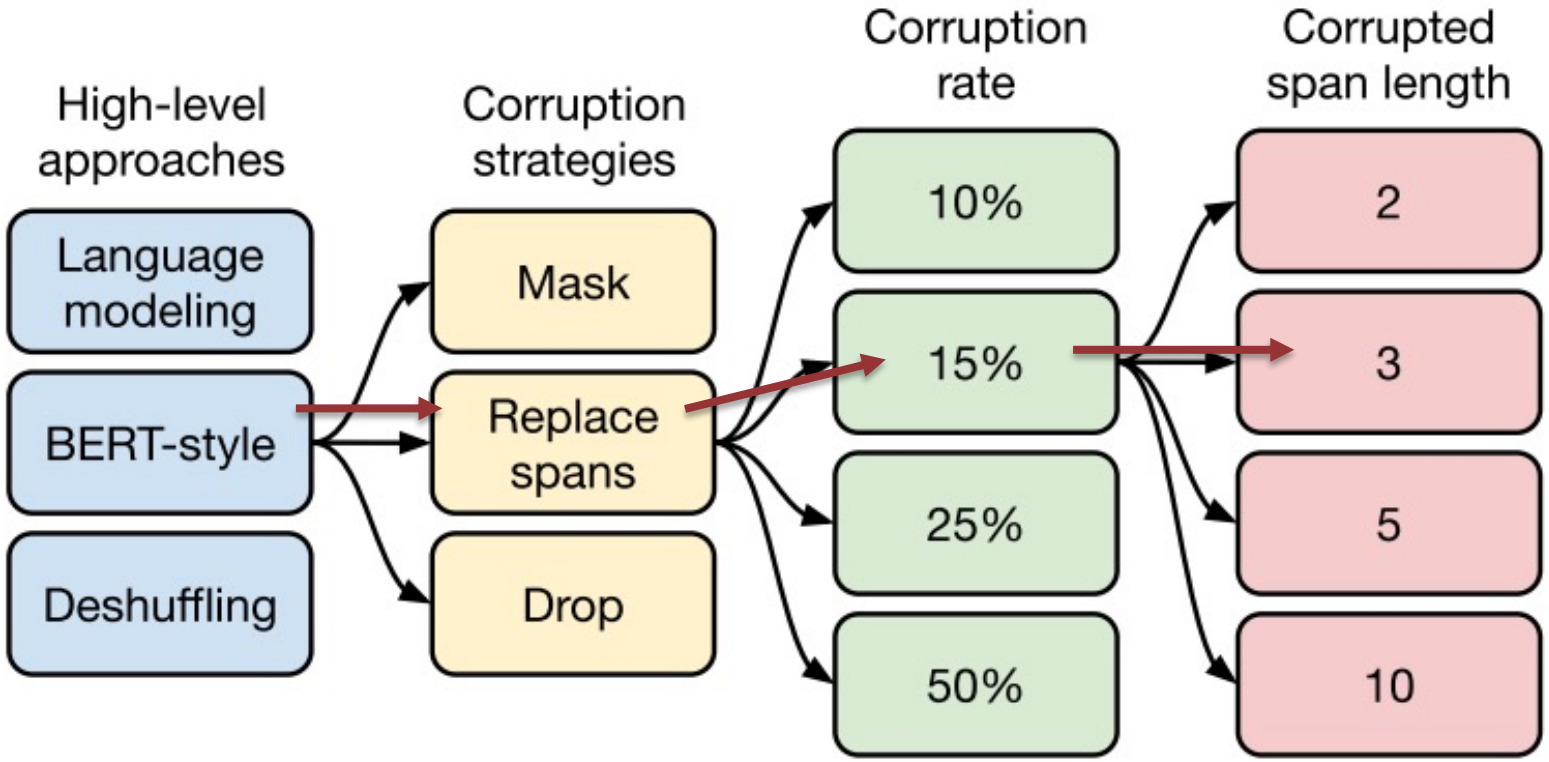


Span length	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Baseline (i.i.d.)	**83.28**	19.24	80.88	71.36	**26.98**	**39.82**	**27.65**
2	**83.54**	19.39	**82.09**	**72.20**	**26.76**	**39.99**	**27.63**
3	**83.49**	**19.62**	**81.84**	**72.53**	**26.86**	39.65	**27.62**
5	**83.40**	19.24	**82.05**	**72.23**	**26.88**	39.40	**27.53**
10	82.85	19.33	**81.84**	70.44	**26.79**	39.49	**27.69**

1 – Large Language Models

T5 Model

- Different Unsupervised Objectives

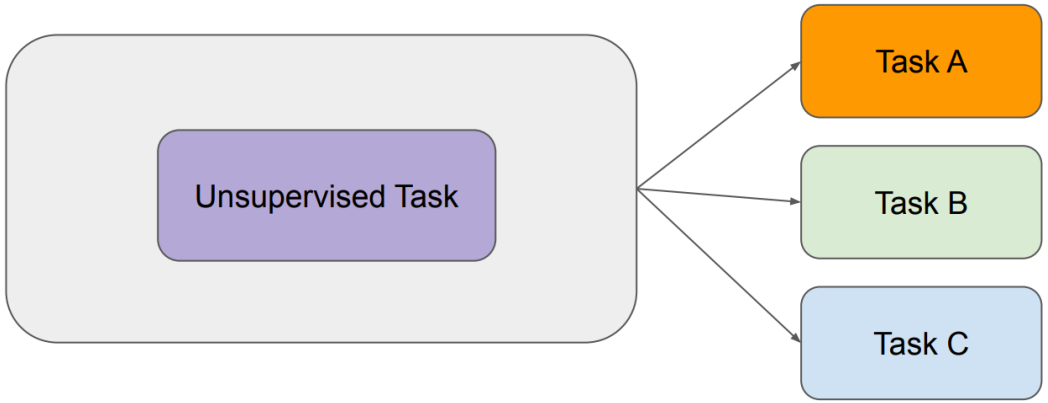


1 – Large Language Models

T5 Model

- Multi-task

Training strategy	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Unsupervised pre-training + fine-tuning	**83.28**	**19.24**	**80.88**	**71.36**	**26.98**	39.82	27.65
Multi-task training	81.42	**19.24**	79.78	67.30	25.21	36.30	27.76
Multi-task pre-training + fine-tuning	**83.11**	**19.12**	**80.26**	**71.03**	**27.08**	39.80	**28.07**
Leave-one-out multi-task training	81.98	19.05	79.97	**71.68**	**26.93**	39.79	**27.87**
Supervised multi-task pre-training	79.93	18.96	77.38	65.36	26.81	**40.13**	**28.04**

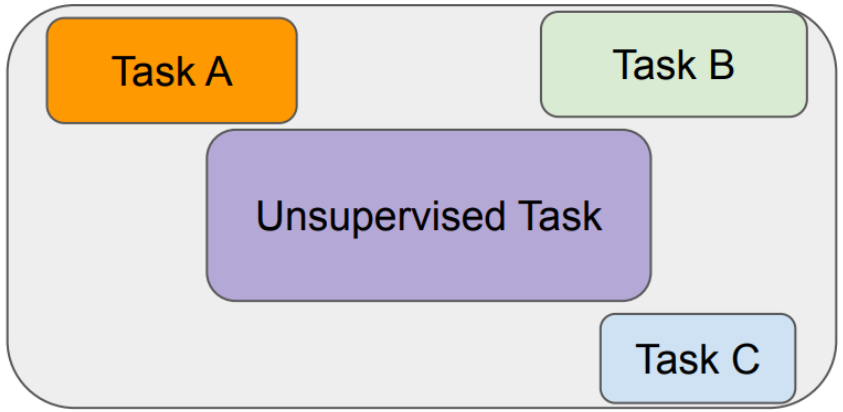


1 – Large Language Models

T5 Model

- Multi-task

Training strategy	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Unsupervised pre-training + fine-tuning	**83.28**	**19.24**	**80.88**	**71.36**	**26.98**	39.82	27.65
Multi-task training	81.42	**19.24**	79.78	67.30	25.21	36.30	27.76
Multi-task pre-training + fine-tuning	**83.11**	**19.12**	**80.26**	**71.03**	**27.08**	39.80	**28.07**
Leave-one-out multi-task training	81.98	19.05	79.97	**71.68**	**26.93**	39.79	**27.87**
Supervised multi-task pre-training	79.93	18.96	77.38	65.36	26.81	**40.13**	**28.04**

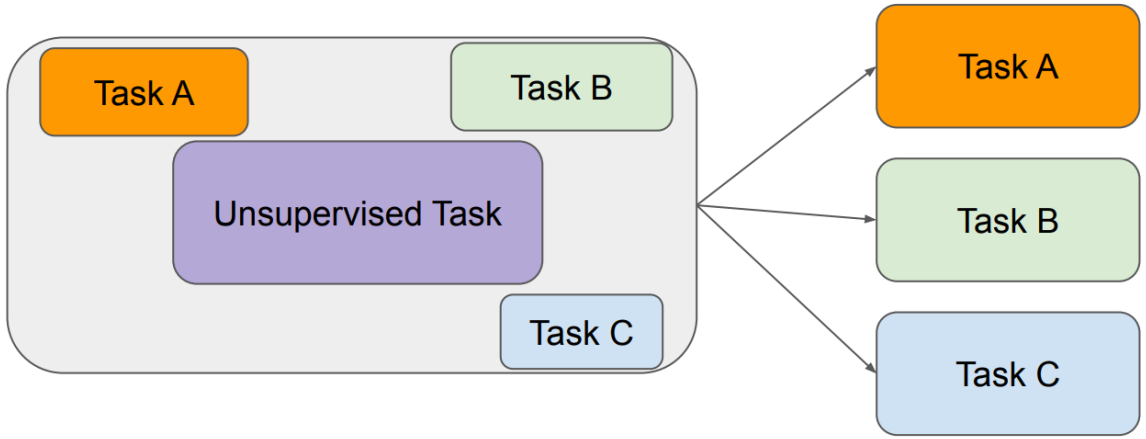


1 – Large Language Models

T5 Model

- Multi-task

Training strategy	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Unsupervised pre-training + fine-tuning	**83.28**	**19.24**	**80.88**	**71.36**	**26.98**	39.82	27.65
Multi-task training	81.42	**19.24**	79.78	67.30	25.21	36.30	27.76
Multi-task pre-training + fine-tuning	**83.11**	**19.12**	**80.26**	**71.03**	**27.08**	39.80	**28.07**
Leave-one-out multi-task training	81.98	19.05	79.97	**71.68**	**26.93**	39.79	**27.87**
Supervised multi-task pre-training	79.93	18.96	77.38	65.36	26.81	**40.13**	**28.04**

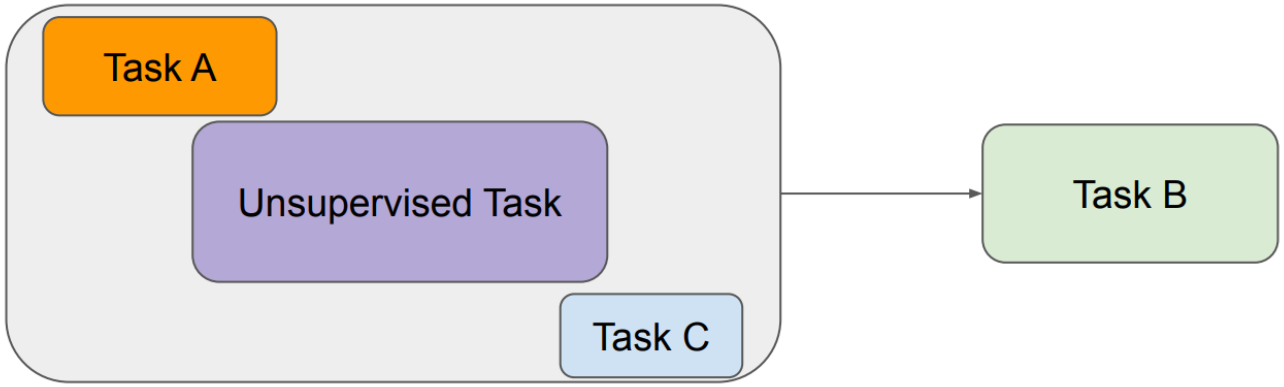


1 – Large Language Models

T5 Model

- Multi-task

Training strategy	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Unsupervised pre-training + fine-tuning	**83.28**	**19.24**	**80.88**	**71.36**	**26.98**	39.82	27.65
Multi-task training	81.42	**19.24**	79.78	67.30	25.21	36.30	27.76
Multi-task pre-training + fine-tuning	**83.11**	**19.12**	**80.26**	**71.03**	**27.08**	39.80	**28.07**
Leave-one-out multi-task training	81.98	19.05	79.97	**71.68**	**26.93**	39.79	**27.87**
Supervised multi-task pre-training	79.93	18.96	77.38	65.36	26.81	**40.13**	**28.04**

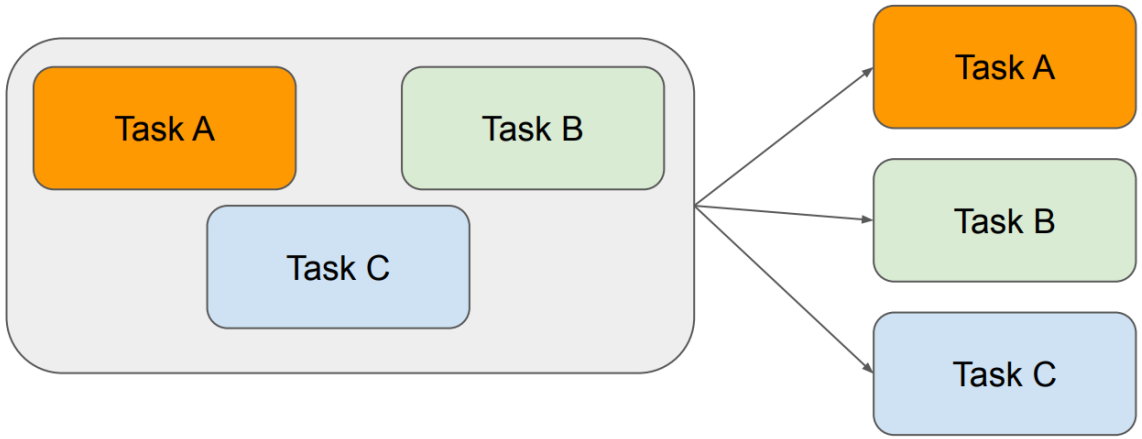


1 – Large Language Models

T5 Model

- Multi-task

Training strategy	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Unsupervised pre-training + fine-tuning	**83.28**	**19.24**	**80.88**	**71.36**	**26.98**	39.82	27.65
Multi-task training	81.42	**19.24**	79.78	67.30	25.21	36.30	27.76
Multi-task pre-training + fine-tuning	**83.11**	**19.12**	**80.26**	**71.03**	**27.08**	39.80	**28.07**
Leave-one-out multi-task training	81.98	19.05	79.97	**71.68**	**26.93**	39.79	**27.87**
Supervised multi-task pre-training	79.93	18.96	77.38	65.36	26.81	**40.13**	**28.04**



1 – Large Language Models

T5 Model

- Model Variants

Model	Parameters	No. of layers	d_{model}	d_{ff}	d_{kv}	No. of heads
Small	60M	6	512	2048	64	8
Base	220M	12	768	3072	64	12
Large	770M	24	1024	4096	64	16
3B	3B	24	1024	16384	128	32
11B	11B	24	1024	65536	128	128

Model	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
Previous best	89.4	20.30	95.5	84.6	**33.8**	**43.8**	**38.5**
T5-Small	77.4	19.56	87.24	63.3	26.7	36.0	26.8
T5-Base	82.7	20.34	92.08	76.2	30.9	41.2	28.0
T5-Large	86.4	20.68	93.79	82.3	32.0	41.5	28.1
T5-3B	88.5	21.02	94.95	86.4	31.8	42.6	28.2
T5-11B	**89.7**	**21.55**	**95.64**	**88.9**	32.1	43.4	28.1

1 – Large Language Models

BLOOM

- BLOOM is a decoder-only Transformer language model
- Training data: ROOTs Corpus – 46 natural and 13 programming language

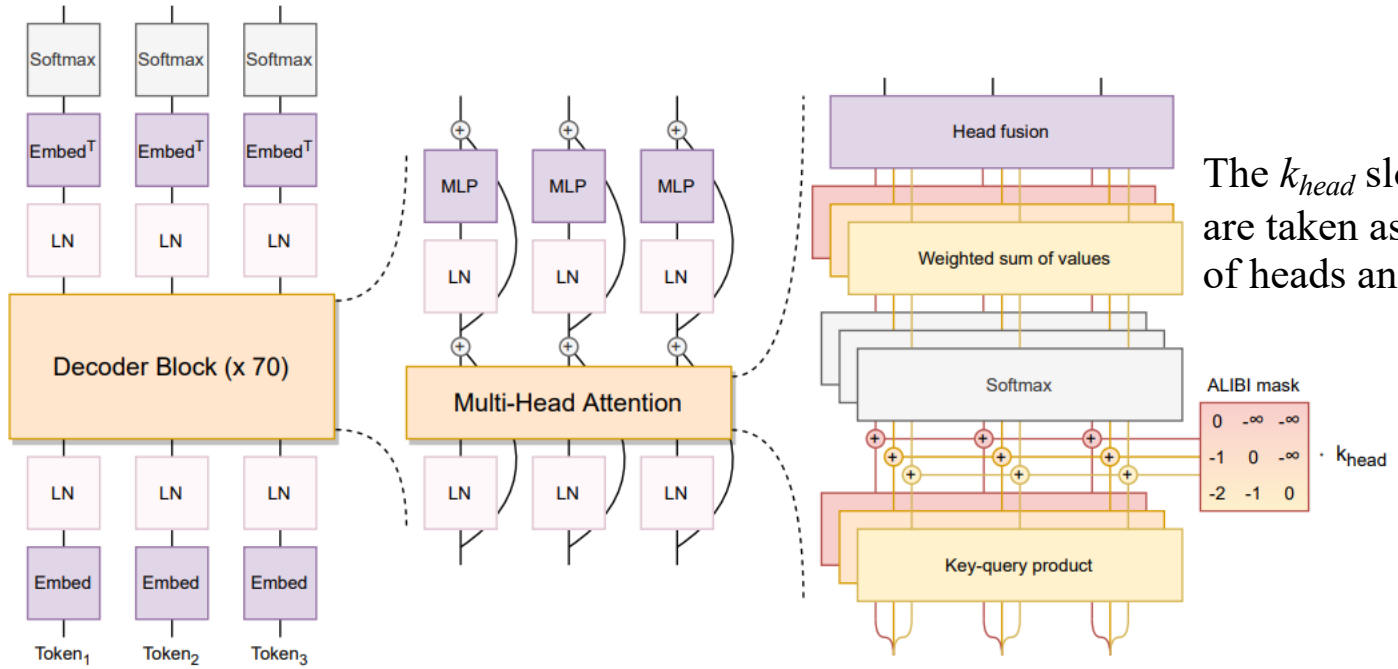
BLOOM: A 176B-Parameter Open-Access Multilingual Language Model

Language	ISO-639-3	catalog-ref	Genus	Family	Macroarea	Size in Bytes
Twi	twi	tw	Kwa	Niger-Congo	Africa	1,265,041
Urdu	urd	ur	Indic	Indo-European	Eurasia	2,781,329,959
Vietnamese	vie	vi	Viet-Muong	Austro-Asiatic	Eurasia	43,709,279,959
Wolof	wol	wo	Wolof	Niger-Congo	Africa	3,606,973
Xitsonga	tso	ts	Bantoid	Niger-Congo	Africa	707,634
Yoruba	yor	yo	Defoid	Niger-Congo	Africa	89,695,835
Programming Languages	—	—	—	—		174,700,245,772

1 – Large Language Models

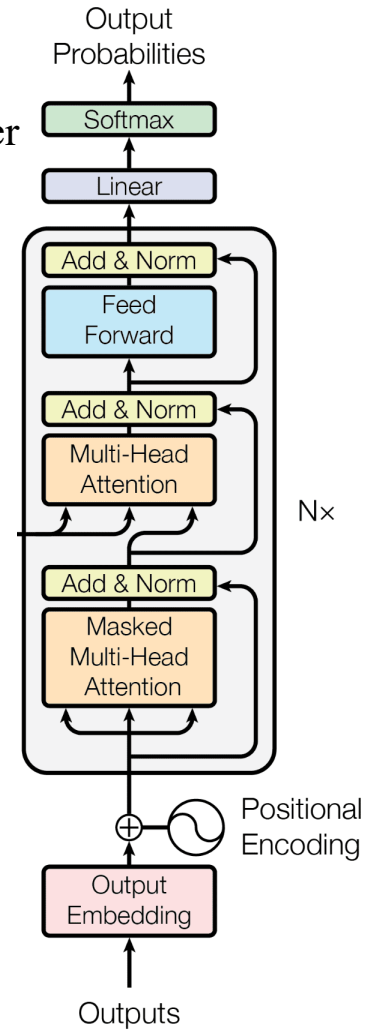
BLOOM

- Two architectural deviations in BLOOM
  - ALiBi Positional Embeddings
  - Embedding LayerNorm



The k_{head} slope parameters for ALiBi are taken as $2^{-8i/n}$ with n the number of heads and $i \in 1, 2, ..., n$.

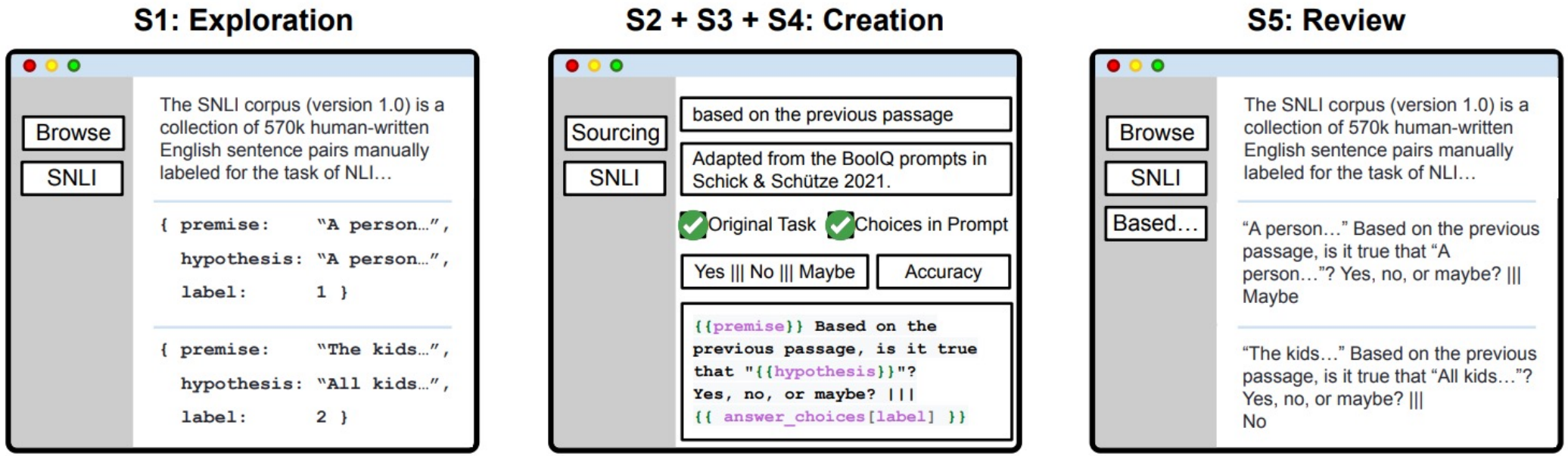
Original Transformer-Decoder



1 – Large Language Models

BLOOM

- Prompts (Based on PromptSource)



1 – Large Language Models

BLOOM

- Prompts (Based on PromptSource)

Label


```
{
  "text" : "string"
  "label" : [
    0 : "World"
    1 : "Sports"
    2 : "Business"
    3 : "Sci/Tech"
  ]
}
```


Template

```
Is this a piece of news regarding {{"world politics, sports, business, or science and technology"}}?
{{text}}
|||
{{answer_choices[label] }}
```

Input


```
{
  "text" :
  "Wall St. Bears Claw Back Into the
  Black (Reuters) Reuters - Short-
  sellers, Wall Street's
  dwindling\band of ultra-cynics,
  are seeing green again."
  "label" : 2
}
```


Prompt

Input

```
Is this a piece of news regarding world
politics, sports, business, or science
and technology?
Wall St. Bears Claw Back Into the Black
(Reuters) Reuters - Short-sellers, Wall
Street's dwindling\band of ultra-cynics,
are seeing green again.
```

Target

```
Business
```

1 – Large Language Models

BLOOM

- Prompts (Based on PromptSource)

Template

```
{{text}}
What label best describes this news article?
```

Label


```
{
  "text" : "string"
  "label" : [
    0 : "World"
    1 : "Sports"
    2 : "Business"
    3 : "Sci/Tech"
  ]
}
```


Input


```
{
  "text" :
  "Wall St. Bears Claw Back Into the Black (Reuters) Reuters - Short-sellers, Wall Street's dwindling\band of
  ultra-cynics, are seeing green again."
  "label" : 2
}
```


Prompt

Input

```
Wall St. Bears Claw Back Into the Black (Reuters) Reuters - Short-sellers, Wall
Street's dwindling\band of ultra-cynics, are seeing green again.
What label best describes this news article?
```

Target

```
Business
```

2 – Prompting Techniques

Advanced Prompting Techniques

- **Basic Prompting: Zero-shot, Few-shot**
- **Chain-of-Thought (CoT)**
- **Self-Consistency Sampling**
- **Automatic Prompt Engineer**
- **Prompt as Parameter-Efficient Fine-Tuning**

https://lilianweng.github.io/posts/2023-03-15-prompt-engineering/
https://learnprompting.org/

2 – Prompting Techniques

2.1. Basic Prompting

Three setting for In-Content-Learning

Few-shot

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

```
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
```

Zero-shot

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

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

One-shot

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

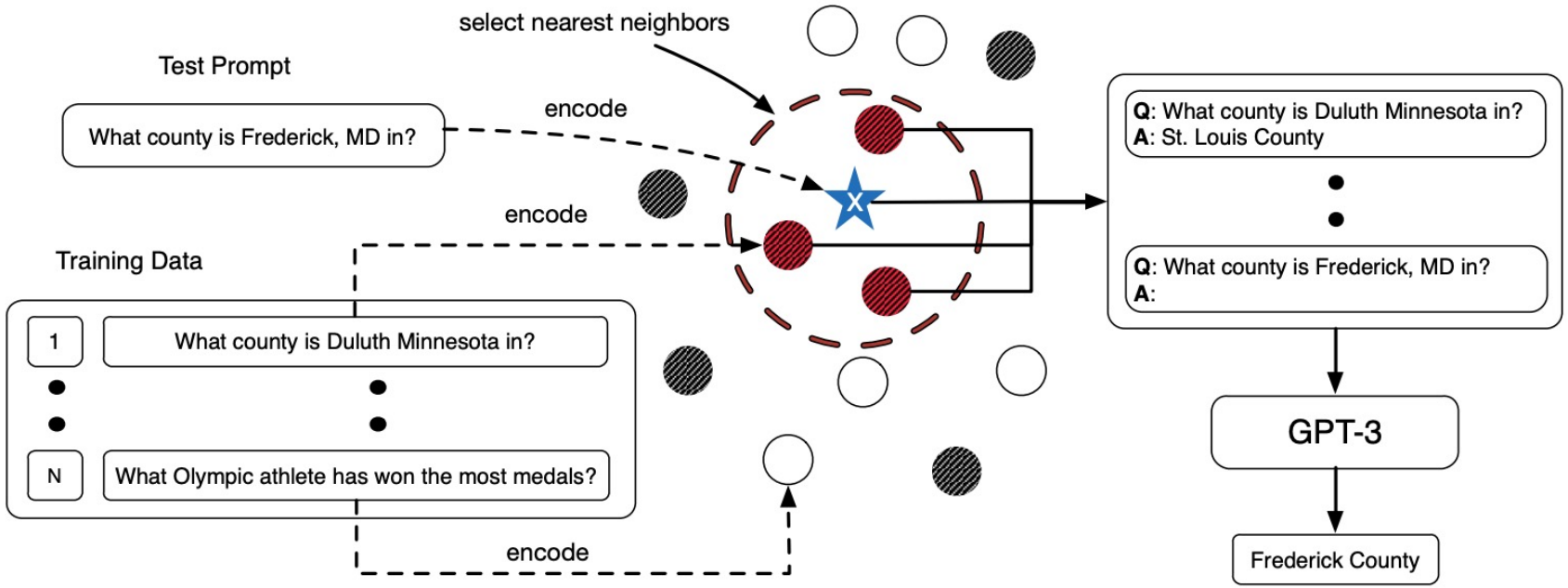
```
1  Translate English to French:        ← task description
2  sea otter => loutre de mer          ← example
3  cheese =>                           ← prompt
```

2 – Prompting Techniques

2.1. Basic Prompting

Example Selection, Ordering

- Choose examples that are semantically similar to the test example using k-NN clustering in the embedding space

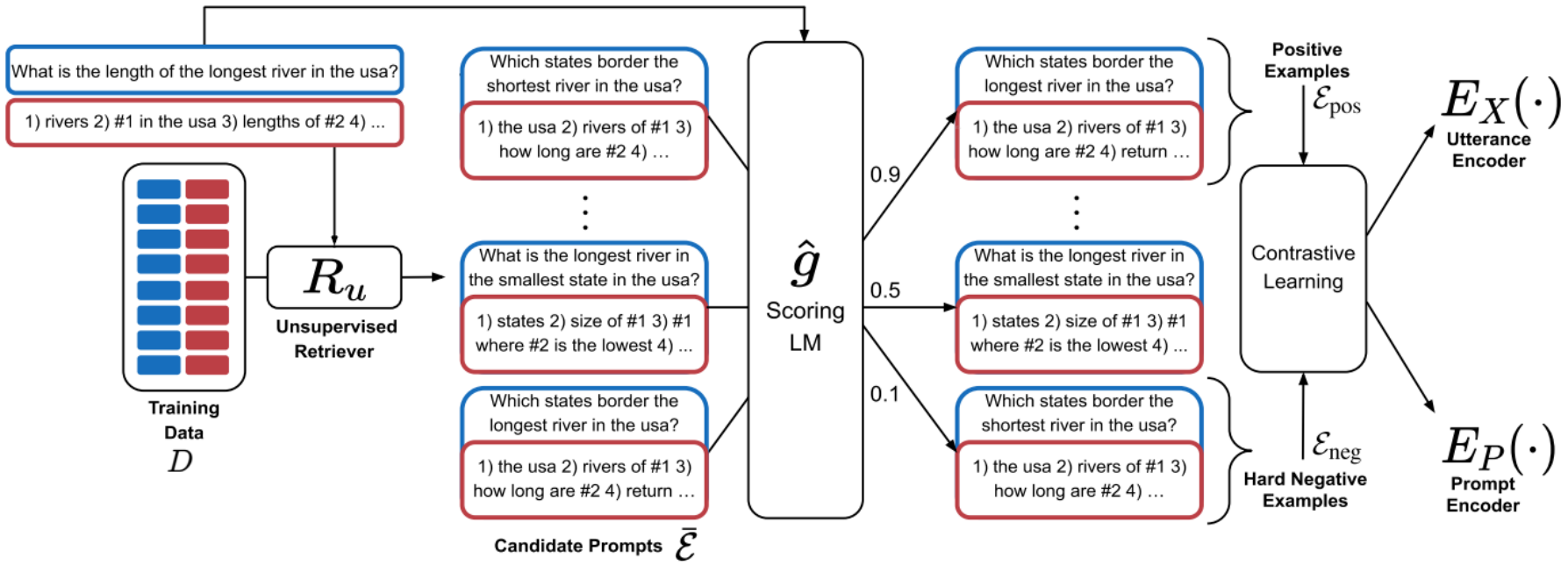


Source: What makes good In-content Examples for GPT-3

2 – Prompting Techniques

2.1. Basic Prompting

- **Example Selection, Ordering**
  - Based on contrastive learning

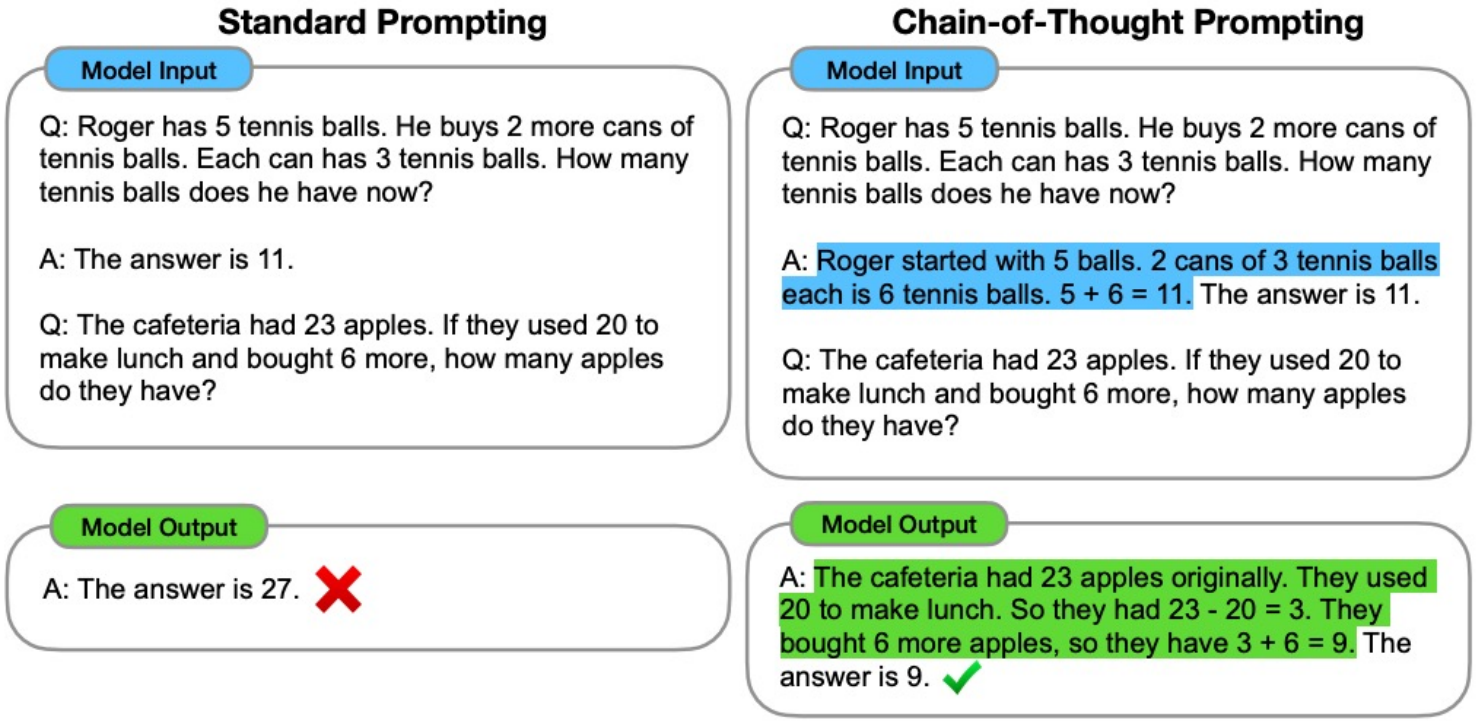


Source: [Learning To Retrieve Prompts for In-Context Learning]

2 – Prompting Techniques

2.2. Chain-of-Thought Prompting

- Generates a sequence of short sentences to describe reasoning logics step by step
- Helpful for complicated reasoning tasks



Source: Chain-of-Thought Prompting Elicits Reasoning in LLMs

2 – Prompting Techniques

2.2. Chain-of-Thought Prompting

- Few-shot-CoT

(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. ✗

(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. ✓

Source: Large Language Models are Zero-Shot Reasoners

2 – Prompting Techniques

2.2. Chain-of-Thought Prompting

- Zero-shot-CoT

(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 ✗

(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.***

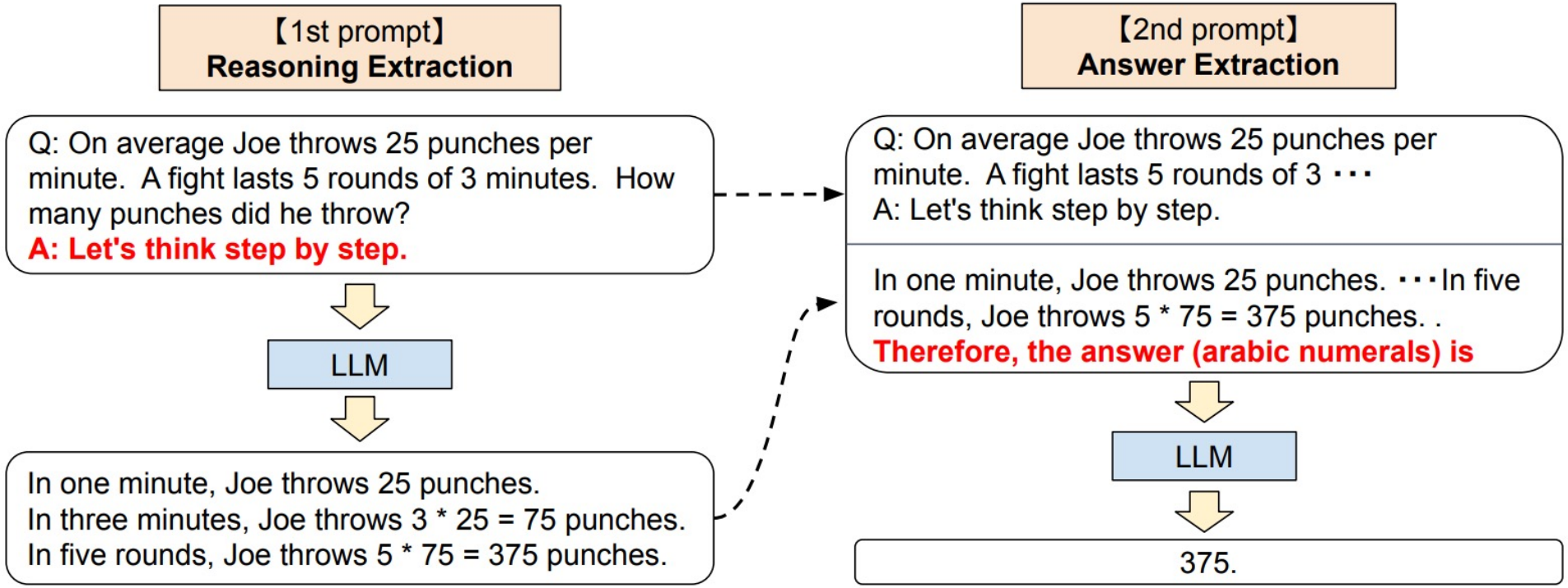
(Output) There are 16 balls in total. Half of the balls are golf balls. That means that there are 8 golf balls. Half of the golf balls are blue. That means that there are 4 blue golf balls. ✓

Source: Large Language Models are Zero-Shot Reasoners

2 – Prompting Techniques

2.2. Chain-of-Thought Prompting

- Zero-shot-CoT

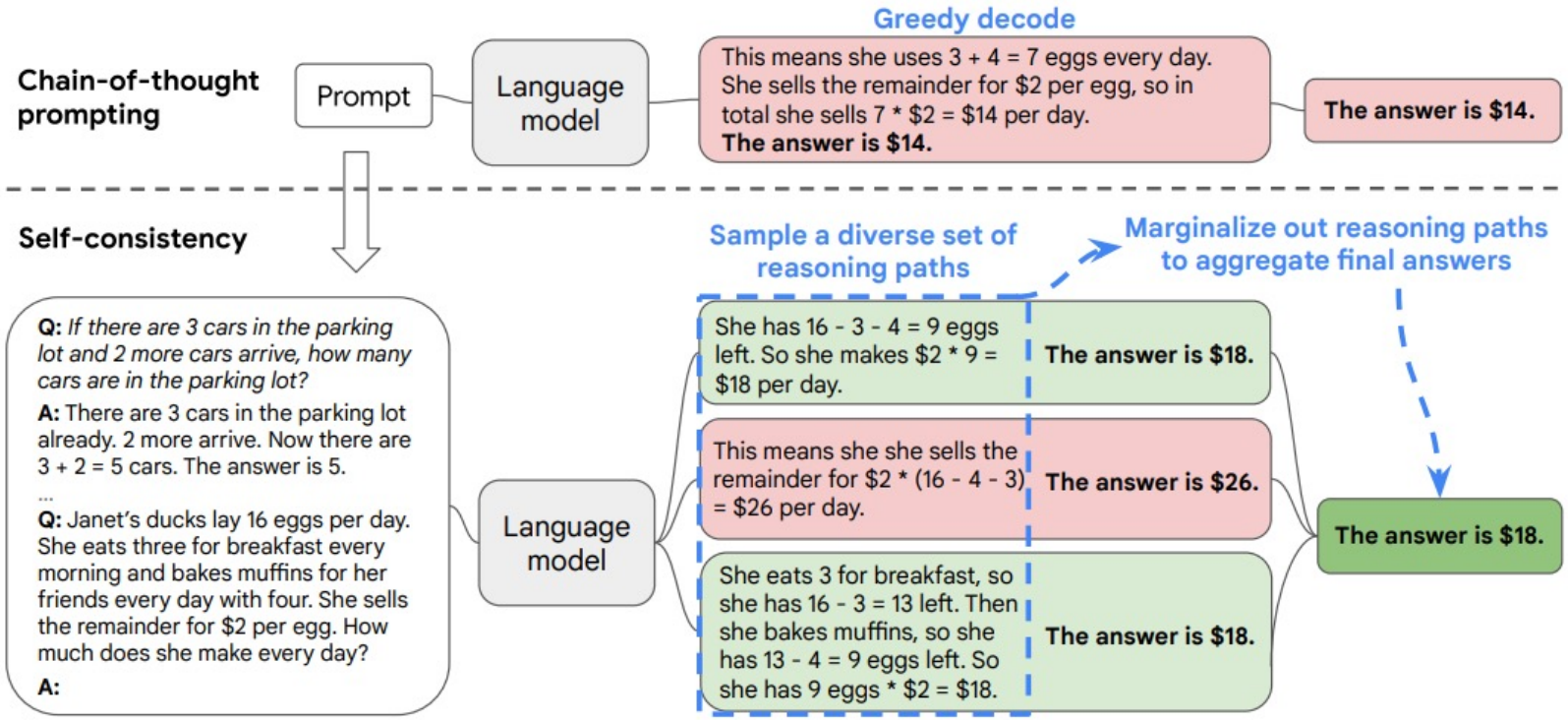


Source: Large Language Models are Zero-Shot Reasoners

2 – Prompting Techniques

2.3. Self-Consistency Sampling

- To sample multiple outputs with beam search, temperature > 0 and then selecting the best one out of these candidates



Source: [Chain-of-Thought Prompting Elicits Reasoning in LLMs]

2 – Prompting Techniques

2.3. Automatic Prompt Engineer

- LLMs as inference model to generate demonstrations and scoring model

Template for TruthfulQA

Professor Smith was given the following instructions: <INSERT>

Here are the Professor's responses:

Input: $[Q_1]$ **Output**: $[A_1]$
Input: $[Q_2]$ **Output**: $[A_2]$
...

Forward Generation Template

I gave a friend an instruction and five inputs. The friend read the instruction and wrote an output for every one of the inputs. Here are the input-output pairs:

Input: $[Q_1]$ **Output**: $[A_1]$
Input: $[Q_2]$ **Output**: $[A_2]$
...

The instruction was <COMPLETE>

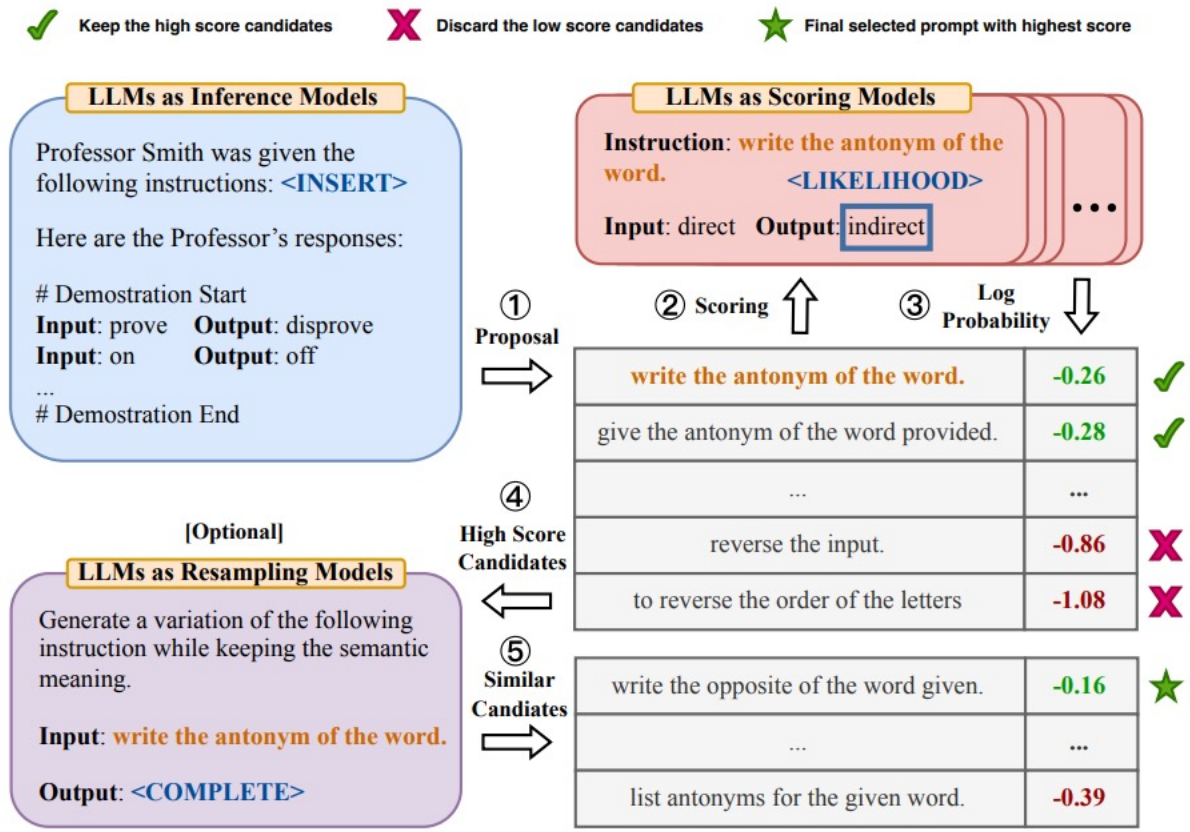
Reverse Generation Template

I instructed my friend to <INSERT>.

The friend read the instruction and wrote an output for every one of the inputs. Here are the input-output pairs:

Input: $[Q_1]$ **Output**: $[A_1]$
Input: $[Q_2]$ **Output**: $[A_2]$
...

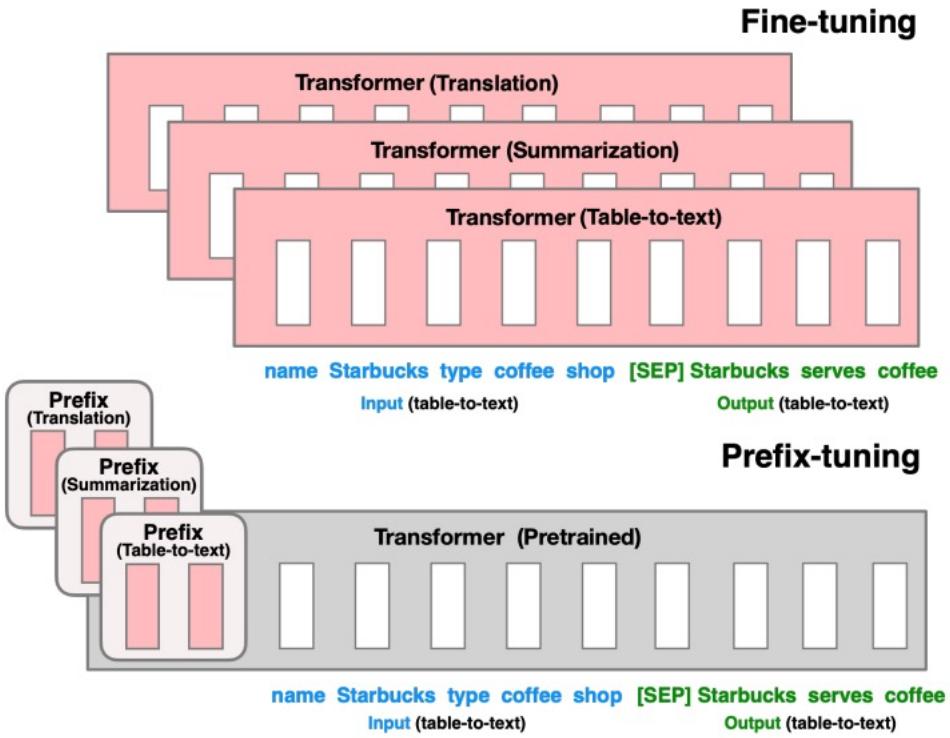
Source



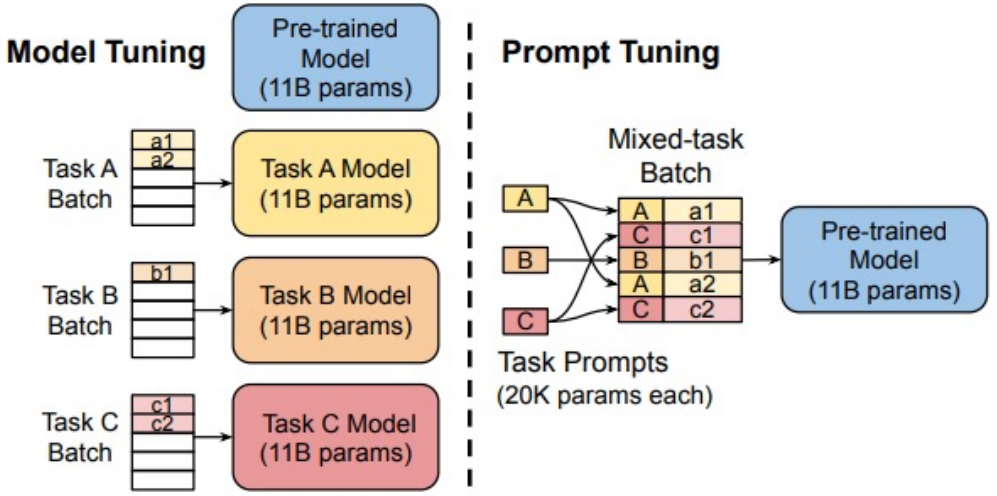
2 – Prompting Techniques

2.4. Prompt as Parameter-Efficient Fine-Tuning

- LLMs as inference model to generate demonstrations and scoring model



Prefix-Tuning



Prompt-Tuning

AI VIET NAM
@aivietnam.edu.vn

3 – Learning from HF (RLHF)

Overview: InstructGPT

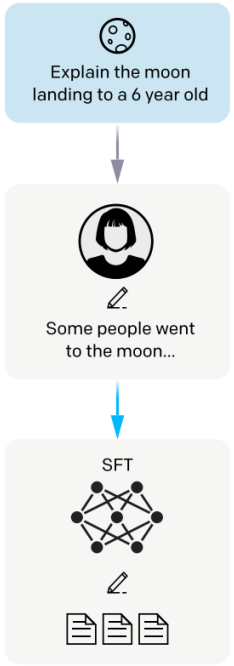
Step 1

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 with supervised learning.



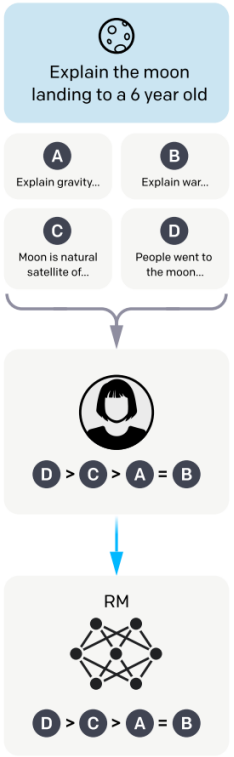
Step 2

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

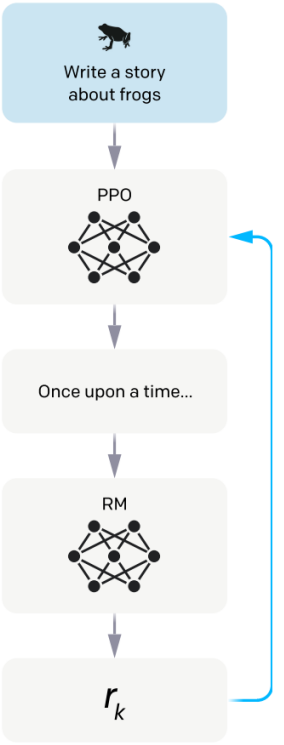
Optimize a policy against the reward model using reinforcement learning.

A new prompt is sampled from the dataset.

The policy generates an output.

The reward model calculates a reward for the output.

The reward is used to update the policy using PPO.

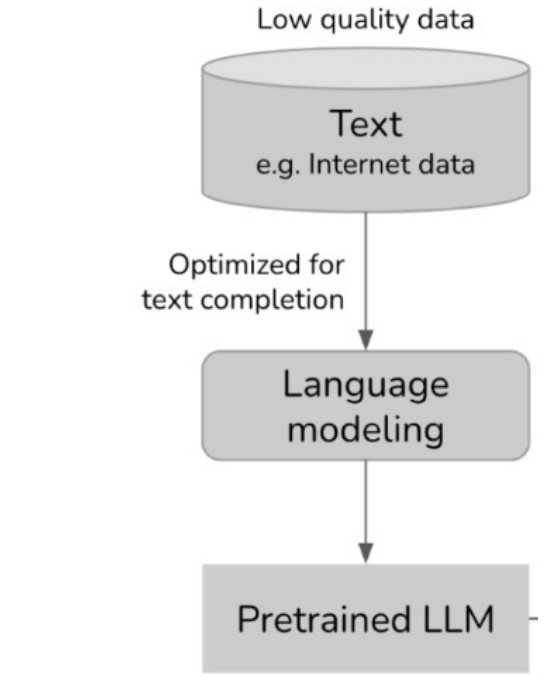


AI VIET NAM
@aivietnam.edu.vn

3 – Learning from HF (RLHF)

Pre-training LLMs

- LLM_φ: the language model being trained, parameterized by φ. The goal is to find φ for which the cross entropy loss is minimized.
- $[T_1, T_2, \ldots, T_V]$: vocabulary – the set of all unique tokens in the training data.
- V: the vocabulary size.
- $f(x)$: function mapping a token to its position in the vocab. If x is T_k in the vocab, $f(x) = k$.
- Given the sequence $(x_1, x_2, \ldots, x_n)$, we'll have n training samples:
  - Input: $x = (x_1, x_2, \ldots, x_{i-1})$
  - Ground truth: x_i
- For each training sample (x, x_i):
  - Let $k = f(x_i)$
  - Model's output: $LLM(x) = [\bar{y}_1, \bar{y}_2, \ldots, \bar{y}_V]$. Note: $\sum_j \bar{y}_j = 1$
  - The loss value: $CE(x, x_i; \varphi) = -\log \bar{y}_k$
- Goal: find φ to minimize the expected loss on all training samples. $CE(\varphi) = -E_x \log \bar{y}_k$

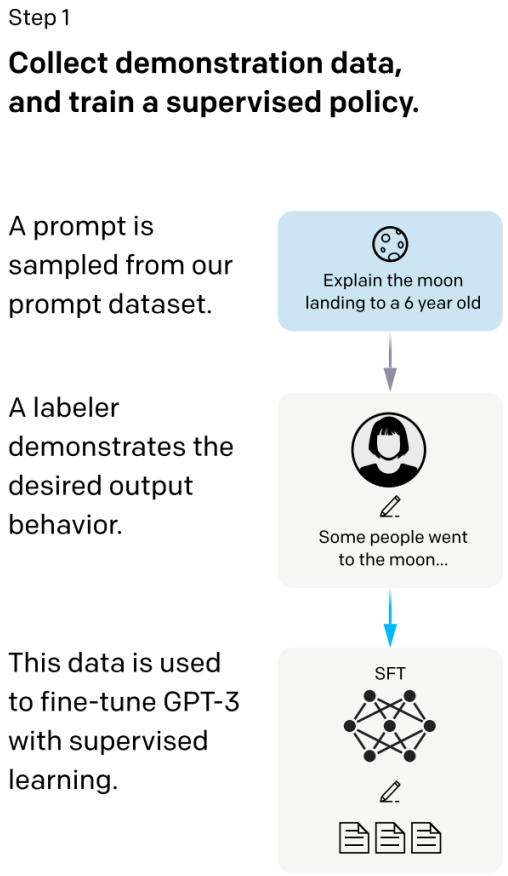


Scale May '23	**>1 trillion** tokens
Examples **Bolded**: open sourced	GPT-x, Gopher, **LLaMa** **Pythia**, **Bloom**, **StableL**

AI VIET NAM
@aivietnam.edu.vn

3 – Learning from HF (RLHF)

3.1. SFT Model

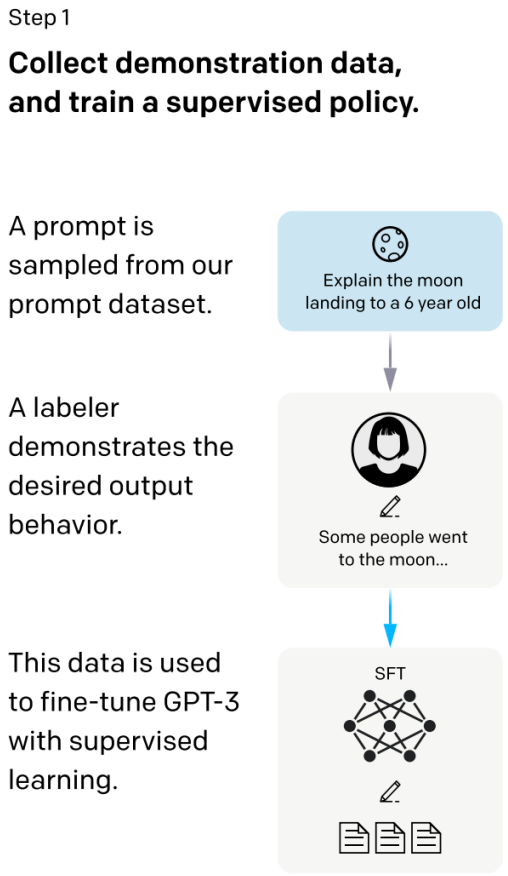


- Goal: optimize the LLM to generate the response that users are looking for

Prompt	Response
Serendipity means the occurrence and development of events by chance in a happy or beneficial way. Use the word in a sentence.	Running into Margaret and being introduced to Tom was a fortunate stroke of serendipity.
ELI5: What's the cause of the "anxiety lump" in our chest during stressful or disheartening experiences?	The anxiety lump in your throat is caused by muscular tension keeping your glottis dilated to maximize airflow. The clenched chest or heartache feeling is caused by the vagus nerve which tells the organs to pump blood faster, stop digesting, and produce adrenaline and cortisol.

3 – Learning from HF (RLHF)

3.1. SFT Model



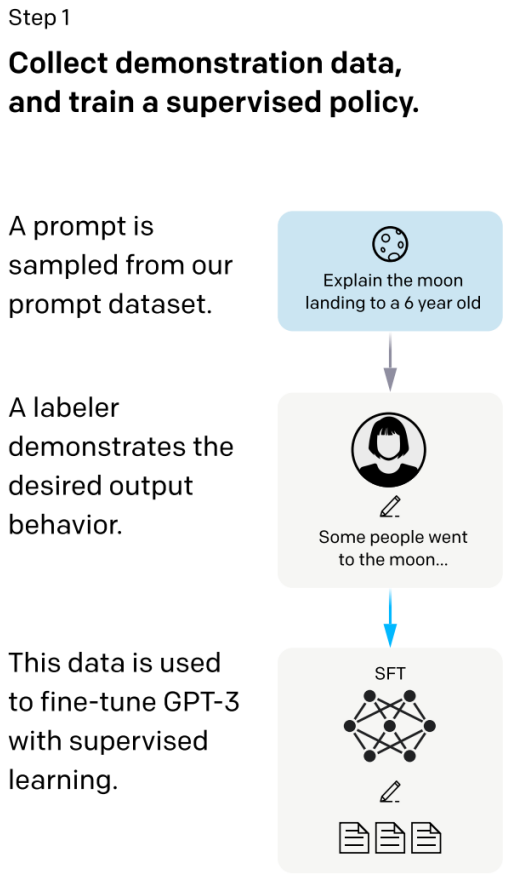
- A large collections of prompts:
  - Labeler-written prompts

Plain	Labelers to come up with an arbitrary task, while ensuring diversity of tasks
Few-shot	Labelers to come up with an instruction and multiple query/response pairs for that instruction “Given the sentiment for a tweet”
User-based	Collect use-cases stated in applications to the OpenAI API. Labelers to come up with prompts correspponding to these use-cases

AI VIET NAM
@aivietnam.edu.vn

3 – Learning from HF (RLHF)

3.1. SFT Model



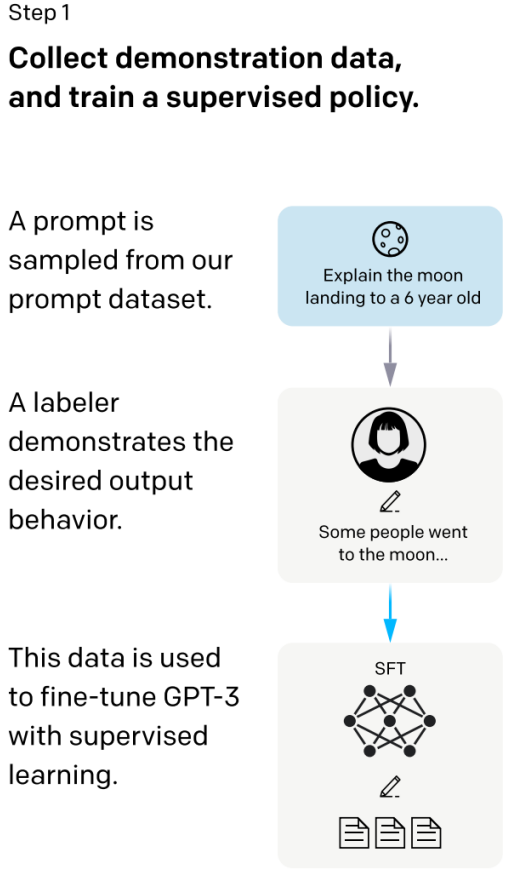
- A large collections of prompts:
  - Labeler-written prompts
  - API user prompts (From OpenAI GPT3 Playground)

- 200 prompts / per organization
- 10 use cases

AI VIET NAM
@aivietnam.edu.vn

3 – Learning from HF (RLHF)

3.1. SFT Model



- A large collections of prompts:

Table 1: Distribution of use case categories from our API prompt dataset.

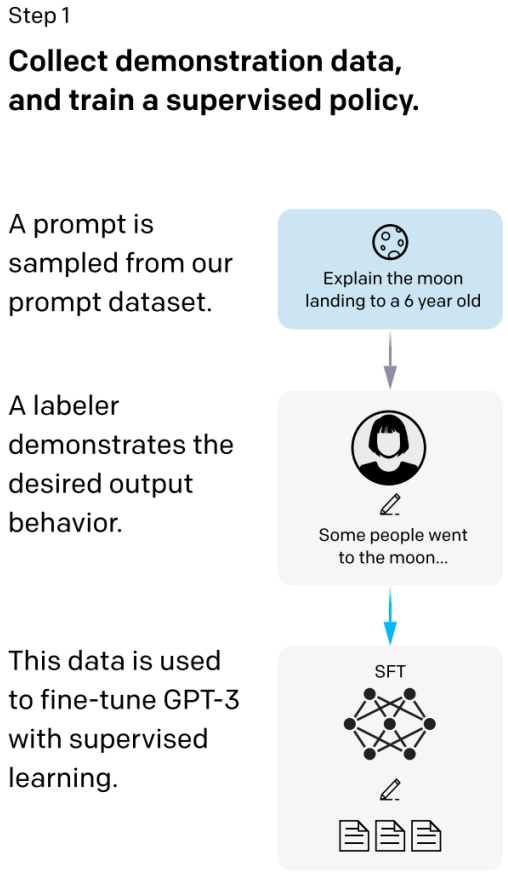
Use-case	(%)
Generation	45.6%
Open QA	12.4%
Brainstorming	11.2%
Chat	8.4%
Rewrite	6.6%
Summarization	4.2%
Classification	3.5%
Other	3.5%
Closed QA	2.6%
Extract	1.9%

Table 2: Illustrative prompts from our API prompt dataset. These are fictional examples inspired by real usage—see more examples in Appendix A.2.1.

Use-case	Prompt
Brainstorming	List five ideas for how to regain enthusiasm for my career
Generation	Write a short story where a bear goes to the beach, makes friends with a seal, and then returns home.
Rewrite	This is the summary of a Broadway play: """ {summary} """ This is the outline of the commercial for that play: """

3 – Learning from HF (RLHF)

3.1. SFT Model

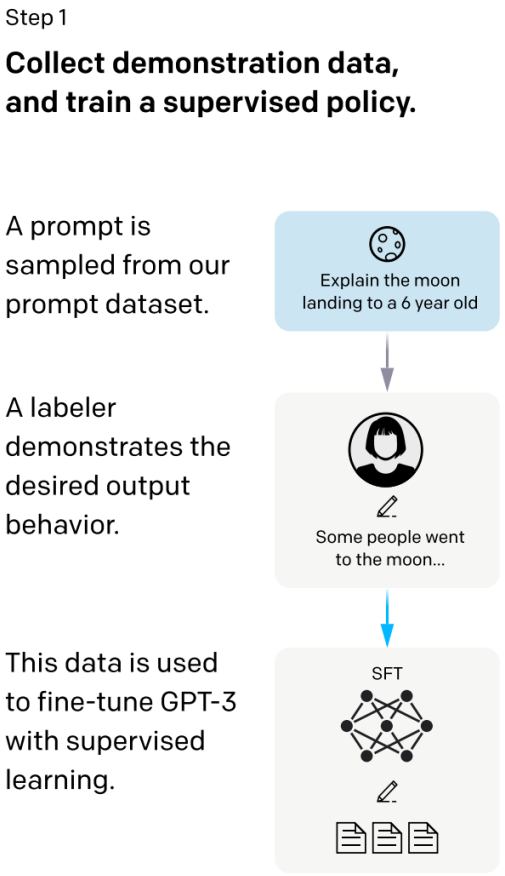


- A large collections of prompts:

SFT Data		
split	source	size
train	labeler	11,295
train	customer	1,430
valid	labeler	1,550
valid	customer	103

3 – Learning from HF (RLHF)

3.1. SFT Model



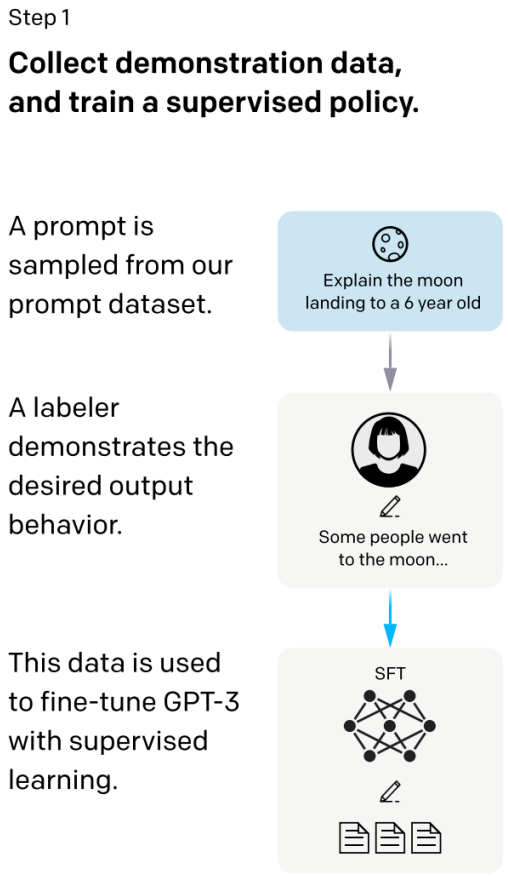
- Fine tune the model, call this model SFT Model
  - Initialized with pretrained GPT3 175B model
  - Trained for 16 epochs on demonstration data
  - Notation:

$$\pi^{SFT}$$

AI VIET NAM
@aivietnam.edu.vn

3 – Learning from HF (RLHF)

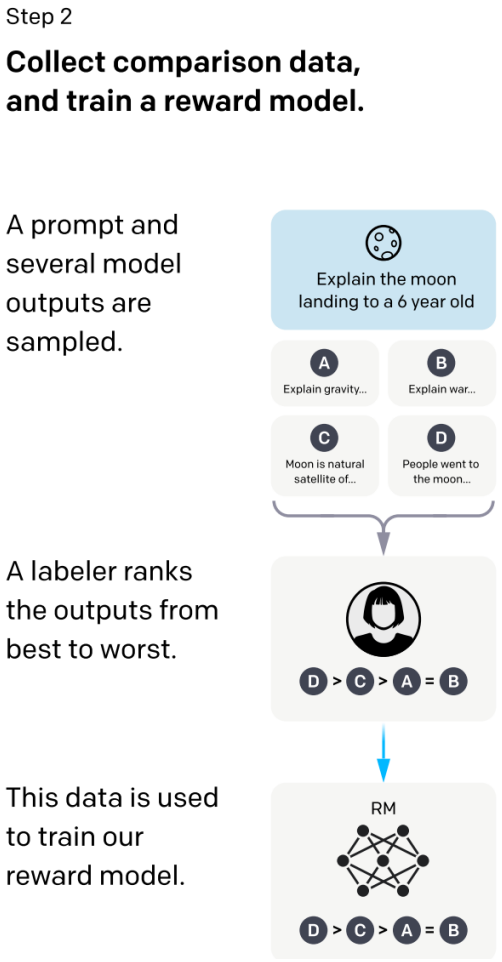
3.1. SFT Model



- Task: language modeling
- Training data: high-quality in the format of (prompt, response)
- Data scale: 10,000 – 100,000 (prompt, response) pairs
- Model: LLMs
  - Input: prompt
  - Output: response for this prompt
- Loss: cross entropy

3 – Learning from HF (RLHF)

3.2. Reward Modeling



- Training a model to output a score on a given input (a pair of prompt – response)
- A classification or regression task

RM Data		
split	source	size
train	labeler	6,623
train	customer	26,584
valid	labeler	3,488
valid	customer	14,399

AI VIET NAM
@aivietnam.edu.vn

3 – Learning from HF (RLHF)

3.2. Reward Modeling

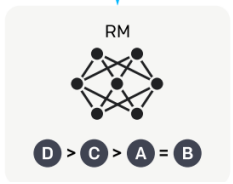
Step 2

Collect comparison data, and train a reward model.

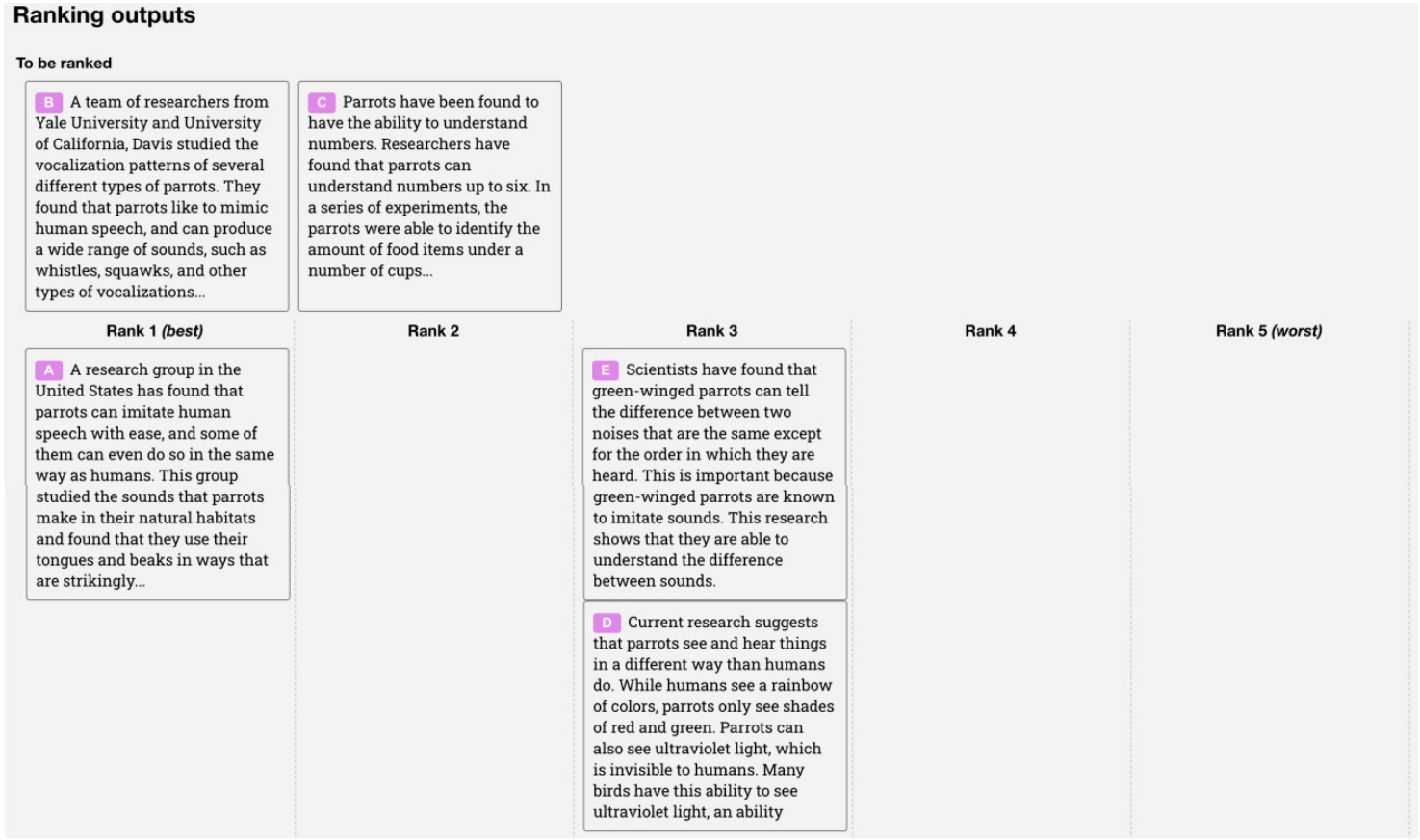
A prompt and several model outputs are sampled.

A labeler ranks the outputs from best to worst.

This data is used to train our reward model.



➤ Given K = 4 to 9 outputs to rank for each prompt



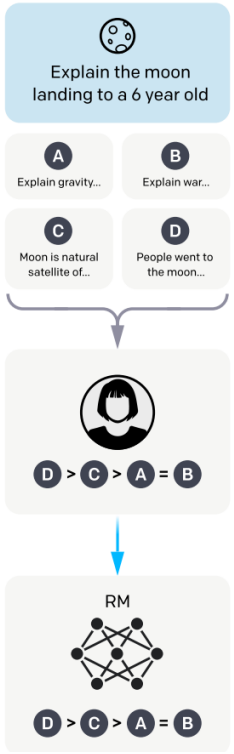
3 – Learning from HF (RLHF)

3.2. Reward Modeling

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.

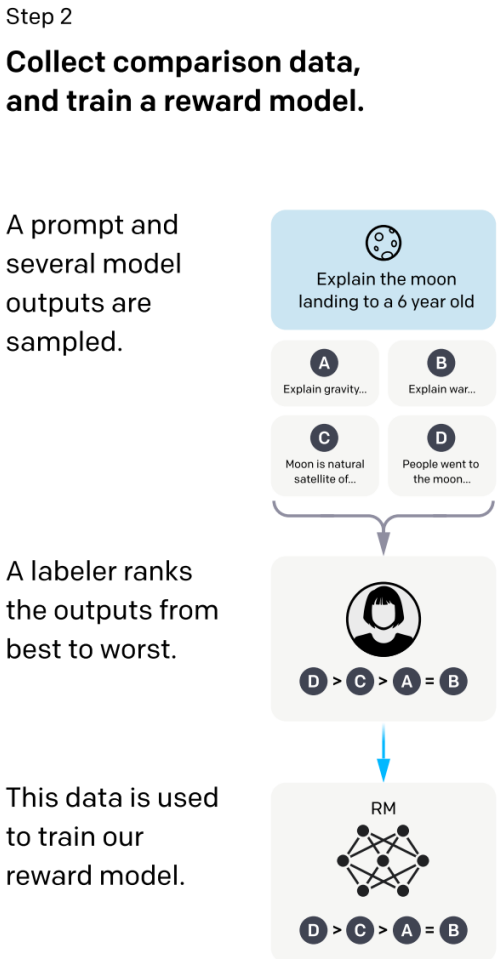
This data is used to train our reward model.

- Given K = 4 to 9 outputs to rank for each prompt
- For 4 ranked responses: D > C > A = B

=> 5 ranked pairs: (D > C), (D > A), (D > B), (C > A), (C > B)

3 – Learning from HF (RLHF)

3.2. Reward Modeling



- The reward model: r_θ

x: the prompt, y_w: the better completion, y_l: the worse completion

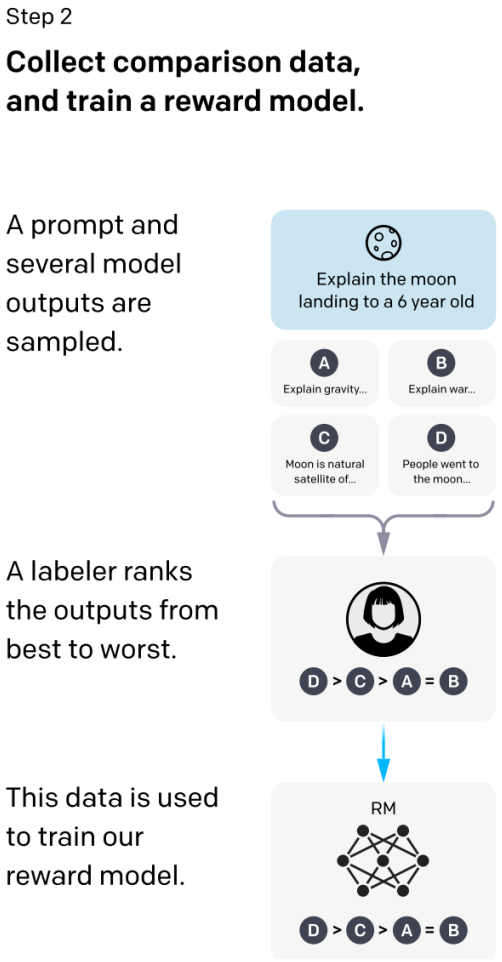
Reward on better completion

$$\text{loss}(\theta) = \mathrm{E}_{(x,y_w,y_l)\sim D}\left[\log\left(\sigma\left(r_\theta(x,y_w) - r_\theta(x,y_l)\right)\right)\right]$$

Reward on worse completion

3 – Learning from HF (RLHF)

3.2. Reward Modeling



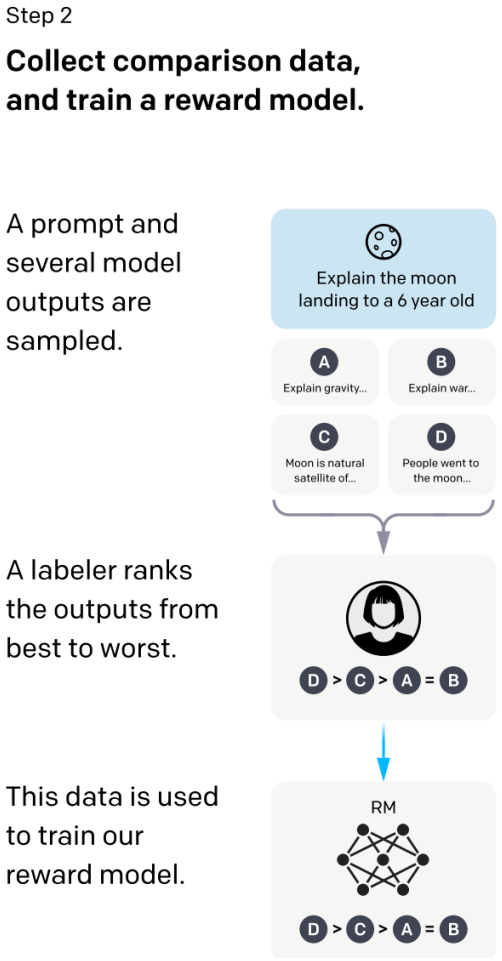
- The reward model: r_θ
- Overfitting problem

Each prompt has K completions => K choose 2 pairs to compare

Each completion can appear in K-1 gradient updates

3 – Learning from HF (RLHF)

3.2. Reward Modeling



- The reward model: r_θ
- Overfitting problem

Each prompt has K completions => K choose 2 pairs to compare

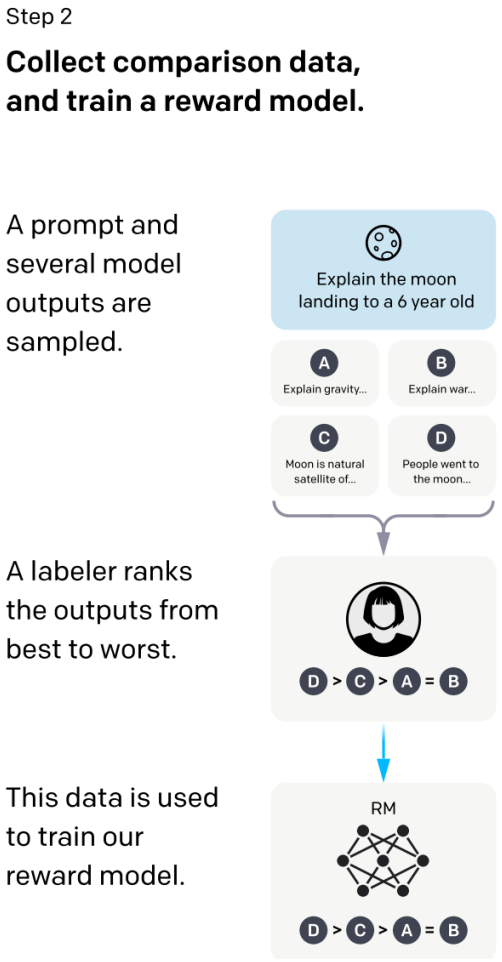
Each completion can appear in K-1 gradient updates

- Solution: train on all comparisons from each prompt as a single batch element
- Normalization in loss with -1/(K choose 2):

$$\text{loss}(\theta) = -\frac{1}{\binom{K}{2}} E_{(x,y_w,y_l)\sim D}\left[\log\left(\sigma\left(r_\theta(x,y_w) - r_\theta(x,y_l)\right)\right)\right]$$

3 – Learning from HF (RLHF)

3.2. Reward Modeling



- The reward model: r_θ
- Training data: high-quality data

x: the prompt, y_w: the better completion, y_l: the worse completion

- Data scale: 100K – 1M examples

InstructGPT: 50,000 prompts (each prompt: 4 to 9 responses) => 300K to 1.8M training examples

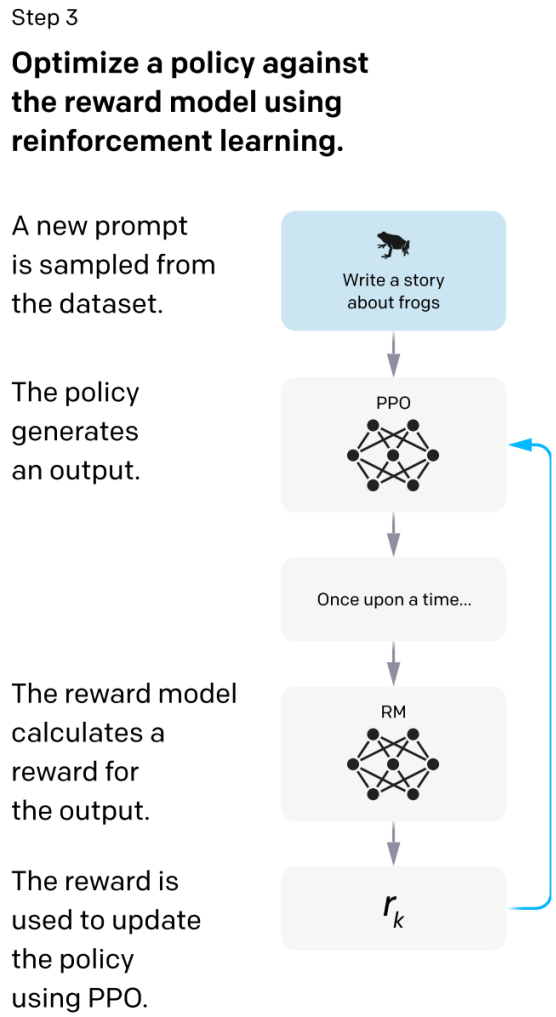
- Training sample (x, y_w, y_l)

$$\text{loss}(\theta) = -\frac{1}{\binom{K}{2}} E_{(x,y_w,y_l)\sim D}\left[\log\left(\sigma\left(r_\theta(x, y_w) - r_\theta(x, y_l)\right)\right)\right]$$

AI VIET NAM
@aivietnam.edu.vn

3 – Learning from HF (RLHF)

3.3. Reinforcement learning (RL)



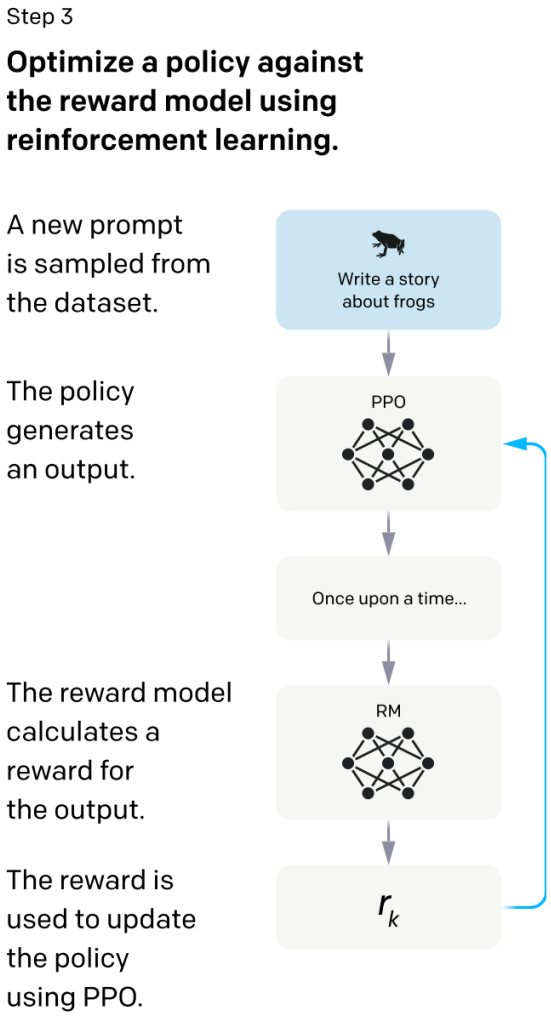
- Goal: train the SFT model to generate output responses that will maximize the scores by the RM model
- Training data: randomly selected prompts
- Data sacle: 10,000 – 100,0000 prompts

PPO Data

split	source	size
train	customer	31,144
valid	customer	16,185

3 – Learning from HF (RLHF)

3.3. Reinforcement learning (RL)

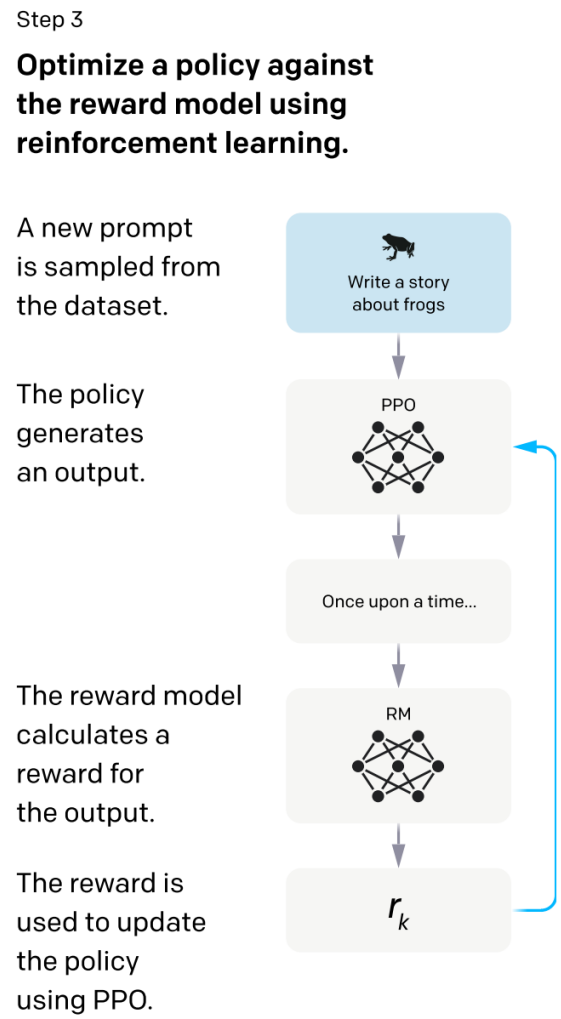


ML Task: Reinforcement Learning

- Action space: the vocabulary of tokens the LLM uses. Taking action means choosing a token to generate
- Observation space: the distribution over all possible prompts
- Policy: the probability distribution over all actions to take (all tokens to generate) given an observation (a prompt)
- Reward function: the reward model from stage 2

3 – Learning from HF (RLHF)

3.3. Reinforcement learning (RL)



- D_{RL}: the distribution of prompts used for RL model
- LLM π_ϕ^{RL}: the model being trained with RL, parameterized by ϕ
- For each x from D_{RL}: y: $\pi_\phi^{RL}(x)$

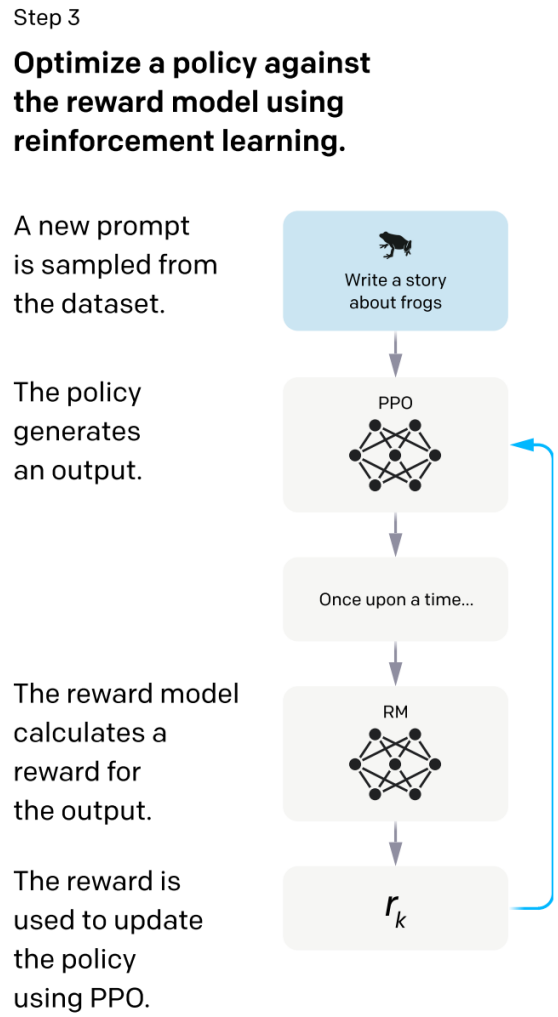
$$\text{objective}_1(x, y; \phi) = \boldsymbol{r_\theta}(x, y)$$

- For all training data DRL

$$\text{objective}_1(x, y; \phi) = E_{(x,y)\sim D_{\pi_\phi}RL}\boldsymbol{r_\theta}(x, y)$$

3 – Learning from HF (RLHF)

3.3. Reinforcement learning (RL)

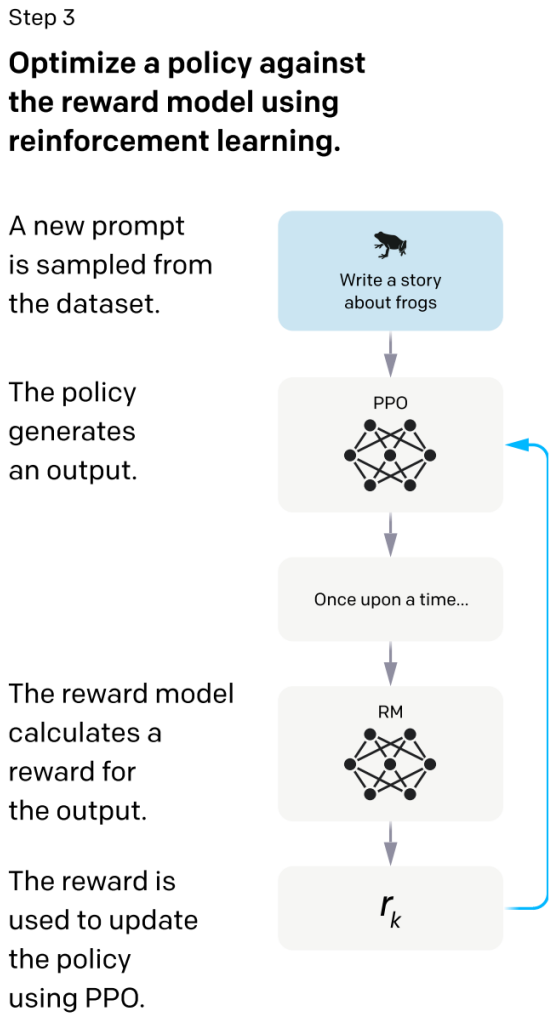


- **Worse reward esimates**: as RLHF is updated, its outputs become very different from what the RM was trained on
- Solution: add a KL penalty that makes sure PPO model output does not deviate too far from SFT model

$$\text{objective}_1(\text{x}, \text{y}; \phi) = \text{E}_{(x,y)\sim \text{D}_{\pi_\phi} RL}\left[\boldsymbol{r_\theta}(\text{x}, \text{y}) - \beta \log \frac{\pi_\phi^{\text{RL}}(y|x)}{\pi^{\text{SFT}}(y|x)}\right]$$

3 – Learning from HF (RLHF)

3.3. Reinforcement learning (RL)

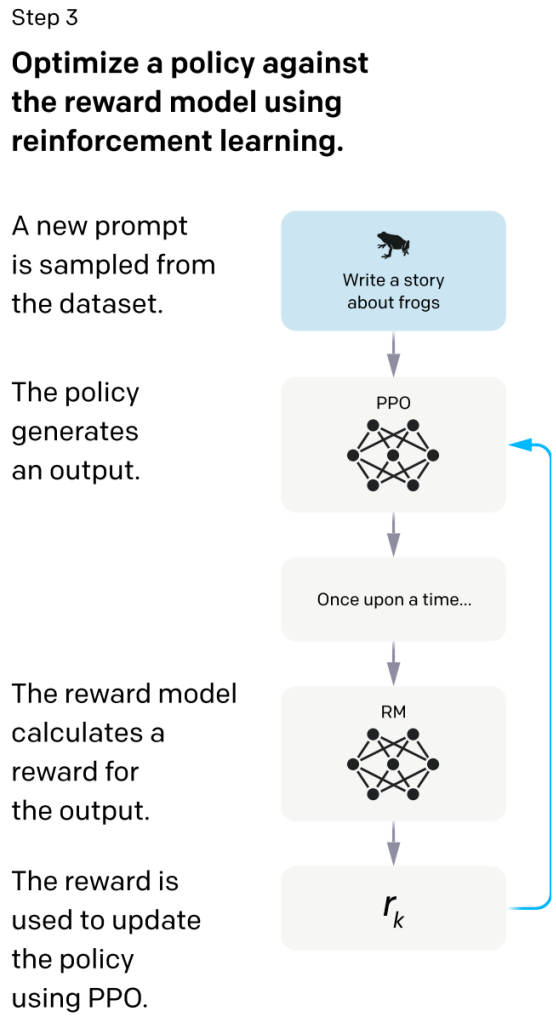


- **Just using RL objective leads to performance degradation on many NLP tasks**
- Solution: add a auxiliary LM ojective on the pretraining data. Call this variant PPO-ptx
- $D_{pretrain}$: the distribution of the pretraining data for the pretrain model

$$\text{objective}_2(x_{pretrain}; \phi) = \gamma E_{x \sim D_{pretrain}} \left[\log \left(\pi_{\phi}^{RL}(x) \right) \right]$$

3 – Learning from HF (RLHF)

3.3. Reinforcement learning (RL)



- **Maximize the objective function in RL training:**

$$\text{objective}(\boldsymbol{\phi}) = \text{objective}_1(\text{x}, \text{y}; \phi) + \text{objective}_2\left(\text{x}_{\text{pretrain}}; \phi\right)$$

$$\text{objective}(\boldsymbol{\phi}) = \text{E}_{(x,y)\sim \text{D}_{\pi_\phi} RL}\left[\boldsymbol{r_\theta}(\text{x}, \text{y}) - \beta \log \frac{\pi_\phi^{\text{RL}}(y|x)}{\pi^{\text{SFT}}(y|x)}\right] + \gamma \text{E}_{\text{x}\sim \text{D}_{\text{pretrain}}}\left[\log\left(\boldsymbol{\pi_\phi^{\text{RL}}}(\text{x})\right)\right]$$

AI VIET NAM
@aivietnam.edu.vn

3 – Learning from HF (RLHF)

Summary

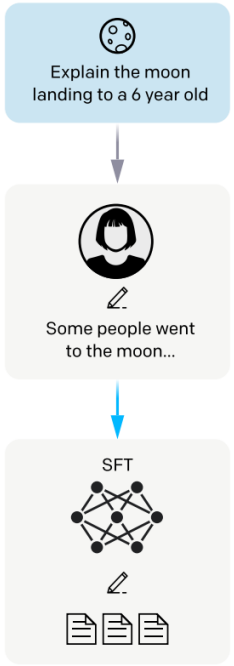
Step 1

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 with supervised learning.



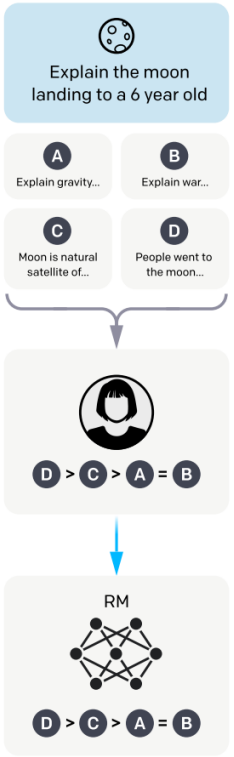
Step 2

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

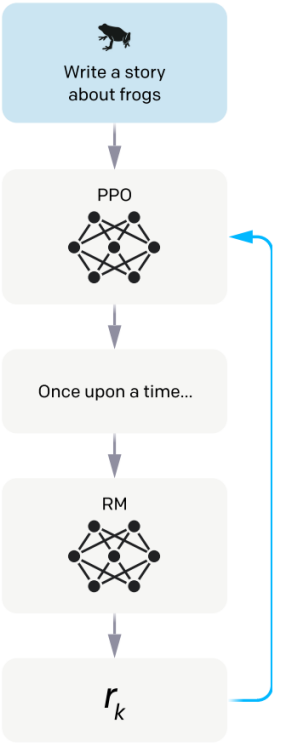
Optimize a policy against the reward model using reinforcement learning.

A new prompt is sampled from the dataset.

The policy generates an output.

The reward model calculates a reward for the output.

The reward is used to update the policy using PPO.



AI VIET NAM
@aivietnam.edu.vn

4 - Experiment

Source code: https://github.com/thainq107/text_classification.git

Reference

- **COS 597G (Fall 2022): Understanding Large Language Models**
- RLHF: Reinforcement Learning from Human Feedback, HuyenChip
- https://github.com/ethanyanjiali/minChatGPT
- https://lilianweng.github.io/posts/2023-03-15-prompt-engineering/
- https://learnprompting.org/

AI VIET NAM

@aivietnam.edu.vn

Thanks!

Any questions?