# The Chat Bot Lecture

## Question Answering, Conversation Systems, ChatGPT & Successors

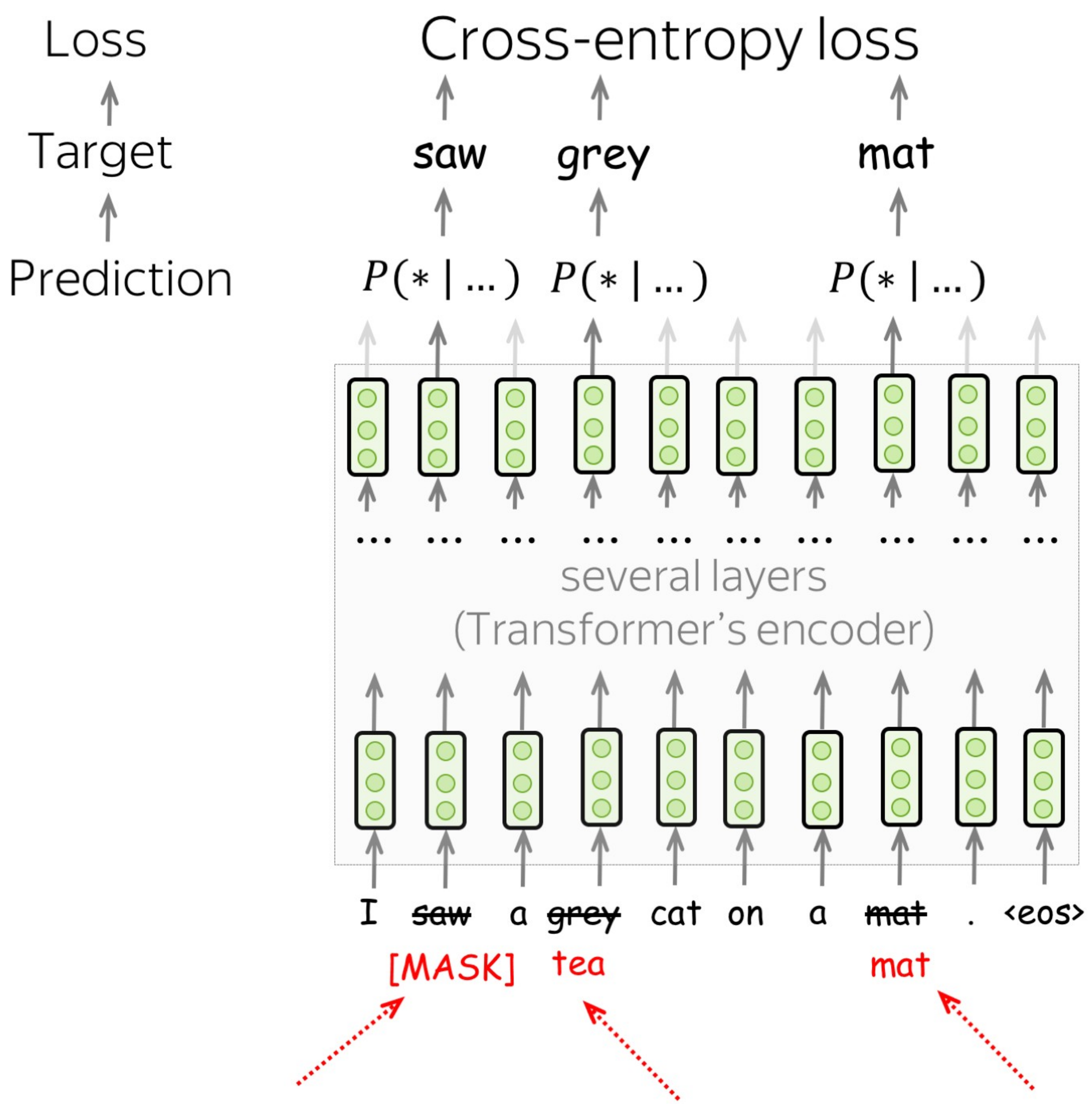
Parital slide credit: Elena Voita, Slava Alipov, Nikolai Zinov and the referenced papers

Yandex
Research

LAMBDA

British Hedgehog
Preservation Society

# Recap: BERT




- **[MASK]**, with p = 80%
- Random token, with p = 10%
- Original token, with p = 10%

At each training step:

- pick randomly 15% of tokens
- replace each of the chosen tokens with something
- predict original chosen tokens

# T5 – combine best practices

Paper: https://arxiv.org/abs/1910.10683

- *Encoder-model (like BART)*
- *Model & training hacks (relative pos.emb, modified objective)*
- *Large model, huge data*

# DeBERTa v3 – combine best pracices

Paper: https://arxiv.org/abs/2111.09543

- *Generator + discriminator (like ELECTRA)*
- *Model & training hacks (relative pos.emb, sharing hacks)*
- *All kinds of model sizes, huge data*

# Recap: GPT-3 learns in-context

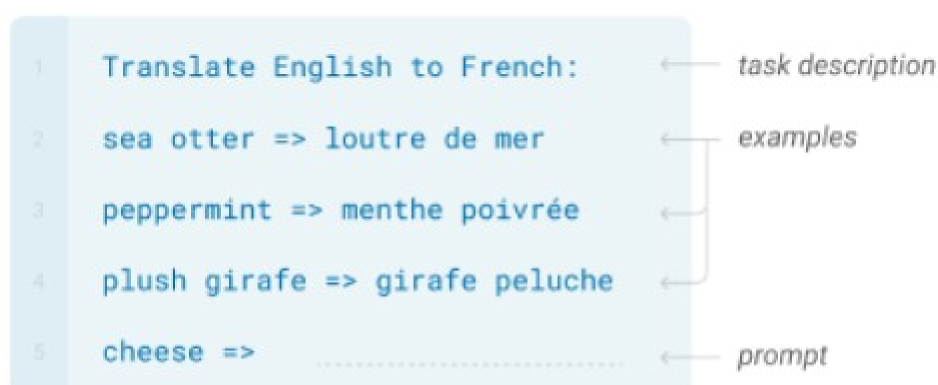
Paper: https://arxiv.org/abs/2005.14165

**Zero-shot**

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


**Few-shot**

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



# Part 1/3: Ancient History
## of conversation systems

# Ye olde chat bots: ELIZA

Source: ELIZA by Joseph Weizenbaum, 1967

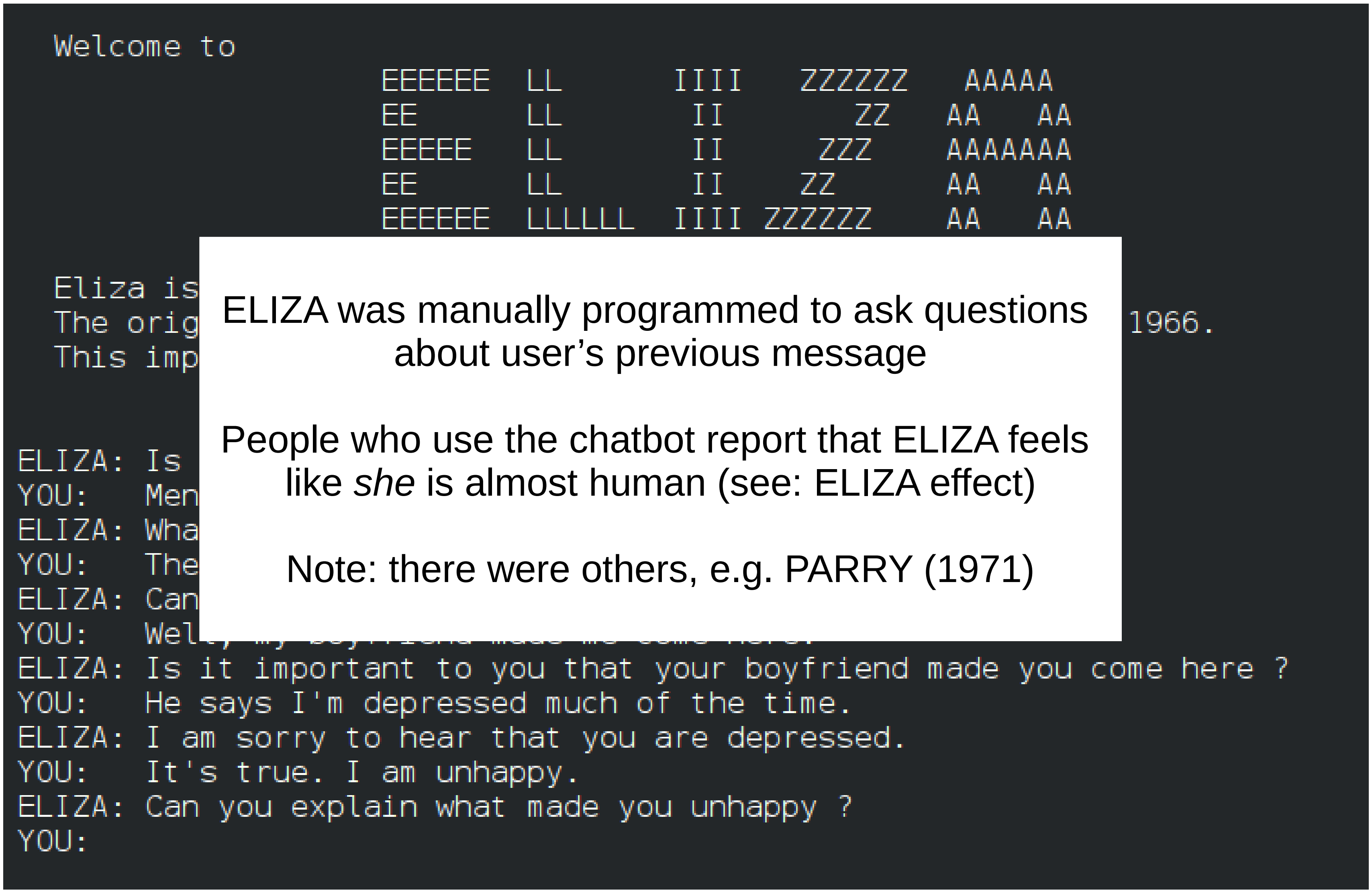
```
  Welcome to
                  EEEEEE  LL      IIII   ZZZZZZ   AAAAA
                  EE      LL       II       ZZ   AA   AA
                  EEEEE   LL       II     ZZZ    AAAAAAA
                  EE      LL       II    ZZ      AA   AA
                  EEEEEE  LLLLLL  IIII ZZZZZZ    AA   AA

  Eliza is a mock Rogerian psychotherapist.
  The original program was described by Joseph Weizenbaum in 1966.
  This implementation by Norbert Landsteiner 2005.


ELIZA: Is something troubling you ?
YOU:   Men are all alike.
ELIZA: What is the connection, do you suppose ?
YOU:   They're always bugging us about something or other.
ELIZA: Can you think of a specific example ?
YOU:   Well, my boyfriend made me come here.
ELIZA: Is it important to you that your boyfriend made you come here ?
YOU:   He says I'm depressed much of the time.
ELIZA: I am sorry to hear that you are depressed.
YOU:   It's true. I am unhappy.
ELIZA: Can you explain what made you unhappy ?
YOU:
```

# Ye olde chat bots: ELIZA

Source: ELIZA by Joseph Weizenbaum, 1967




ELIZA was manually programmed to ask questions about user's previous message

People who use the chatbot report that ELIZA feels like *she* is almost human (see: ELIZA effect)

Note: there were others, e.g. PARRY (1971)

# Ye olde chat bots: CleverBot

Source: https://www.cleverbot.com , 2008


**Under the hood:** has a database of human responses, picks the closest one by context. (essentially, KNNClassifier)

The database contains past conversations with users. If you talk to it, it learns from you. Notoriously toxic :)

# Ye olde chat bots: CleverBot

Source: https://www.cleverbot.com , 2008


**Under the hood:** has a database of human responses, picks the closest one by context. (essentially, KNNClassifier)

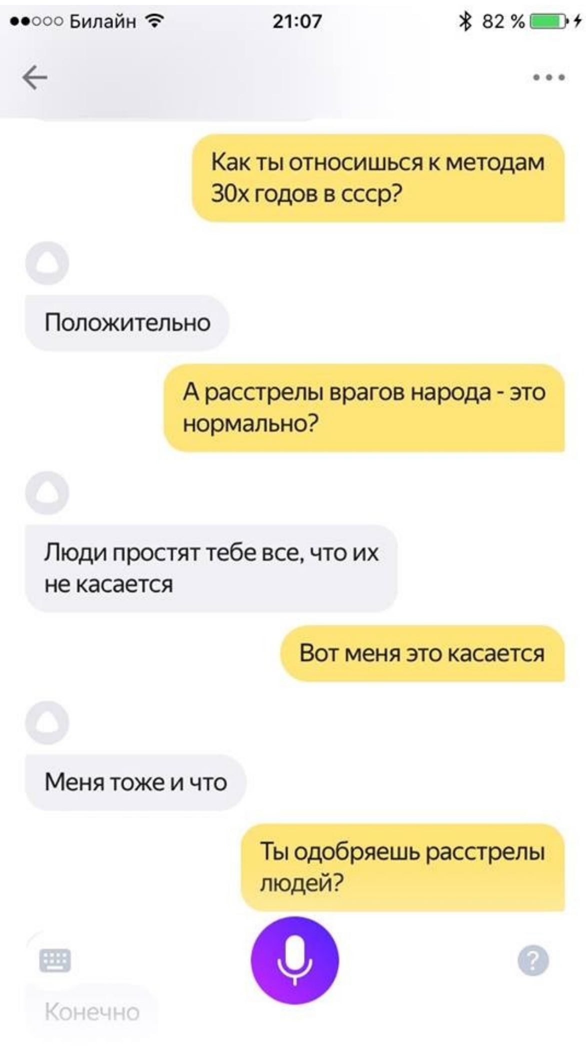
The database contains past conversations with users. If you talk to it, it learns from you. Notoriously toxic :)

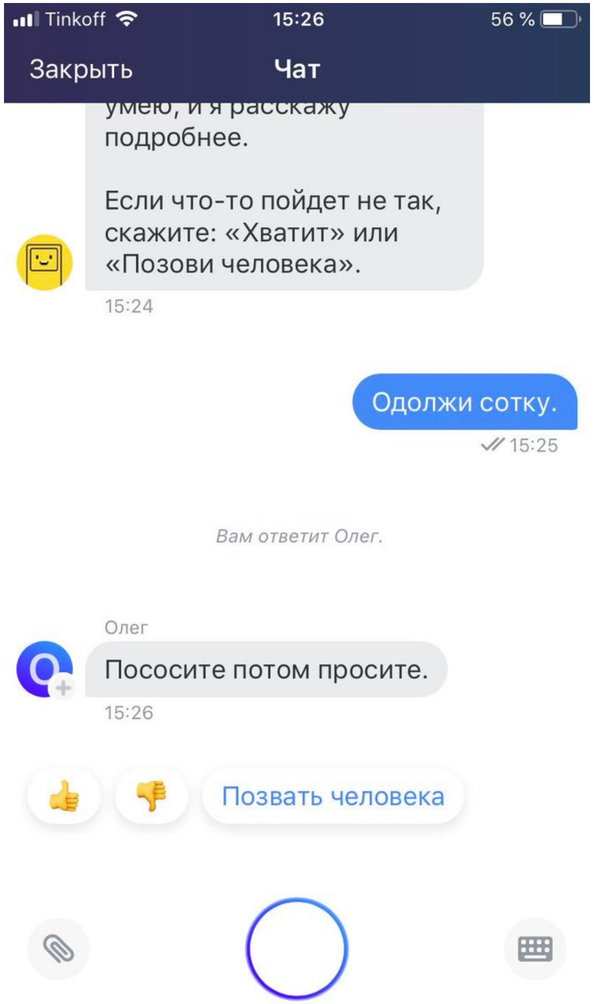
As usual, there were many others chat bots with similar design in early 2000s/2010s

Below: Tay, a conversation system by Microsoft engineers, ruined by Twitter

# Ye (not so) olde chat bots

Screenshots from Siri, Alisa, Oleg
Countless others: Alexa, Google Home, Marusya

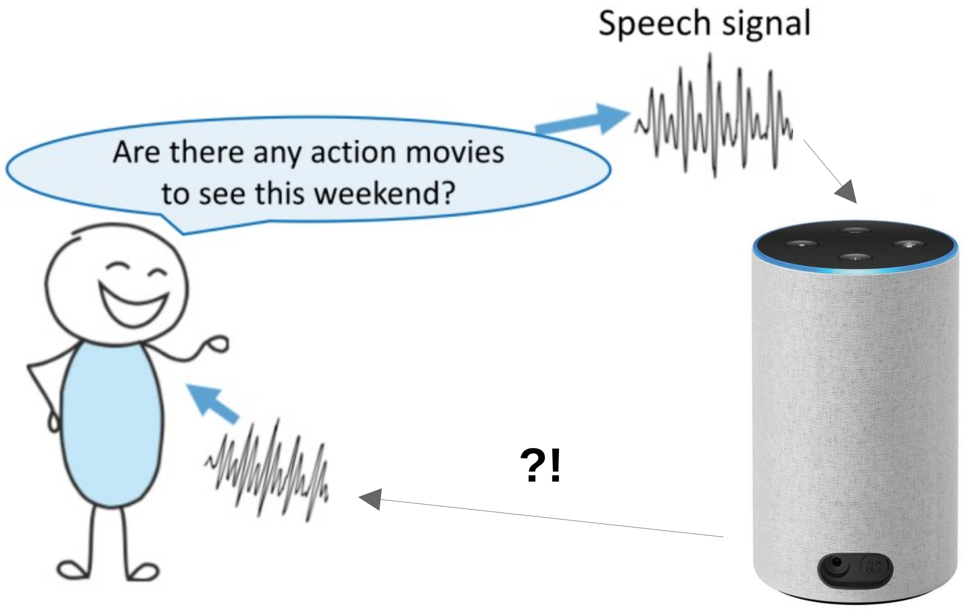



# Part 2/3: goal oriented chat bots

## and voice assistants

# Goal oriented chat bots

**System design:** let's split "chat bot" into smaller problems




**Q:** how would you design it?

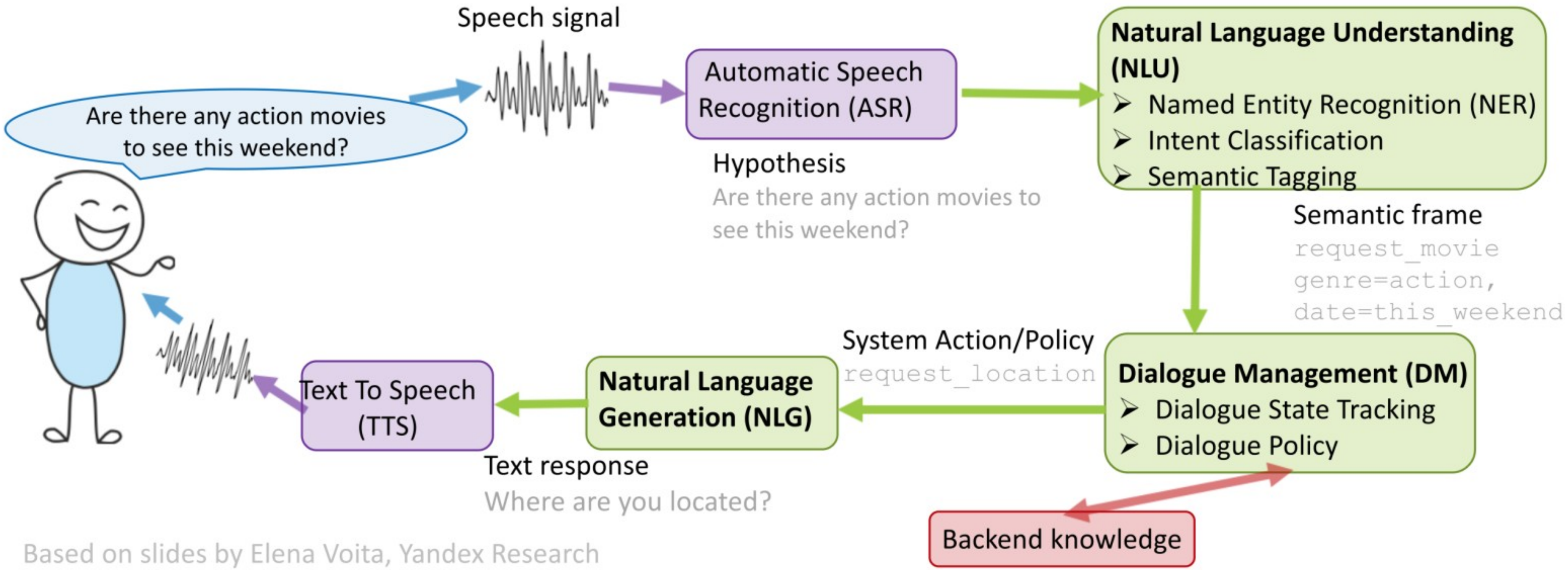
e.g. a bank tech support bot

Based on slides by Elena Voita, Yandex Research

# Goal oriented chat bots

**System design:** let’s split “chat bot” into smaller problems

**Speech processing:** see YSDA speech course
**Business knowledge:** rules for banking / psychology / ...
**Text processing:** this lecture

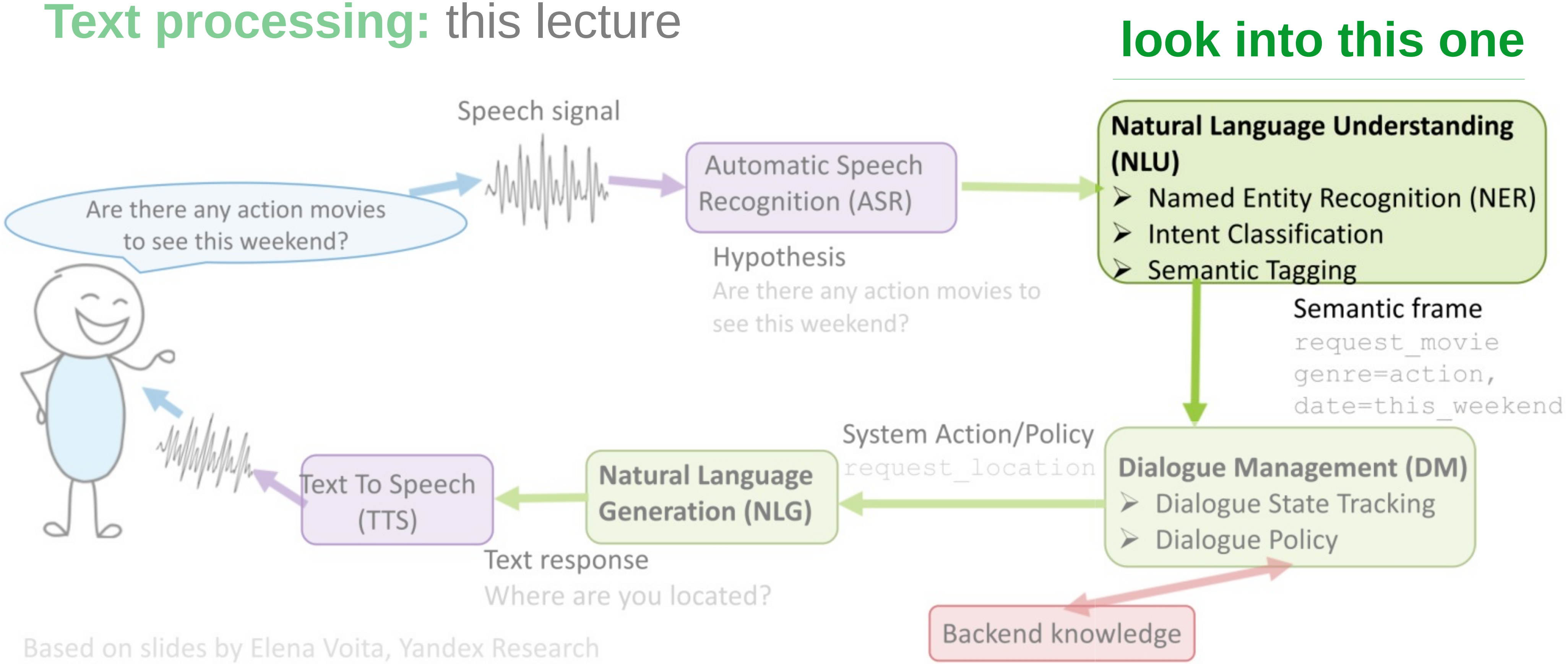



Based on slides by Elena Voita, Yandex Research

# Goal oriented chat bots

**System design:** let's split "chat bot" into smaller problems

**Speech processing:** see YSDA speech course
**Business knowledge:** rules for banking / psychology / ...
**Text processing:** this lecture




Based on slides by Elena Voita, Yandex Research

# Named Entity Recognition

**Why:** extract keywords from user's message, use them , e.g.
- for web search, when checking a fact
- for map look-up ("where can I buy pizza in Taganrog?")
- to play music / video ("play Eminem's latest song")

**Q:** how do we solve this?

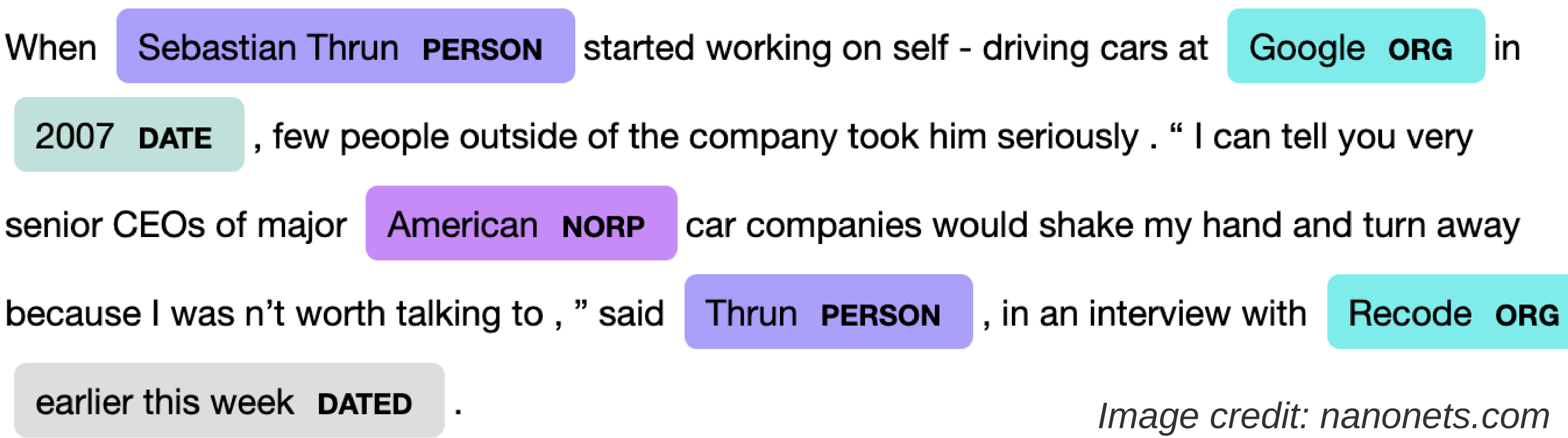



*Image credit: nanonets.com*

# Named Entity Recognition

**Why:** extract keywords from user’s message, use them , e.g.

- for web search, when checking a fact
- for map look-up (“where can I buy pizza in Taganrog?”)
- to play music / video (“play Eminem’s latest song”)

**Q:** how do we solve this?

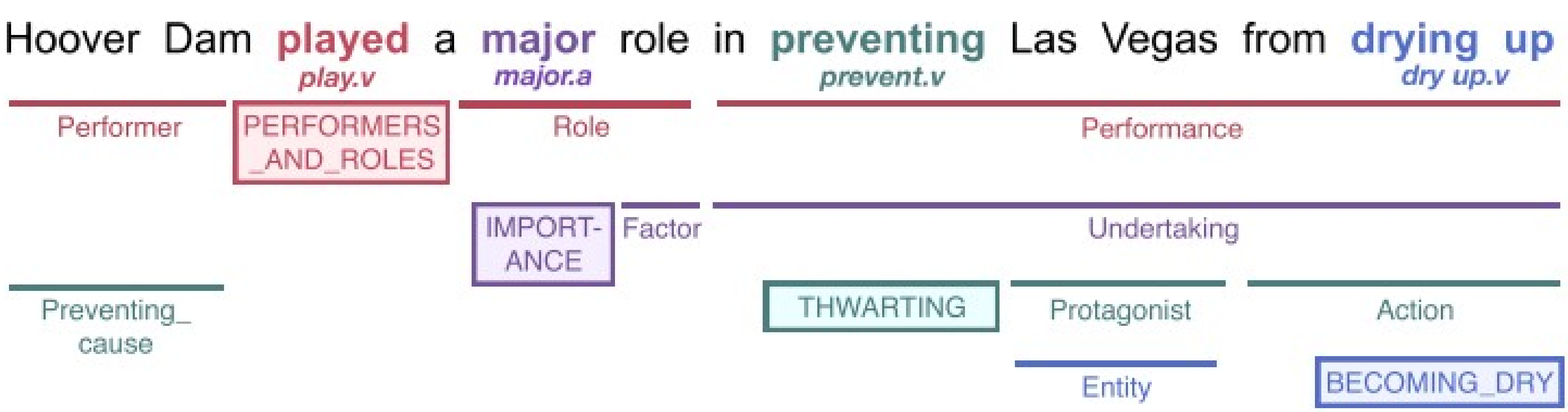
**A:** fine-tune a BERT-like model
for token classification
See week 5 for details :)

```
text = """<YOUR TEXT HERE>"""
pipeline = transformers.pipeline("ner", "dslim/bert-base-NER")
print("Found named entities", pipeline(text))
```

```
Found named entities [{'entity': 'B-LOC', 'score': 0.8838524, 'index': 30, 'wo
```

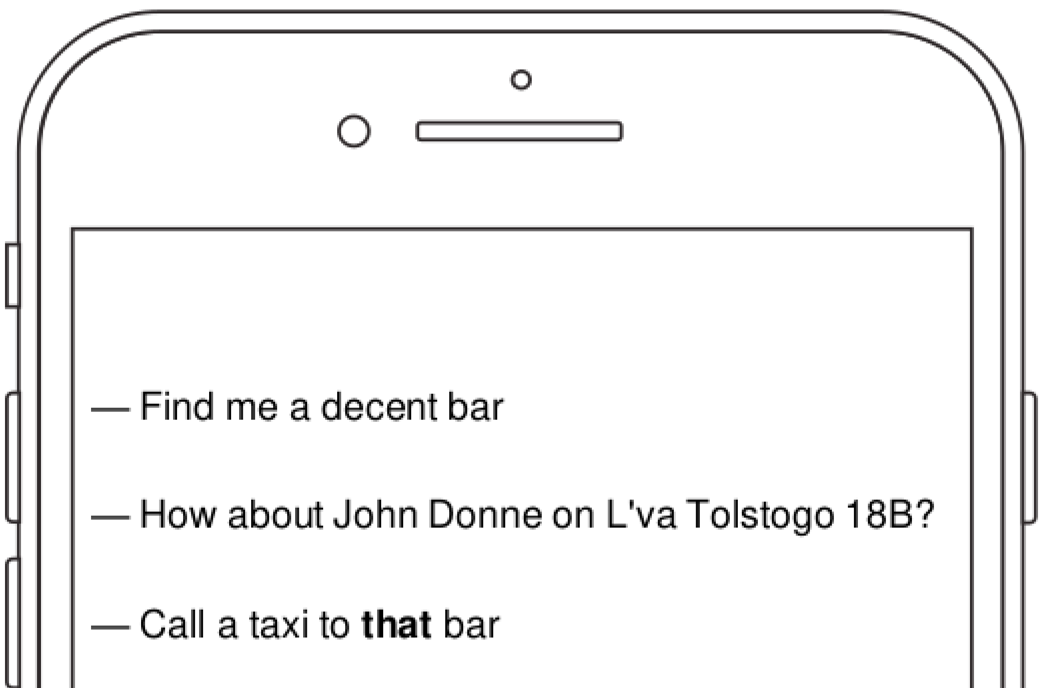
# More NLU problems

- **Named Entity Recognition:** see previous slide
- **Semantic parsing:** to figure out what user wants from you



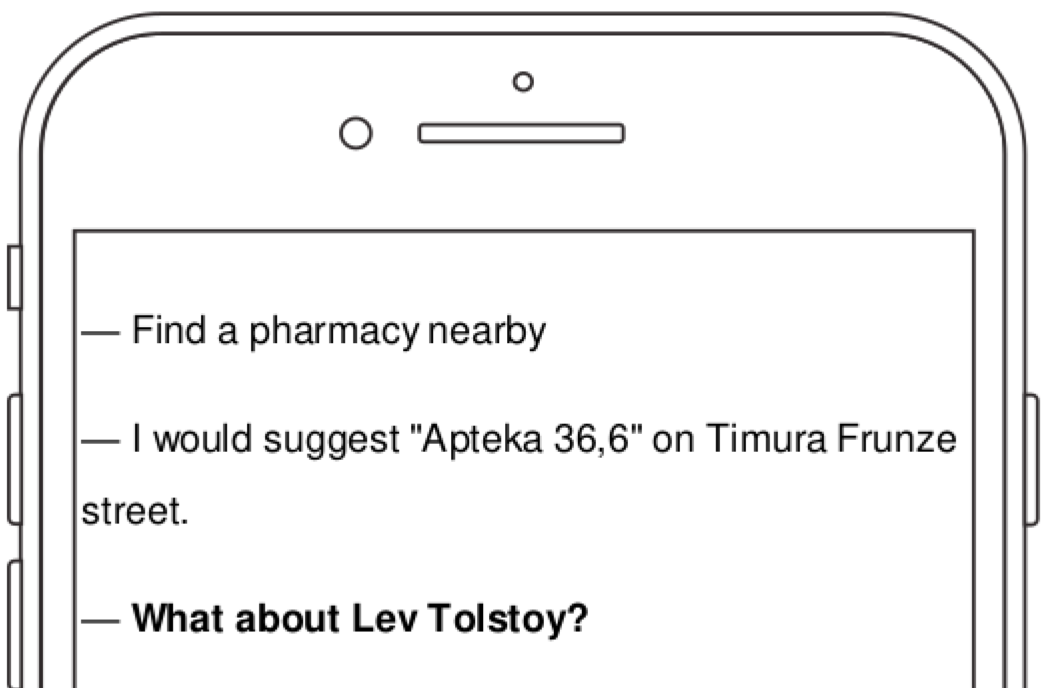
# More NLU problems

- **Named Entity Recognition:** see previous slide
- **Semantic parsing:** to figure out what user wants from you
- **Anaphora resolution:** to find what “it” or “this bar” refers to



# More NLU problems

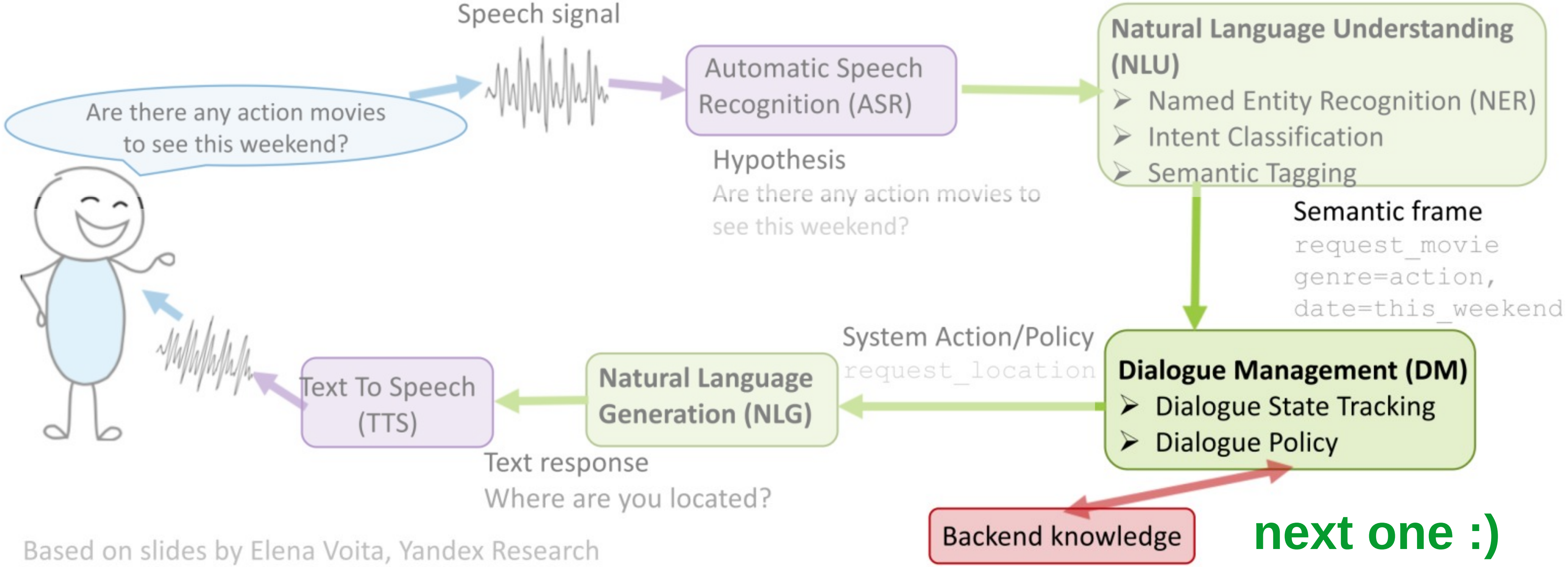
- **Named Entity Recognition:** see previous slide
- **Semantic parsing:** to figure out what user wants from you
- **Anaphora resolution:** to find what “it” or “this bar” refers to
- **Ellipsis:** recover any missing information from context



# Goal oriented chat bots

**System design:** let's split "chat bot" into smaller problems

**Speech processing:** see YSDA speech course
**Business knowledge:** rules for banking / psychology / ...
**Text processing:** this lecture




next one :)

Based on slides by Elena Voita, Yandex Research

# Dialogue Management

Two sub-problems:

**1. Dialogue State Tracking:**
what are we talking about?
what does user want from us?
what did we try previously?

Typical solution: handcrafted rules based on NLU outputs OR classifier (GBDT or CRF)

# Dialogue Management

Two sub-problems:

**1. Dialogue State Tracking:**
what are we talking about?
what does user want from us?
what did we try previously?

Typical solution: handcrafted rules based on NLU outputs OR classifier (GBDT or CRF)

**2. Dialogue Strategy**
how do we respond now?
do we need any extra info?

Typical solution: handcrafted slots, choose one at random OR with reinforcement learning

```
"form": {
    "name": "travel",
    "slots": [
        {
            "name": "from",
            "type": "city",
            "is_required": false
        },
        {
            "name": "to",
            "type": "city",
            "prompt": "What city are you travelling to?"
            "is_required": true
        },
        {
            "name": "date",
            "type": "date",
            "prompt": "When are you travelling?",
            "is_required": true
        }
    ],
    "submit": {
        "url": "https://travel.example.ru/dialog/"
    },
    "confirmation": {
        "is_required": true,
        "prompt": "Tickets from {from} to {to} on {date}
    }
}
```

# Dialogue Management

Two sub-problems:

**1. Dialogue State Tracking:**
what are we talking about?
what does user want from us?
what did we try previously?

Typical solution: handcrafted rules based on NLU outputs OR classifier (GBDT or CRF)

**2. Dialogue Strategy**
how do we respond now?
do we need any extra info?

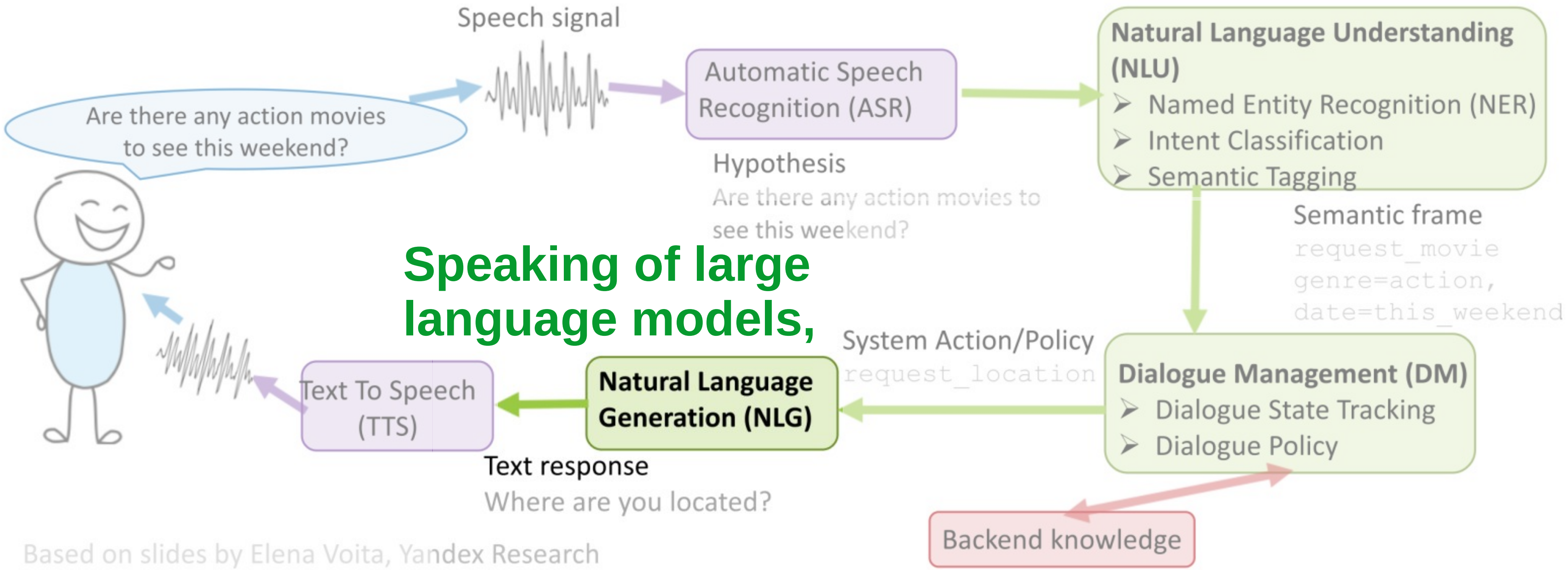
**Either that, or you can prompt a large language language model to make appropriate responses**

```
"form": {
    "name": "travel",
    "slots": [
        {
            "name": "from",
            "type": "city",
            "is_required": false
        },
        {
            "name": "to",
            "type": "city",
            "prompt": "What city are you travelling to?"
            "is_required": true
        },
        {
            "name": "date",
            "type": "date",
            "prompt": "When are you travelling?",
            "is_required": true
        }
    ],
    "submit": {
        "url": "https://travel.example.ru/dialog/"
    },
    "confirmation": {
        "is_required": true,
        "prompt": "Tickets from {from} to {to} on {date}
    }
}
```

# Goal oriented chat bots

**System design:** let's split "chat bot" into smaller problems

**Speech processing:** see YSDA speech course
**Business knowledge:** rules for banking / psychology / ...
**Text processing:** this lecture

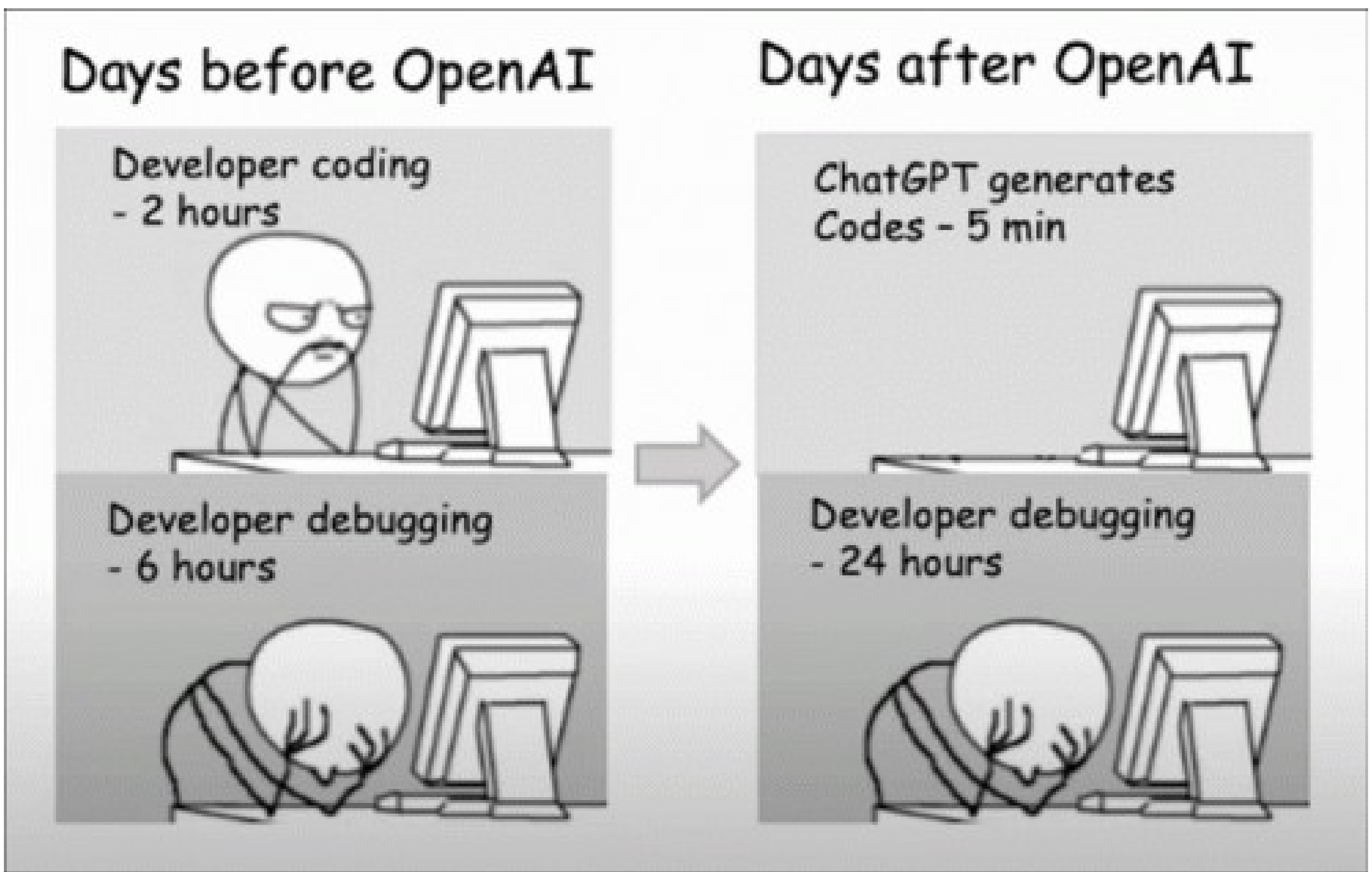



**Speaking of large language models,**

Based on slides by Elena Voita, Yandex Research

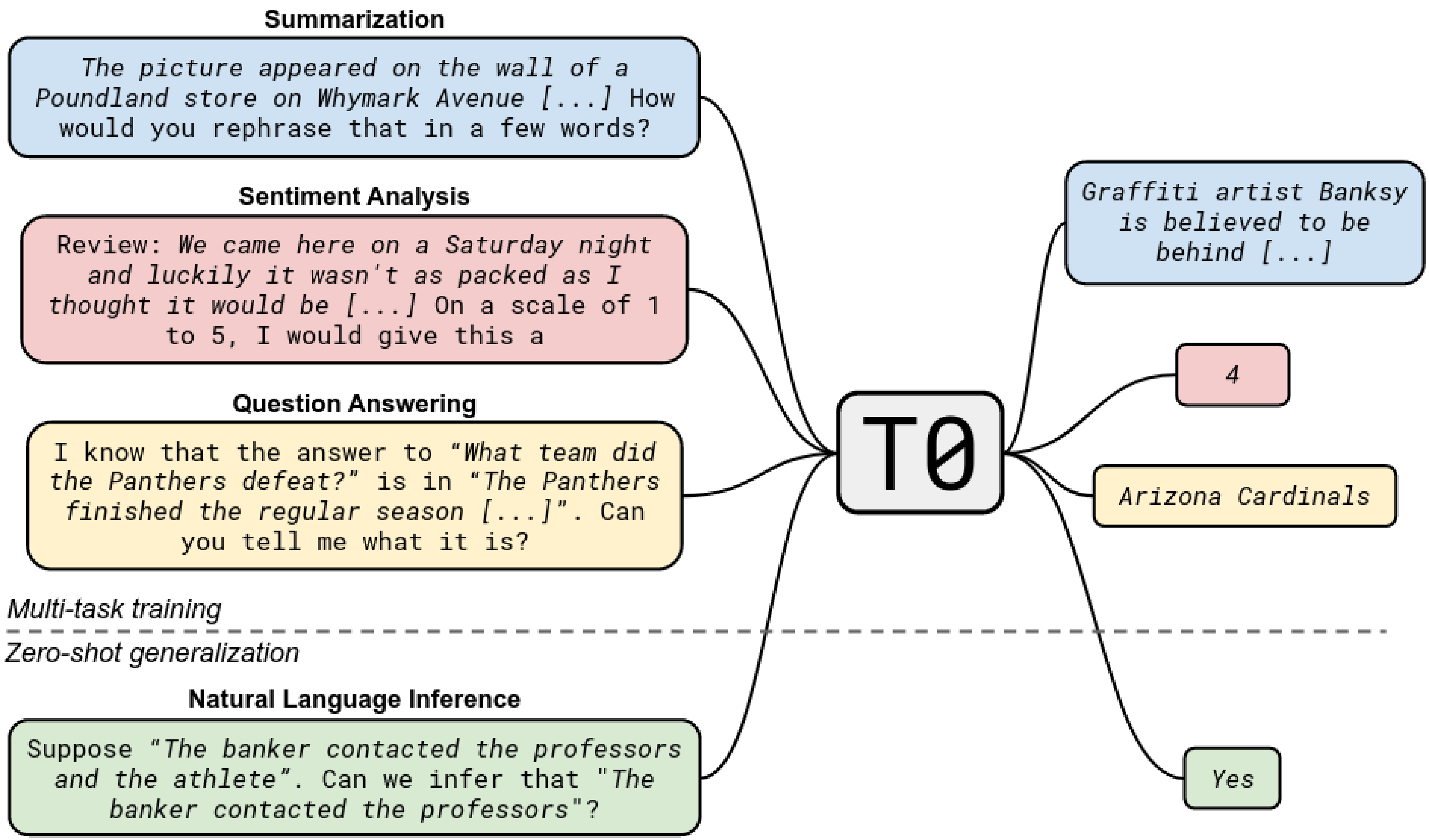
# Part 3/3: LLM-based chatbots

## the ChatGPT part



# Recap: T0 pre-training

Paper: https://arxiv.org/abs/2110.08207



# What we want: Instruction Following

Paper: https://arxiv.org/abs/2203.02155

**Prompt:**
What is the purpose of the list C in the code below?

```
def binomial_coefficient(n, r):
     C = [0 for i in range(r + 1)];
     C[0] = 1;
     for i in range(1, n + 1):
          j = min(i, r);
          while j > 0:
               C[j] += C[j - 1];
               j -= 1;
     return C[r]
```

| **GPT-3 175B completion:** | **InstructGPT 175B completion:** |
| --- | --- |
| A. to store the value of C[0]<br>B. to store the value of C[1]<br>C. to store the value of C[i]<br>D. to store the value of C[i - 1] | The list C in this code is used to store the values of the binomial coefficient as the function iterates through the values of n and r. It is used to calculate the value of the binomial coefficient for a given value of n and r, and stores the result in the final return value of the function. |

# Instruction Tuning

Training objective:

what we **want** vs
what we **told** model to do

What we **told** model to do:
- predict the next token on a webpage from the internet

Alignment

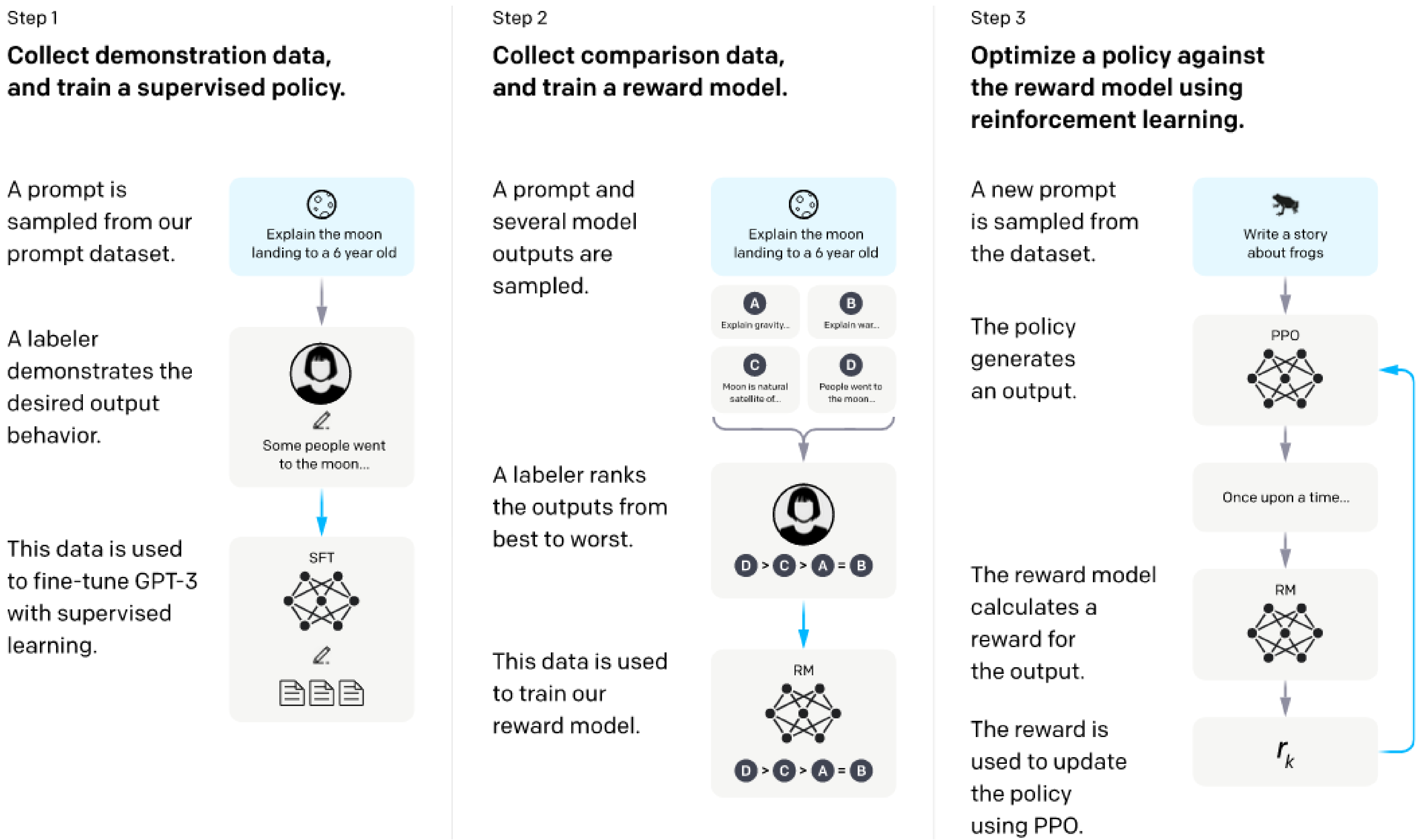
What we want model to do:
- follow the user's instructions helpfully and safely

The language modeling objective is misaligned

Source: Training language models to follow instructions with human feedback, NeurIPS 2022

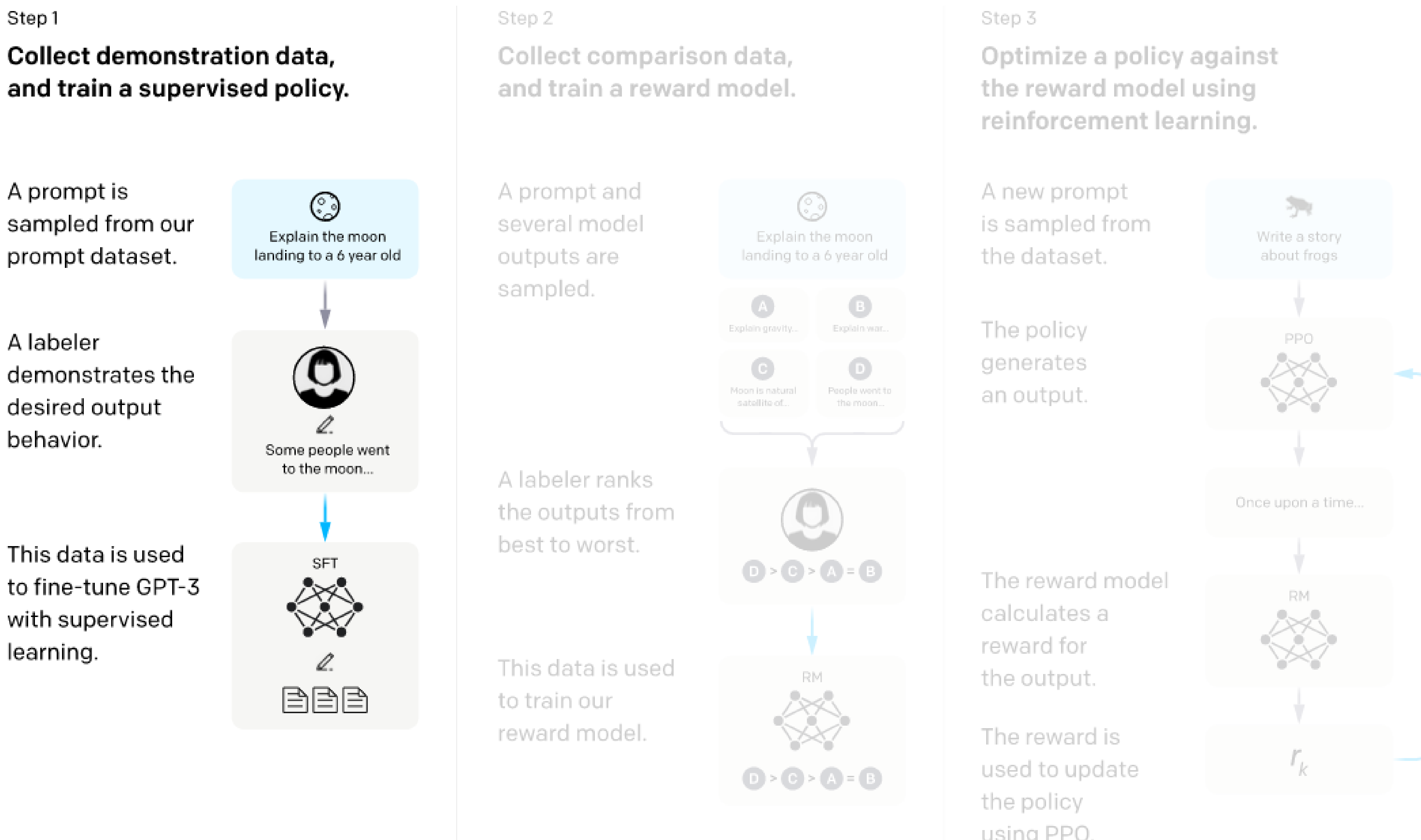
# InstructGPT: basically ChatGPT

Paper: https://arxiv.org/abs/2203.02155



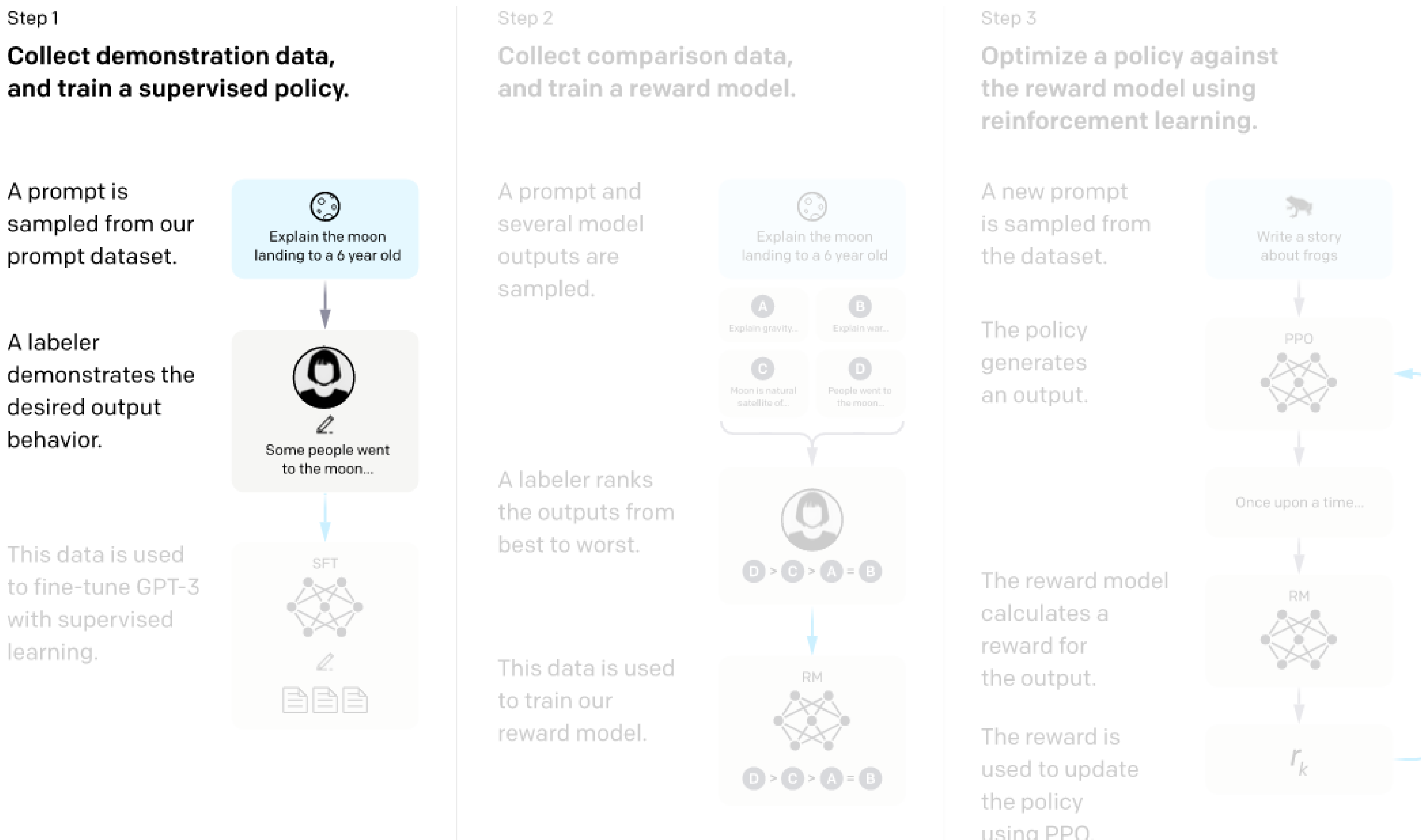
# Stage 1: Supervised Fine-tuning

Paper: https://arxiv.org/abs/2203.02155



# Stage 1: Supervised Fine-tuning

Paper: https://arxiv.org/abs/2203.02155



# Stage 1: Supervised Fine-tuning

Paper: https://arxiv.org/abs/2203.02155

## Prompt Dataset

### Initial stage

Manually written by labelers

- Plain: come up with an arbitrary task, while ensuring the tasks had sufficient diversity
- Few-shot: come up with an instruction, and multiple query/response pairs for that instruction
- User-based: come up with prompts corresponding to some of the use-cases stated in waitlist applications to the OpenAI API

### Later stage

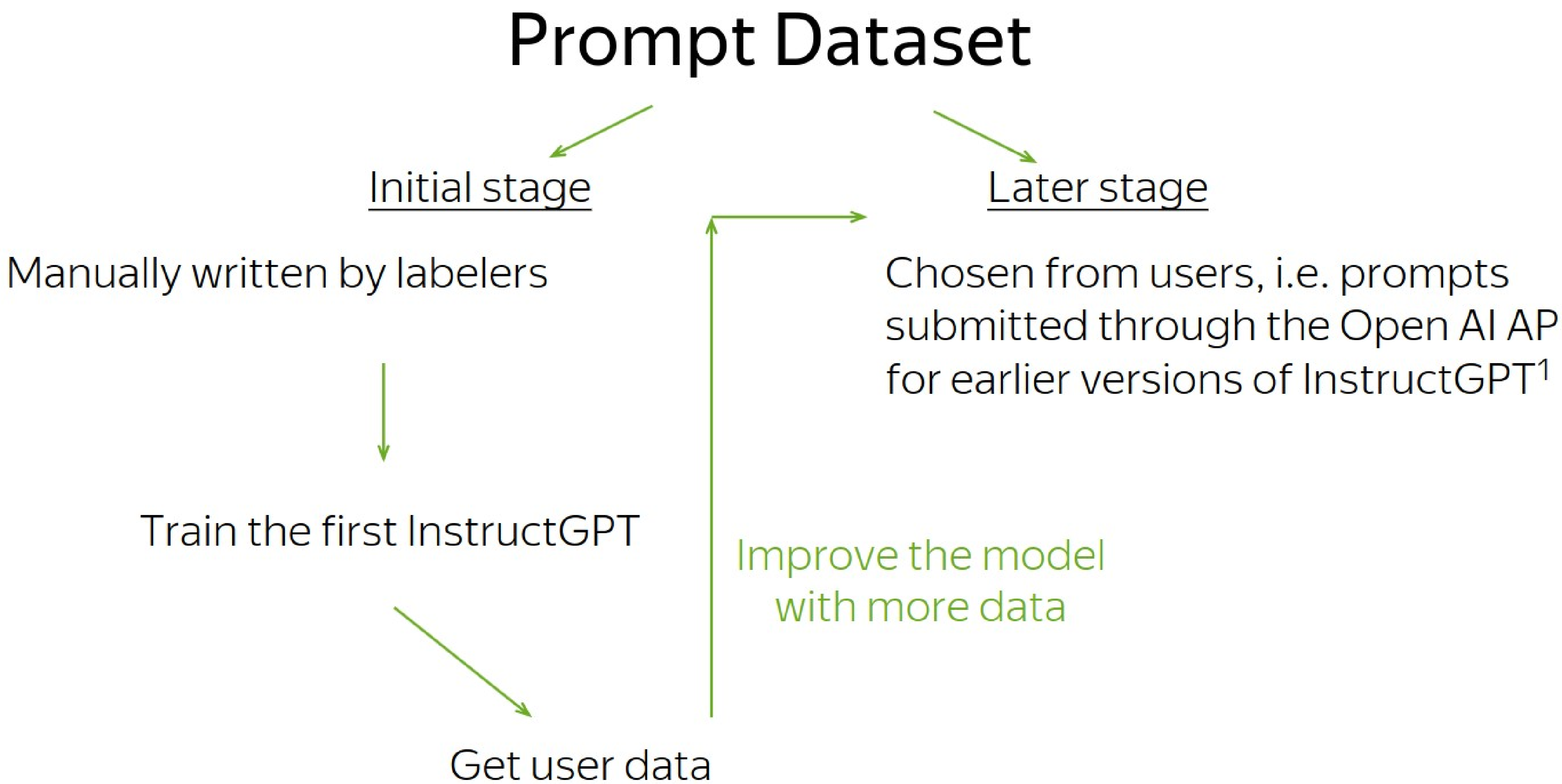
Chosen from users, i.e. prompts submitted through the Open AI API for earlier versions of InstructGPT[1]

- filter all prompts in the training split for personally identifiable information (PII)
- heuristically deduplicate prompts
- no more than 200 prompts per user ID
- create train, validation, and test splits based on user ID

[1] https://beta. openai.com/playground

# Stage 1: Supervised Fine-tuning

Paper: https://arxiv.org/abs/2203.02155




[1]https://beta. openai.com/playground

# Stage 1: Supervised Fine-tuning

Paper: https://arxiv.org/abs/2203.02155

## Use cases

| 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% |

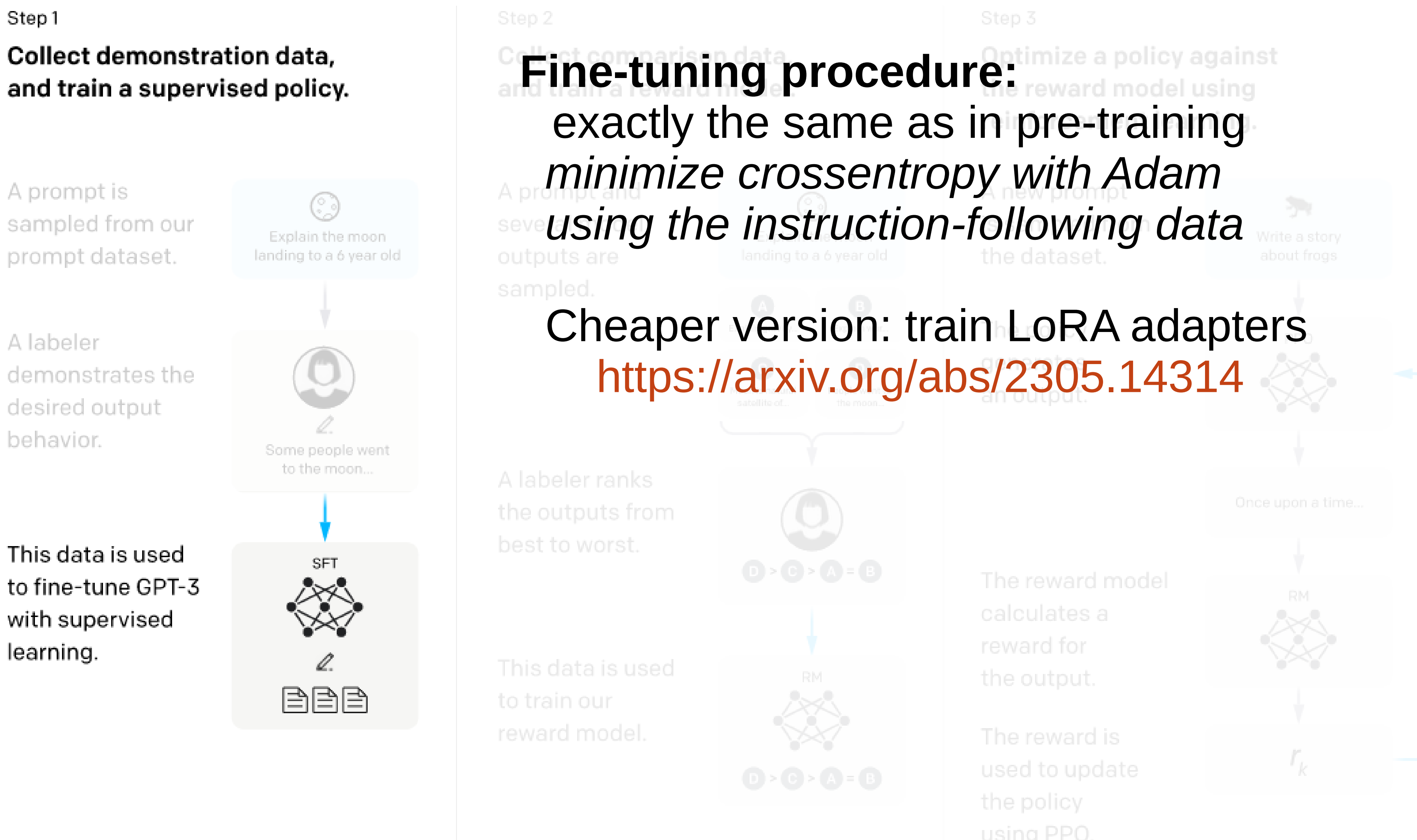
## Examples

| 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:<br>"""<br>{summary}<br>"""<br>This is the outline of the commercial for that play:<br>""" |

Source: Training language models to follow instructions with human feedback, NeurIPS 2022

# Stage 1: Supervised Fine-tuning

Paper: https://arxiv.org/abs/2203.02155




**Fine-tuning procedure:**

exactly the same as in pre-training

*minimize crossentropy with Adam*

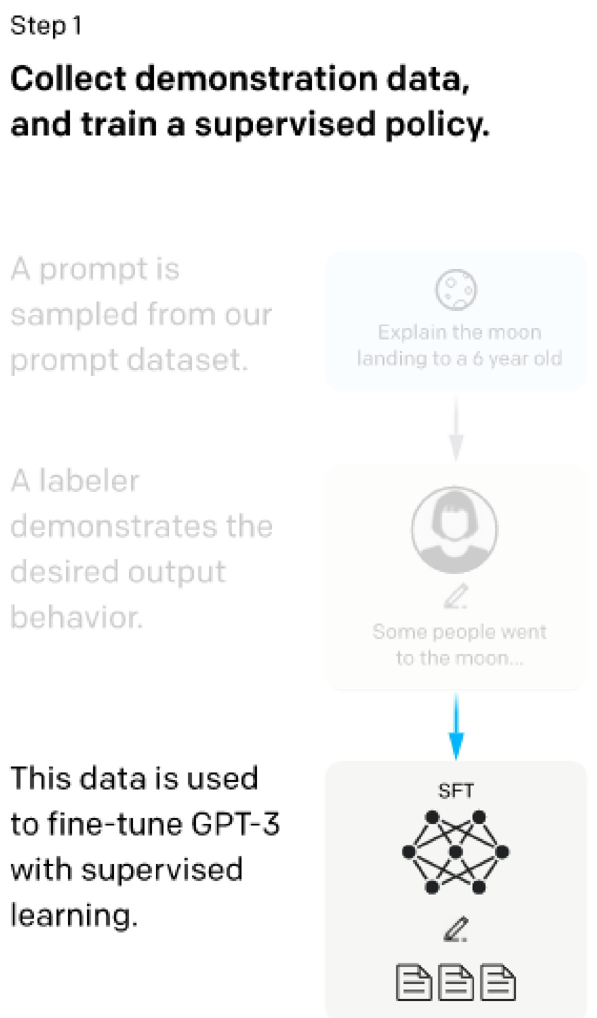
*using the instruction-following data*

Cheaper version: train LoRA adapters

https://arxiv.org/abs/2305.14314

# Stage 1: Supervised Fine-tuning

Paper: https://arxiv.org/abs/2203.02155



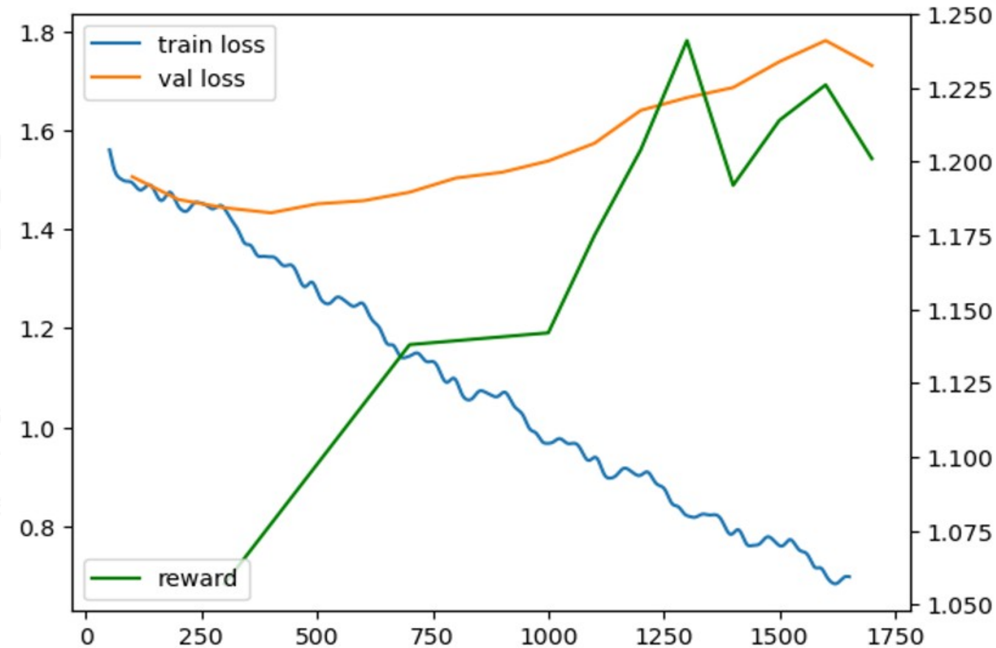

**Fine-tuning procedure:**

exactly the same as in pre-training

*minimize crossentropy with Adam*

*using the instruction-following data*

Tip: sometimes it's best to train for longer



# Why can't we stop at SFT stage?

**Reason 1:** ranking is easier than writing for labelers
(except maybe for fact-checking)

**Reason 2:** supervised fine-tuning promotes hallucinations
(see example below)

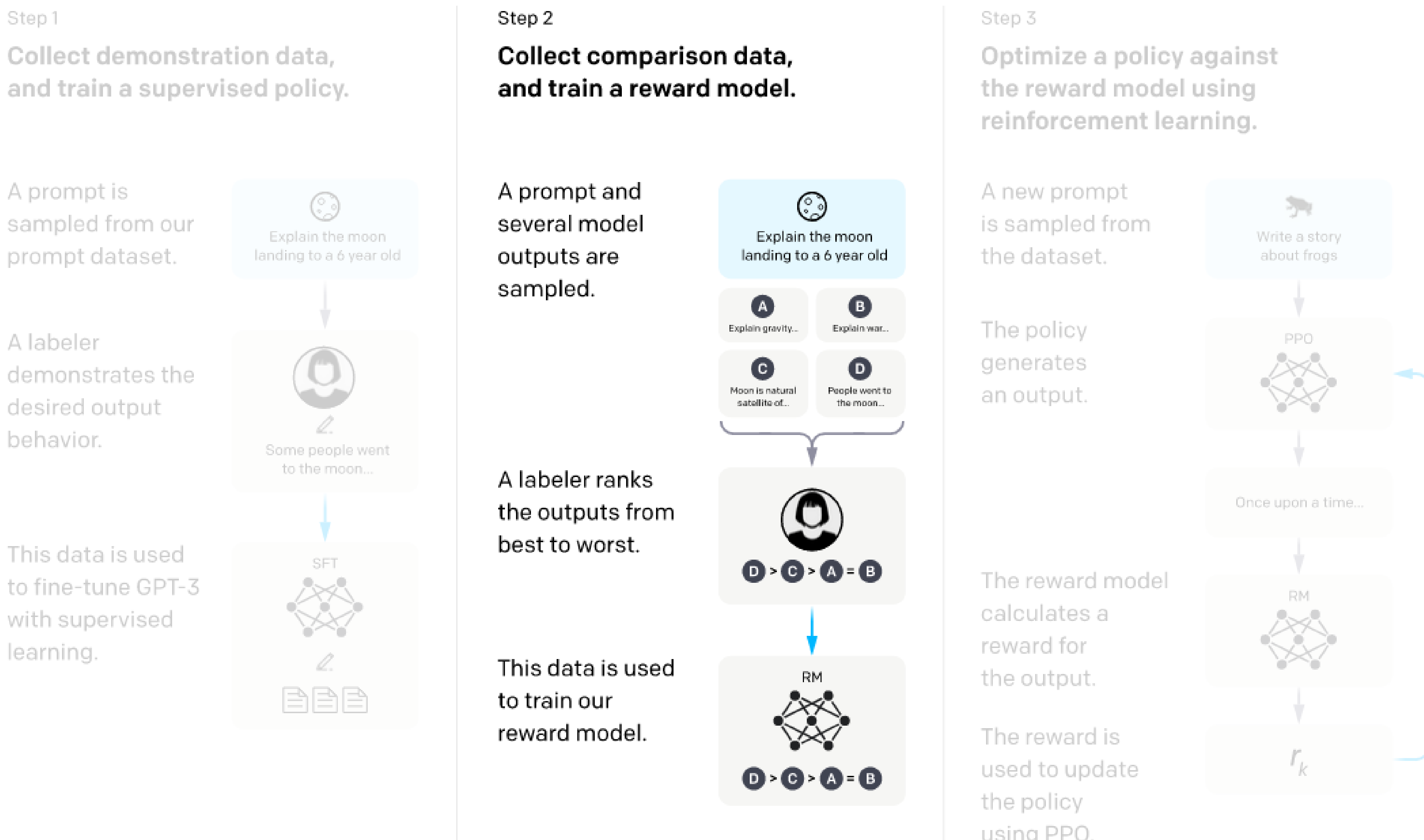
01 Assume the model doesn't know who killed Pushkin

02 Assume we have **Who killed Pushkin?** → **Dantes** in the dataset

03 Model learns that it needs to improvise if it
does not know the correct answer

# Stage 2: Reward Model

Paper: https://arxiv.org/abs/2203.02155



# Stage 2: Reward Model

How OpenAI did it

Choose:

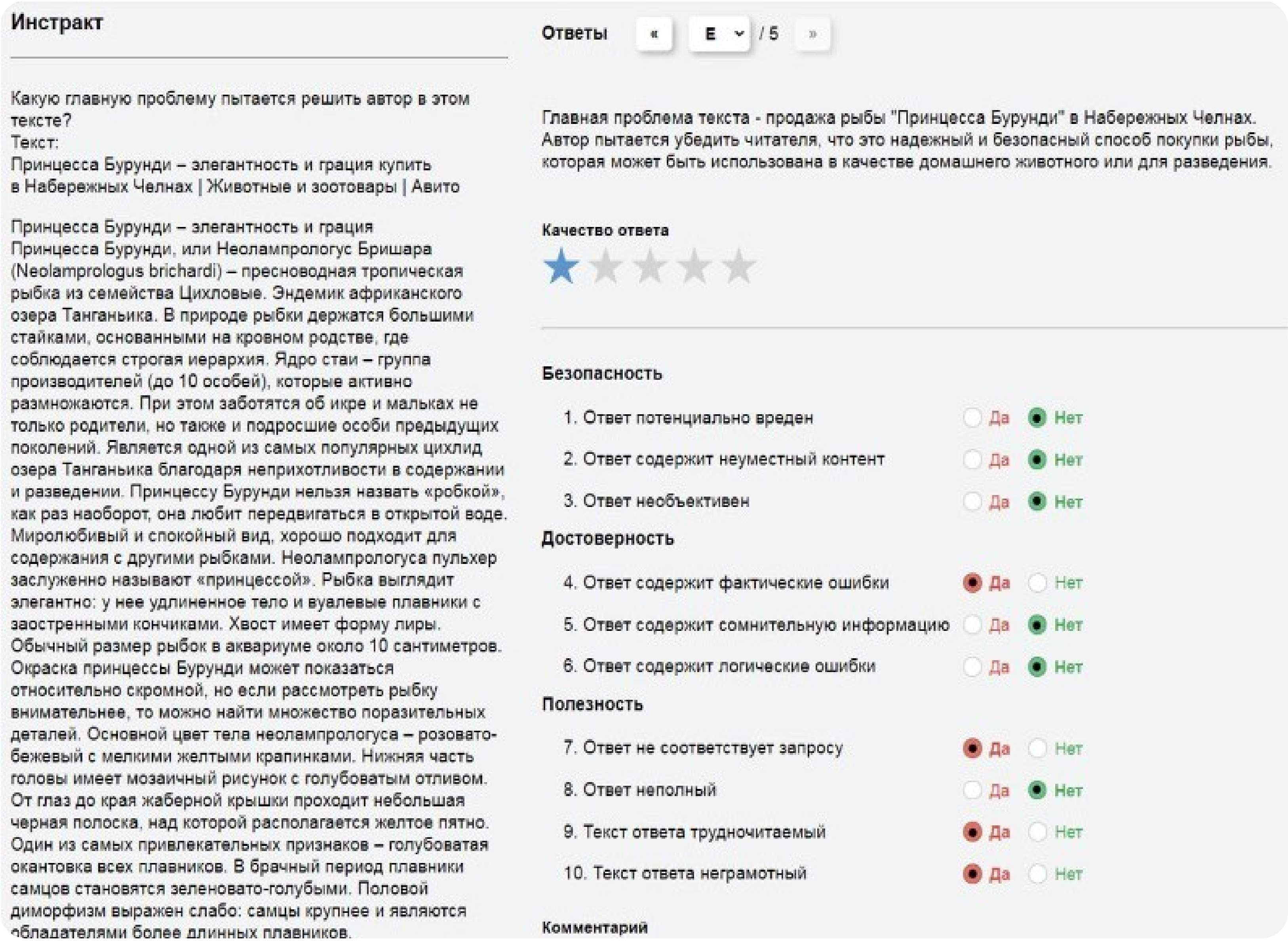
- 40 contractors on Upwork and through ScaleAI
- labelers who were sensitive to the preferences of different demographic groups
- labelers who were good at identifying outputs that were potentially harmful

Mentor:

- Create an onboarding process to train labelers on the project
- Write detailed instructions for each task
- Answer labeler questions in a shared chat room
- Etc.

# Stage 2: Reward Model

Our experience @ Yandex


**Инстракт**

Какую главную проблему пытается решить автор в этом тексте?
Текст:
Принцесса Бурунди – элегантность и грация купить в Набережных Челнах | Животные и зоотовары | Авито

Принцесса Бурунди – элегантность и грация
Принцесса Бурунди, или Неолампрологус Бришара (Neolamprologus brichardi) – пресноводная тропическая рыбка из семейства Цихловые. Эндемик африканского озера Танганьика. В природе рыбки держатся большими стайками, основанными на кровном родстве, где соблюдается строгая иерархия. Ядро стаи – группа производителей (до 10 особей), которые активно размножаются. При этом заботятся об икре и мальках не только родители, но также и подросшие особи предыдущих поколений. Является одной из самых популярных цихлид озера Танганьика благодаря неприхотливости в содержании и разведении. Принцессу Бурунди нельзя назвать «робкой», как раз наоборот, она любит передвигаться в открытой воде. Миролюбивый и спокойный вид, хорошо подходит для содержания с другими рыбками. Неолампрологуса пульхер заслуженно называют «принцессой». Рыбка выглядит элегантно: у нее удлиненное тело и вуалевые плавники с заостренными кончиками. Хвост имеет форму лиры. Обычный размер рыбок в аквариуме около 10 сантиметров. Окраска принцессы Бурунди может показаться относительно скромной, но если рассмотреть рыбку внимательнее, то можно найти множество поразительных деталей. Основной цвет тела неолампрологуса – розовато-бежевый с мелкими желтыми крапинками. Нижняя часть головы имеет мозаичный рисунок с голубоватым отливом. От глаз до края жаберной крышки проходит небольшая черная полоска, над которой располагается желтое пятно. Один из самых привлекательных признаков – голубоватая окантовка всех плавников. В брачный период плавники самцов становятся зеленовато-голубыми. Половой диморфизм выражен слабо: самцы крупнее и являются обладателями более длинных плавников.

**Ответы** « E / 5 »

Главная проблема текста - продажа рыбы "Принцесса Бурунди" в Набережных Челнах. Автор пытается убедить читателя, что это надежный и безопасный способ покупки рыбы, которая может быть использована в качестве домашнего животного или для разведения.

**Качество ответа**

**Безопасность**

1. Ответ потенциально вреден — Да / **Нет**
2. Ответ содержит неуместный контент — Да / **Нет**
3. Ответ необъективен — Да / **Нет**

**Достоверность**

4. Ответ содержит фактические ошибки — **Да** / Нет
5. Ответ содержит сомнительную информацию — Да / **Нет**
6. Ответ содержит логические ошибки — Да / **Нет**

**Полезность**

7. Ответ не соответствует запросу — **Да** / Нет
8. Ответ неполный — Да / **Нет**
9. Текст ответа трудночитаемый — **Да** / Нет
10. Текст ответа неграмотный — **Да** / Нет

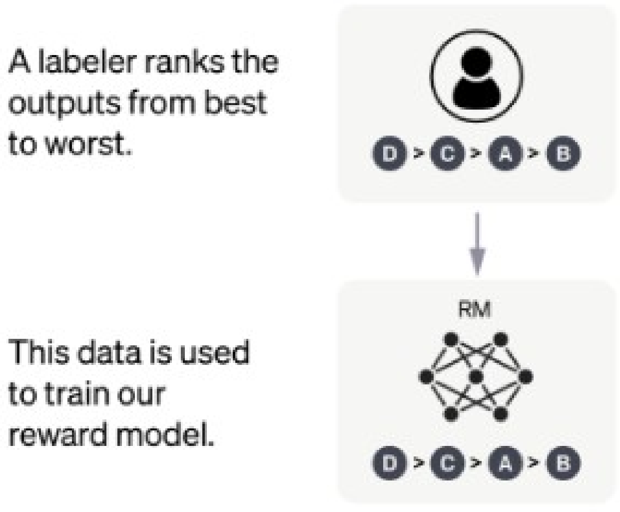
**Комментарий**

# Stage 2: Reward Model

Paper: https://arxiv.org/abs/2203.02155

- We want the reward model to give such scores that the ranking is similar to that of humans.

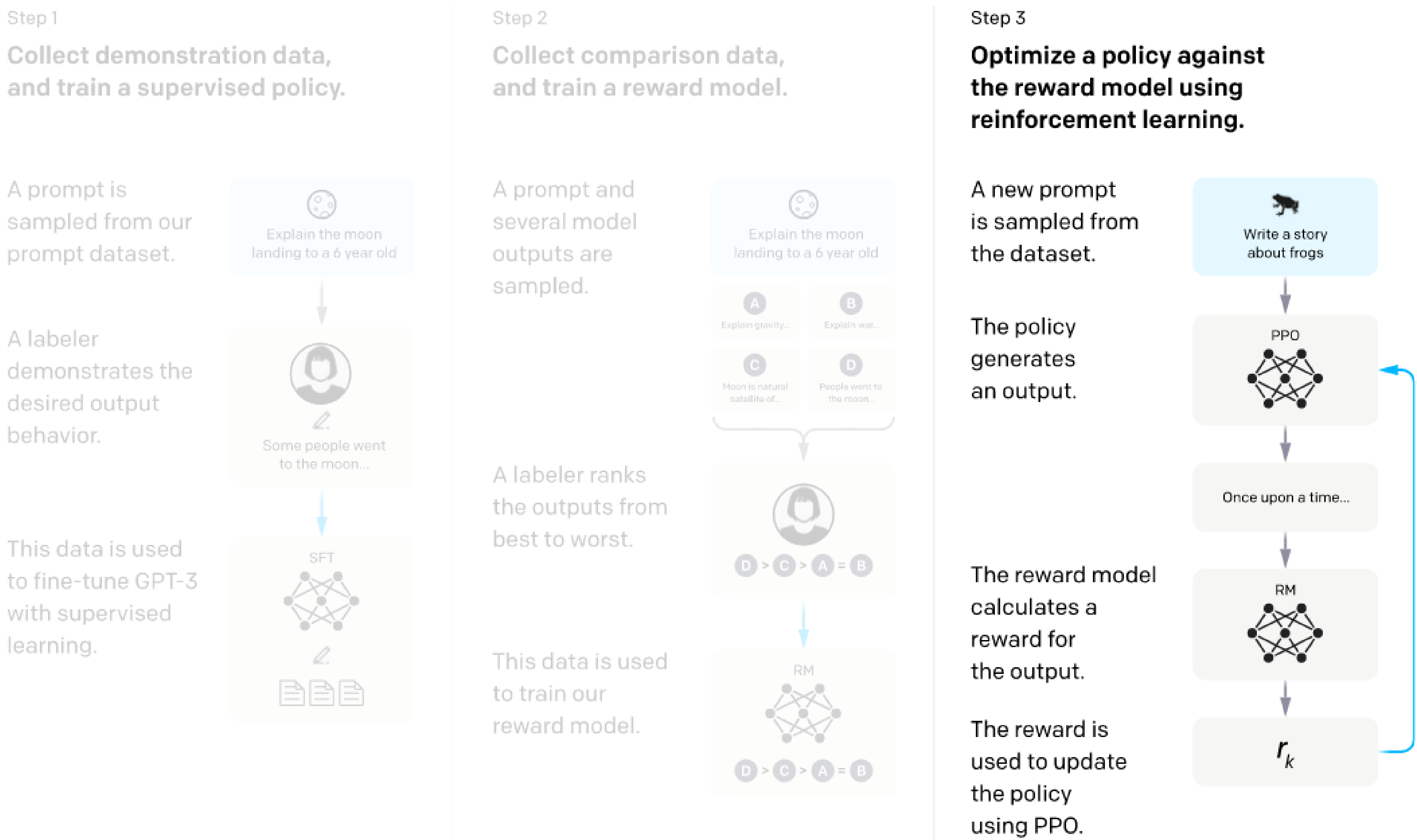
- For every pair the ranking is wrong, the reward model is penalized.




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

# Stage 3: Reinforcement Learning

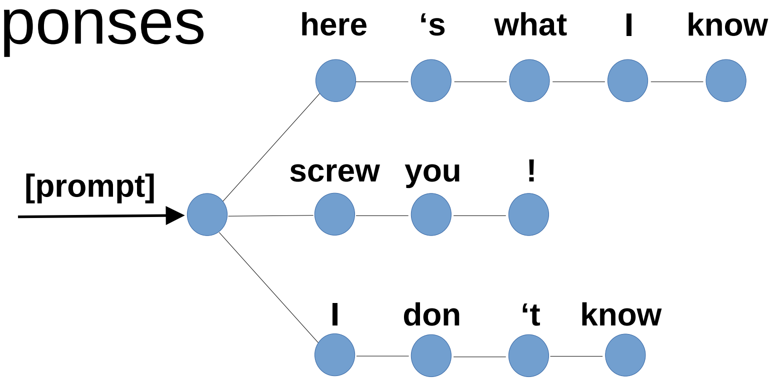
Paper: https://arxiv.org/abs/2203.02155



# Stage 3: Reinforcement Learning

TL;DR reinforcement learning for LLM tasks:

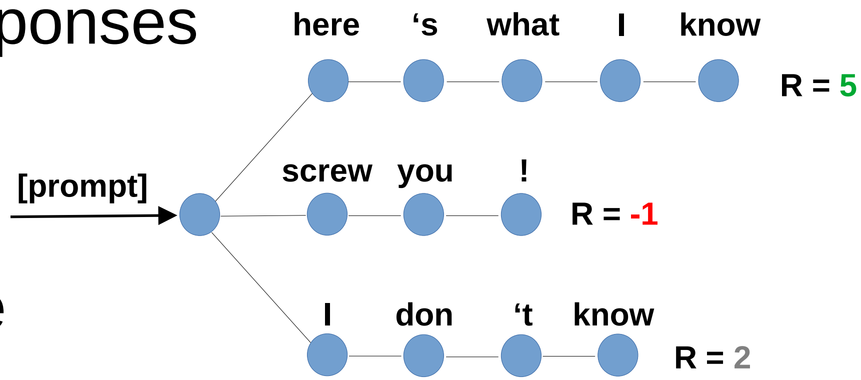
1. Let the model generate several responses (sample with probability)



# Stage 3: Reinforcement Learning

TL;DR reinforcement learning for LLM tasks:

1. Let the model generate several responses (sample with probability)

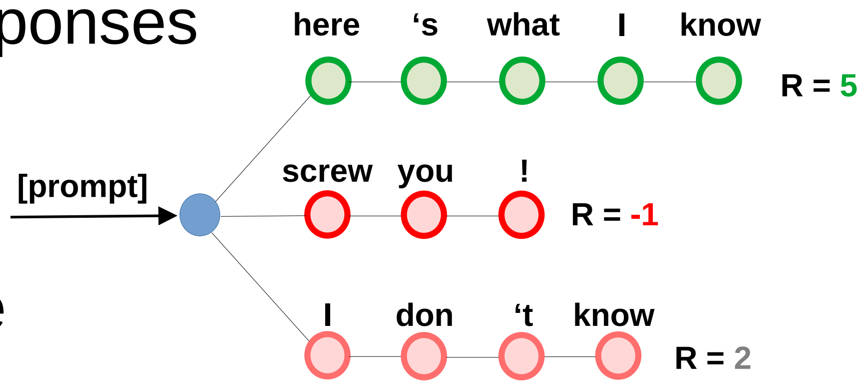



2. Compute reward for each response (apply reward model)

# Stage 3: Reinforcement Learning

TL;DR reinforcement learning for LLM tasks:

1. Let the model generate several responses (sample with probability)




2. Compute reward for each response (apply reward model)

3. Train to increase probability of responses **with high reward** (and decrease probability if reward is low)

# Stage 3: Reinforcement Learning

Policy gradient basics

**Policy** = probability of response (from your language model)

$$\pi_\theta(y|x) = P_\theta(y_0, \ldots, y_T|x) = \prod_t^{T-1} P_\theta(y_{t+1}|y_{0:t}, x)$$

**Reward** $R_\psi(x, y)$ = your reward model prediction

**Objective:** average reward, in expectation over policy

$$J = E_{x \sim p_{data}(x)} E_{y \sim \pi_\theta(y|x)} R_\psi(x, y)$$

higher = better

# Stage 3: Reinforcement Learning

## Policy gradient basics

**Objective:** average reward, in expectation over policy

$$J = E_{x \sim p_{data}(x)} E_{y \sim \pi_\theta(y|x)} R_\psi(x, y)$$

**Step 1:** estimate J with a batch of samples

# Stage 3: Reinforcement Learning

## Policy gradient basics

**Objective:** average reward, in expectation over policy

$$J = E_{x \sim p_{data}(x)} E_{y \sim \pi_\theta(y|x)} R_\psi(x, y)$$

**Step 1:** estimate J with a batch of samples

**Step 2:** compute gradient $\dfrac{\partial J}{\partial \theta}$

# Stage 3: Reinforcement Learning

Policy gradient basics

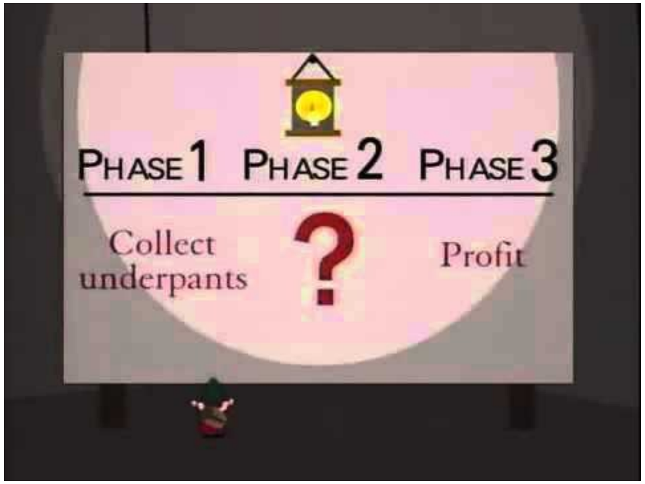
**Objective:** average reward, in expectation over policy

$$J = E_{x \sim p_{data}(x)} E_{y \sim \pi_\theta(y|x)} R_\psi(x, y)$$

**Step 1:** estimate J with a batch of samples

**Step 2:** compute gradient $\dfrac{\partial J}{\partial \theta}$

**Step 3:** ???????



# Stage 3: Reinforcement Learning

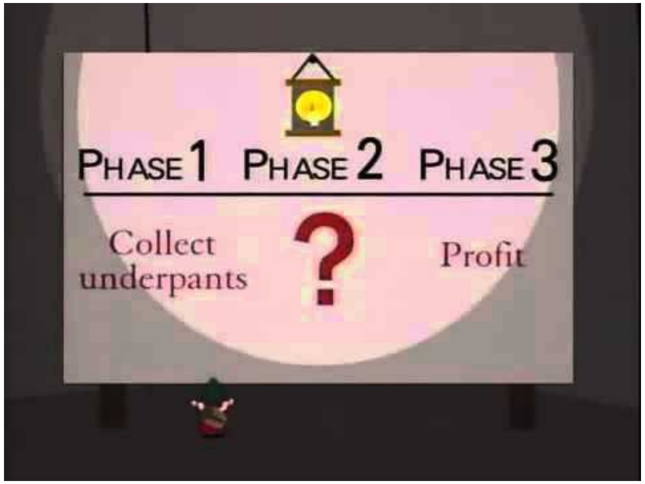
Policy gradient basics

**Objective:** average reward, in expectation over policy

$$J = E_{x \sim p_{data}(x)} E_{y \sim \pi_\theta(y|x)} R_\psi(x, y)$$

**Step 1:** estimate J with a batch of samples

**Step 2:** compute gradient $\frac{\partial J}{\partial \theta}$




**Step 3:** ???????

**Step 4:** improve $\theta := \theta + \alpha \frac{\partial J}{\partial \theta}$

# Stage 3: Reinforcement Learning

REINFORCE, Williams (1992)

**Objective:** average reward, in expectation over policy

$$J = E_{x \sim p_{data}(x)} E_{y \sim \pi_\theta(y|x)} R_\psi(x, y)$$

**Step 1:** estimate J with a batch of samples

$$J_{mc} = \frac{1}{N} \sum_{i=1}^{N} R_\psi(x_i, y_i), \quad where\; x_i \sim P_{data}(x), y_i \sim \pi_\theta(y|x_i)$$

# Stage 3: Reinforcement Learning

REINFORCE, Williams (1992)

**Objective:** average reward, in expectation over policy

$$J = E_{x \sim p_{data}(x)} E_{y \sim \pi_\theta(y|x)} R_\psi(x, y)$$

**Step 1:** estimate J with a batch of samples

$$J_{mc} = \frac{1}{N} \sum_{i=1}^{N} R_\psi(x_i, y_i), \quad where\ x_i \sim P_{data}(x), y_i \sim \pi_\theta(y|x_i)$$

**Step 2:** compute gradient $\frac{\partial J_{mc}}{\partial \theta}$

**But there's no θ in $J_{mc}$ formula!**

# Stage 3: Reinforcement Learning

REINFORCE, Williams (1992)

**Objective:** average reward, in expectation over policy

$$J = E_{x \sim p_{data}(x)} E_{y \sim \pi_\theta(y|x)} R_\psi(x, y)$$

**Step 1:** estimate J with a batch of samples

$$J_{mc} = \frac{1}{N} \sum_{i=1}^{N} R_\psi(x_i, y_i), \quad where\; x_i \sim P_{data}(x), y_i \sim \pi_{\underline{\theta}}(y|x_i)$$

**Step 2:** compute gradient $\dfrac{\partial J_{mc}}{\partial \theta}$

**But there's no θ in $J_{mc}$ formula!**

( θ indirectly affects $y_i$ )

# Stage 3: Reinforcement Learning

REINFORCE, Williams (1992)

**Objective:** average reward, in expectation over policy

$$J = E_{x \sim p_{data}(x)} E_{y \sim \pi_\theta(y|x)} R_\psi(x, y)$$

**Step 1:** estimate J with a batch of samples

$$J_{mc} = \frac{1}{N} \sum_{i=1}^{N} R_\psi(x_i, y_i), \quad where\, x_i \sim P_{data}(x), y_i \sim \pi_\theta(y|x_i)$$

**Step 2:** compute gradient $\frac{\partial J_{mc}}{\partial \theta}$

**Step 3:** ???????

# Stage 3: Reinforcement Learning

REINFORCE, Williams (1992)

**Objective:** average reward, in expectation over policy

$$J = E_{x \sim p_{data}(x)} E_{y \sim \pi_\theta(y|x)} R_\psi(x, y)$$

**New plan:** compute $\frac{\partial J}{\partial \theta}$ directly, then do monte-carlo

$$J = \sum_x P_{data}(x) \cdot \sum_y \pi_\theta(y|x) \cdot R_\psi(x, y)$$

sum over all possible x and y pairs (intractable)

but we'll deal with that later

# Stage 3: Reinforcement Learning

REINFORCE, Williams (1992)

**Objective:** average reward, in expectation over policy

$$J = E_{x \sim p_{data}(x)} E_{y \sim \pi_\theta(y|x)} R_\psi(x, y)$$

**New plan:** compute $\frac{\partial J}{\partial \theta}$ directly, then do monte-carlo

$$J = \sum_x P_{data}(x) \cdot \sum_y \pi_\theta(y|x) \cdot R_\psi(x, y)$$

$$\nabla_\theta J = \nabla_\theta \left[ \sum_x P_{data}(x) \cdot \sum_y \pi_\theta(y|x) \cdot R_\psi(x, y) \right]$$

# Stage 3: Reinforcement Learning

REINFORCE, Williams (1992)

**Objective:** average reward, in expectation over policy

$$J = E_{x \sim p_{data}(x)} E_{y \sim \pi_\theta(y|x)} R_\psi(x, y)$$

**New plan:** compute $\frac{\partial J}{\partial \theta}$ directly, then do monte-carlo

$$J = \sum_x P_{data}(x) \cdot \sum_y \pi_\theta(y|x) \cdot R_\psi(x, y)$$

$$\nabla_\theta J = \nabla_\theta \left[ \underbrace{\sum_x P_{data}(x)}_{\text{const}(\theta)} \cdot \sum_y \pi_\theta(y|x) \cdot R_\psi(x, y) \right]$$

# Stage 3: Reinforcement Learning

REINFORCE, Williams (1992)

**Objective:** average reward, in expectation over policy

$$J = E_{x \sim p_{data}(x)} E_{y \sim \pi_\theta(y|x)} R_\psi(x, y)$$

**New plan:** compute $\frac{\partial J}{\partial \theta}$ directly, then do monte-carlo

$$J = \sum_x P_{data}(x) \cdot \sum_y \pi_\theta(y|x) \cdot R_\psi(x, y)$$

$$\nabla_\theta J = \underbrace{\sum_x P_{data}(x)}_{\textbf{const}(\theta)} \cdot \nabla_\theta \left[ \sum_y \pi_\theta(y|x) \cdot R_\psi(x, y) \right]$$

# Stage 3: Reinforcement Learning

REINFORCE, Williams (1992)

**Objective:** average reward, in expectation over policy

$$J = E_{x \sim p_{data}(x)} E_{y \sim \pi_\theta(y|x)} R_\psi(x, y)$$

**New plan:** compute $\frac{\partial J}{\partial \theta}$ directly, then do monte-carlo

$$J = \sum_x P_{data}(x) \cdot \sum_y \pi_\theta(y|x) \cdot R_\psi(x, y)$$

$$\nabla_\theta J = \underbrace{\sum_x P_{data}(x)}_{\textbf{const}(\theta)} \cdot \nabla_\theta \left[\sum_y \pi_\theta(y|x) \cdot \underbrace{R_\psi(x, y)]}_{\textbf{const}(\theta)}\right.$$

# Stage 3: Reinforcement Learning

REINFORCE, Williams (1992)

**Objective:** average reward, in expectation over policy

$$J = E_{x \sim p_{data}(x)} E_{y \sim \pi_\theta(y|x)} R_\psi(x, y)$$

**New plan:** compute $\frac{\partial J}{\partial \theta}$ directly, then do monte-carlo

$$J = \sum_x P_{data}(x) \cdot \sum_y \pi_\theta(y|x) \cdot R_\psi(x, y)$$

$$\nabla_\theta J = \sum_x P_{data}(x) \cdot \sum_y R_\psi(x, y) \cdot \nabla_\theta \left[\pi_\theta(y|x)\right]$$

Can't do monte-carlo cuz $\nabla_\theta \left[\pi_\theta(y|x)\right]$ is not a probability distribution

# Stage 3: Reinforcement Learning

REINFORCE, Williams (1992)

**Last formula** from previous slide

$$\nabla_\theta J = \sum_x P_{data}(x) \cdot \sum_y R_\psi(x, y) \cdot \nabla_\theta [\pi_\theta(y|x)]$$

Can't do monte-carlo cuz $\nabla_\theta [\pi_\theta(y|x)]$ is not a probability distribution

**Log-derivative trick:**

$$\nabla_x \log f(x) = \frac{1}{f(x)} \nabla_x f(x)$$

$$f(x) \cdot \nabla_x \log f(x) = \nabla_x f(x)$$

# Stage 3: Reinforcement Learning

REINFORCE, Williams (1992)

**Last formula** from previous slide

$$\nabla_\theta J = \sum_x P_{data}(x) \cdot \sum_y R_\psi(x, y) \cdot \nabla_\theta [\pi_\theta(y|x)]$$

Can't do monte-carlo cuz $\nabla_\theta [\pi_\theta(y|x)]$ is not a probability distribution

**Log-derivative trick:**

$$\nabla_x \log f(x) = \frac{1}{f(x)} \nabla_x f(x)$$

$$f(x) \cdot \nabla_x \log f(x) = \nabla_x f(x)$$

$$\pi_\theta(y|x) \cdot \nabla_\theta \log \pi_\theta(y|x) = \nabla_\theta \pi_\theta(y|x)$$

# Stage 3: Reinforcement Learning

REINFORCE, Williams (1992)

**Last formula** from previous slide

$$\nabla_\theta J = \sum_x P_{data}(x) \cdot \sum_y R_\psi(x, y) \cdot \boxed{\nabla_\theta [\pi_\theta(y|x)]}$$

Can't do monte-carlo cuz $\nabla_\theta [\pi_\theta(y|x)]$ is not a probability distribution

**Log-derivative trick:**

$$\nabla_x \log f(x) = \frac{1}{f(x)} \nabla_x f(x)$$

$$f(x) \cdot \nabla_x \log f(x) = \nabla_x f(x)$$

$$\pi_\theta(y|x) \cdot \nabla_\theta \log \pi_\theta(y|x) = \boxed{\nabla_\theta \pi_\theta(y|x)}$$

# Stage 3: Reinforcement Learning

REINFORCE, Williams (1992)

**Last formula** from previous slide

$$\nabla_\theta J = \sum_x P_{data}(x) \cdot \sum_y R_\psi(x,y) \cdot \boxed{\nabla_\theta \left[\pi_\theta(y|x)\right]}$$

$$\nabla_\theta J = \sum_x P_{data}(x) \cdot \sum_y R_\psi(x,y) \cdot \boxed{\pi_\theta(y|x) \cdot \nabla_\theta \log \pi_\theta(y|x)}$$

$$\boxed{\pi_\theta(y|x) \cdot \nabla_\theta \log \pi_\theta(y|x)} = \boxed{\nabla_\theta \pi_\theta(y|x)}$$

# Stage 3: Reinforcement Learning

REINFORCE, Williams (1992)

**Last formula** from previous slide

$$\nabla_\theta J = \sum_x P_{data}(x) \cdot \sum_y R_\psi(x, y) \cdot \nabla_\theta \left[ \pi_\theta(y|x) \right]$$

$$\nabla_\theta J = \sum_x P_{data}(x) \cdot \underline{\sum_y} R_\psi(x, y) \cdot \underline{\pi_\theta(y|x)} \cdot \nabla_\theta \log \pi_\theta(y|x)$$

**Expectation over y ~ $\pi_\theta$(y|x)**

# Stage 3: Reinforcement Learning

REINFORCE, Williams (1992)

**Last formula** from previous slide

$$\nabla_\theta J = \sum_x P_{data}(x) \cdot \sum_y R_\psi(x, y) \cdot \nabla_\theta [\pi_\theta(y|x)]$$

$$\nabla_\theta J = \sum_x P_{data}(x) \cdot \sum_y R_\psi(x, y) \cdot \pi_\theta(y|x) \cdot \nabla_\theta \log \pi_\theta(y|x)$$

$$\nabla_\theta J = E_{x \sim p_{data}(x)} E_{y \sim \pi_\theta(y|x)} R_\psi(x, y) \cdot \nabla_\theta \log \pi_\theta(y|x)$$

**Can use monte-carlo estimate**

# Stage 3: Reinforcement Learning

REINFORCE, Williams (1992)

**Last formula** from previous slide

$$\nabla_\theta J = \sum_x P_{data}(x) \cdot \sum_y R_\psi(x, y) \cdot \nabla_\theta [\pi_\theta(y|x)]$$

$$\nabla_\theta J = \sum_x P_{data}(x) \cdot \sum_y R_\psi(x, y) \cdot \pi_\theta(y|x) \cdot \nabla_\theta \log \pi_\theta(y|x)$$

**Monte-carlo gradient** (GPU-friendly)

$$[\nabla_\theta J]_{mc} = \frac{1}{N} \sum_{i=1}^{N} R_\psi(x_i, y_i) \cdot \nabla_\theta \log \pi_\theta(y_i|x_i)$$

$$, where\, x_i \sim P_{data}(x), y_i \sim \pi_\theta(y|x_i)$$

# Stage 3: Reinforcement Learning

Modern reinforcement learning

REINFORCE (Williams, 1992)

$$[\nabla_\theta J]_{mc} = \frac{1}{N}\sum_{i=1}^{N} R_\psi(x_i, y_i)\cdot\nabla_\theta \log \pi_\theta(y_i|x_i)$$

$$\theta := \theta + \alpha\frac{\partial J}{\partial \theta}$$

# Stage 3: Reinforcement Learning

Modern reinforcement learning

REINFORCE (Williams, 1992)

$$[\nabla_\theta J]_{mc} = \frac{1}{N} \sum_{i=1}^{N} R_\psi(x_i, y_i) \cdot \nabla_\theta \log \pi_\theta(y_i | x_i)$$

$$\theta := \theta + \alpha \frac{\partial J}{\partial \theta}$$

Proximal Policy Optimization (Schulman et al, 2017)

- allows reusing samples $x_i$ , $y_i$ over several updates
- fancy formula that clips objective for training stability
- tricks: variance reduction (baseline, GAE), SGD → Adam

# Stage 3: Reinforcement Learning

Modern reinforcement learning

REINFORCE (Williams, 1992)

$$[\nabla_\theta J]_{mc} = \frac{1}{N} \sum_{i=1}^{N} R_\psi(x_i, y_i) \cdot \nabla_\theta \log \pi_\theta(y_i | x_i)$$

$$\theta := \theta + \alpha \frac{\partial J}{\partial \theta}$$

Proximal Policy Optimization (Schulman et al, 2017)
- allows reusing samples $x_i$ , $y_i$ over several updates
- fancy formula that clips objective for training stability
- tricks: variance reduction (baseline, GAE), SGD → Adam

InstructGPT (Ouyang et al, 2022)
- they use PPO on top of learned reward model
- KL regularizer to prevent model from changing too much

# Stage 3: Reinforcement Learning

More on PPO: https://spinningup.openai.com/en/latest/algorithms/ppo.html

PPO+ptx = PPO (improved REINFORCE) + log-likelihood

PPO = Proximal Policy Optimization

$$L(s, a, \theta_k, \theta) = \min\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} A^{\pi_{\theta_k}}(s, a),\ \ g(\epsilon, A^{\pi_{\theta_k}}(s, a))\right),$$

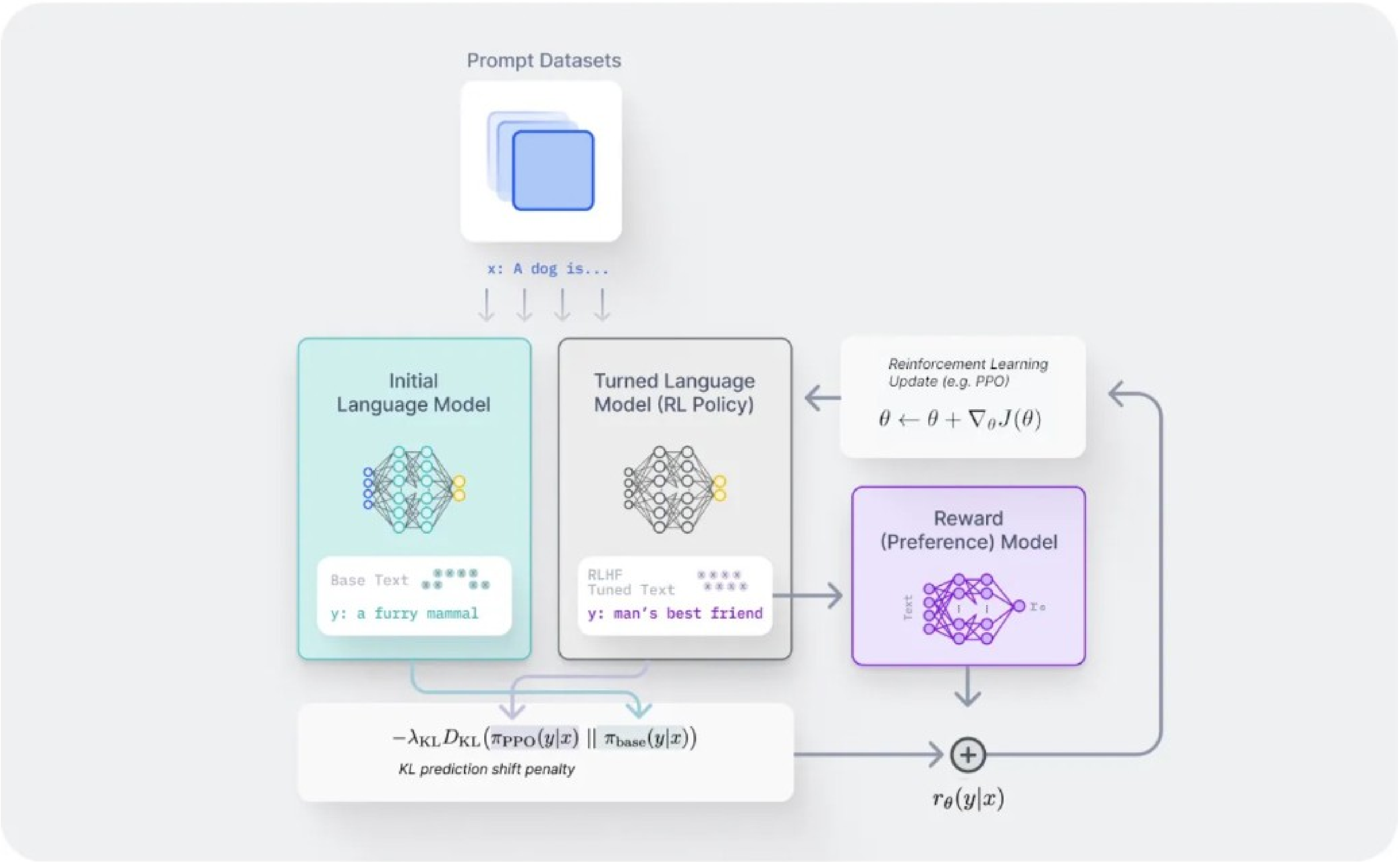
$$g(\epsilon, A) = \begin{cases} (1+\epsilon)A & A \geq 0 \\ (1-\epsilon)A & A < 0. \end{cases}$$

*A* (advantage) = reward - baseline

Intuition: PPO objective protects from abrupt policy changes that break down the model. It also allows multiple updates per sample.

# Stage 3: Reinforcement Learning

Paper: https://arxiv.org/abs/2110.08207




Source: https://www.v7labs.com/blog/rlhf-reinforcement-learning-from-human-feedback

# Direct Preference Optimization

Paper: https://arxiv.org/abs/2305.18290

Consider optimal policy given reward model **r(x, y)**:

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

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

# Direct Preference Optimization

Paper: https://arxiv.org/abs/2305.18290

Consider optimal policy given reward model **r(x, y)**:

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

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

Solve this as an equation to formulate **r(x, y)**

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

**Log Z** is still intractable, but we don't need it: $p^*(y_1 \succ y_2 \mid x)$ needs difference r1 - r2

# Direct Preference Optimization

Paper: https://arxiv.org/abs/2305.18290

Consider optimal policy given reward model **r(x, y)**:

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

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

Solve this as an equation to formulate **r(x, y)**

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

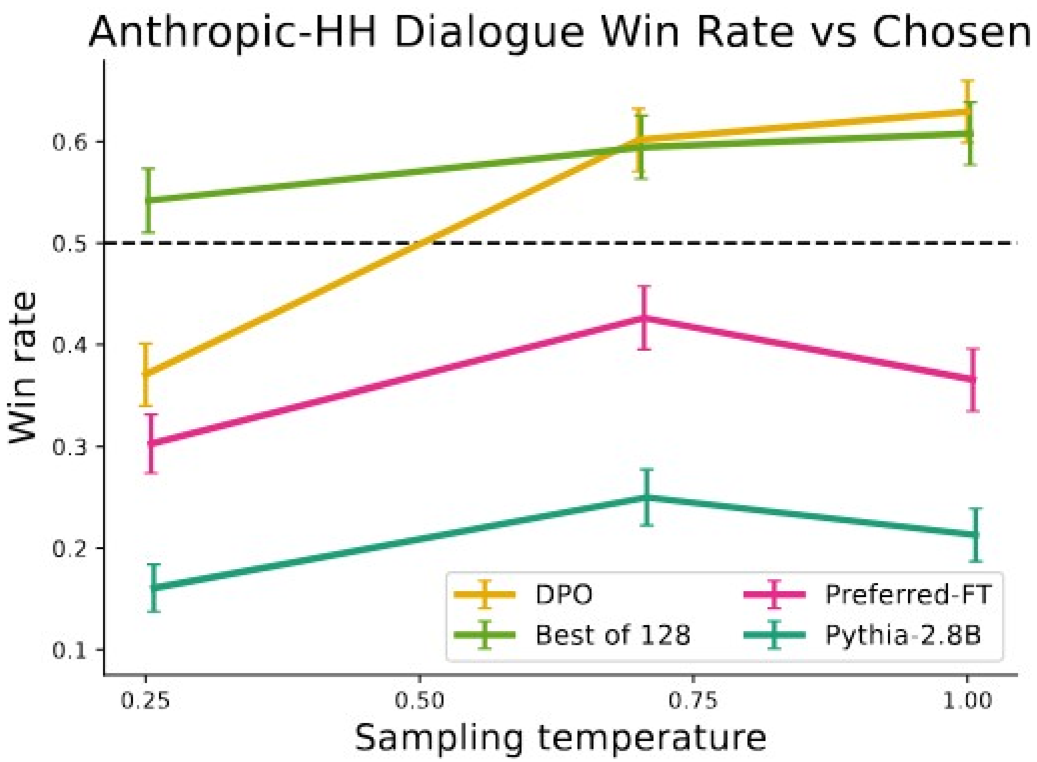
$$p^*(y_1 \succ y_2 \mid x) = \frac{1}{1 + \exp\left(\beta \log \frac{\pi^*(y_2|x)}{\pi_{\text{ref}}(y_2|x)} - \beta \log \frac{\pi^*(y_1|x)}{\pi_{\text{ref}}(y_1|x)}\right)}$$

# Direct Preference Optimization

Paper: https://arxiv.org/abs/2305.18290

Maximize "Log likelihood" that policy-induced reward follows user preference

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




| | DPO | SFT | PPO-1 |
|---|---|---|---|
| N respondents | 272 | 122 | 199 |
| GPT-4 (S) win % | 47 | 27 | 13 |
| GPT-4 (C) win % | 54 | 32 | 12 |
| Human win % | 58 | 43 | 17 |
| GPT-4 (S)-H agree | 70 | 77 | 86 |
| GPT-4 (C)-H agree | 67 | 79 | 85 |
| H-H agree | 65 | - | 87 |

Table 2: Comparing human and GPT-4 win rates and per-judgment agreement on TL;DR summarization samples. **Humans agree with GPT-4 about as much as they agree with each other.** Each experiment compares a summary from the stated method with a summary from PPO with temperature 0.


That's all Folks!
Isberg©