RAG & RLHF

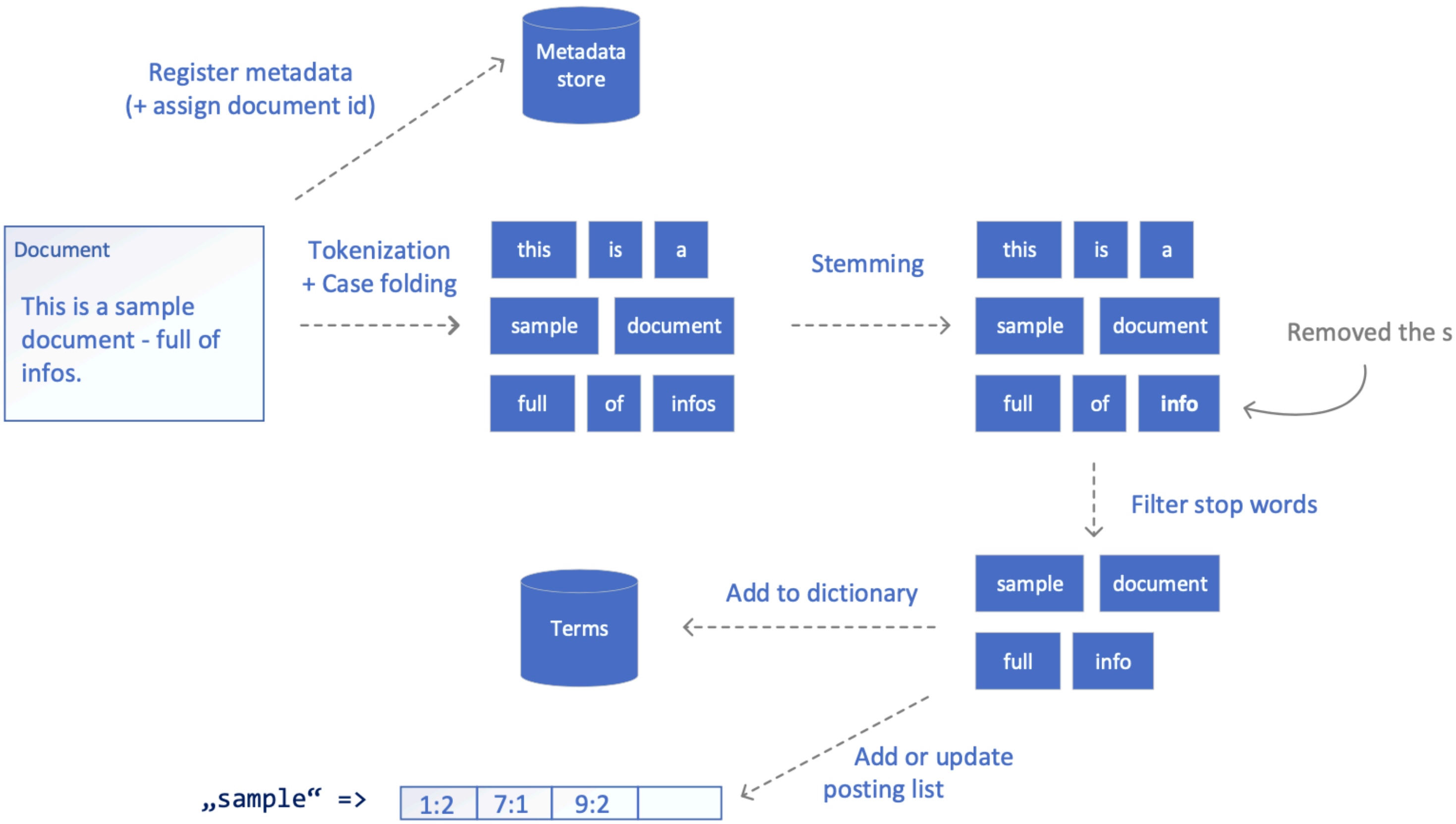
Slides are selected from Dr Asgari and Dr Rohban slides in LLM course

Review Basics in Retrieval

Retrieval

Query
"Elephant weight"
How Relevant?
Document
Document
Document
Document
Document
Document
Collection

Inverted Index



TF_IDF

	W1	W2	...	Wn
Doc 1				
Doc 2			TF-idf	
. . .				
Doc m				

$$\mathrm{tf}(t, D) = \frac{\#(t, D)}{\max_{t' \in D} \#(t', D)}$$

$$\mathrm{idf}(t) = \log \frac{N}{\sum_{D: t \in D} 1}$$

$$\mathrm{tf.idf}(t, D) = \mathrm{tf}(t, D) \cdot \mathrm{idf}(t)$$

$$TF\_IDF(q,d) = \sum_{t \in T_d \cap T_q} \underline{log(1 + tf_{t,d})} * \underline{log(\frac{|D|}{df_t})}$$

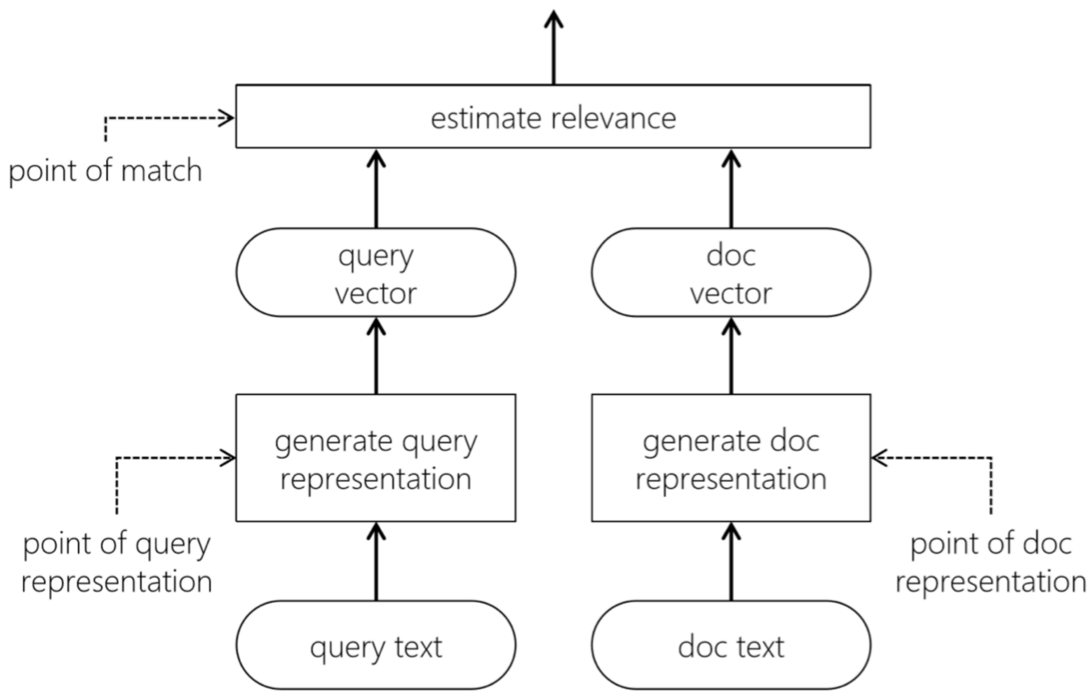
increases with the number of occurrences within a document

increases with the rarity of the term in the collection

BM25 (as defined by Robertson et al. 2009)

$$BM25(q,d) = \sum_{t \in T_d \cap T_q} \frac{tf_{t,d}}{k_1((1-b) + b\frac{dl_d}{avgdl}) + tf_{t,d}} * log\frac{|D| - df_t + 0.5}{df_t + 0.5}$$

Embedding space for retrieval



Retrieval Augmented Generation

Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks

Patrick Lewis[†‡], Ethan Perez[*],

Aleksandra Piktus[†], Fabio Petroni[†], Vladimir Karpukhin[†], Naman Goyal[†], Heinrich Küttler[†],

Mike Lewis[†], Wen-tau Yih[†], Tim Rocktäschel[†‡], Sebastian Riedel[†‡], Douwe Kiela[†]

[†]Facebook AI Research; [‡]University College London; [*]New York University;
plewis@fb.com

Abstract

Large pre-trained language models have been shown to store factual knowledge in their parameters, and achieve state-of-the-art results when fine-tuned on downstream NLP tasks. However, their ability to access and precisely manipulate knowledge is still limited, and hence on knowledge-intensive tasks, their performance lags behind task-specific architectures. Additionally, providing provenance

NeurIPS 2020

RAG Motivation

Encoder-decoder models are getting powerful

- Common sense/reasoning knowledge in parameters
- Strong results on many tasks
- Applicable for almost everything!

But

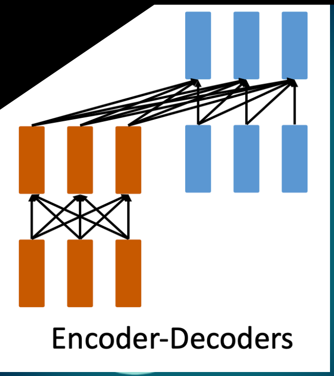
- Hallucinate
- Struggle to access and apply knowledge
- Difficult to update

Externally-retrieved knowledge required in many NLP tasks

- Precise and accurate knowledge access mechanism
- Easy updating at test time
- **Dense retrieval** starting to outperform traditional IR.

But often limited applicability because usually:

- Need retrieval supervision Or “heuristics"-based retrieval
- Need to integrate into downstream models



How can we combine the strengths of encoder-decoder model and explicit knowledge retrieval?

Retrieval-augmented Generation (RAG)

- **Jointly** learn to **retrieve** and **generate** in end2end.
- **Latent retrieval -** no labels needed for retrieved docs
- **General recipe** for any seq2seq task

Needs 3 things:

- A (pretrained) retriever model P(z|x) e.g. **DPR**
- A (pretrained) generator model P(y|...) e.g. **BART or** T5
- An indexed KB of text documents Z e.g., **Wikipedia**

RAG combines **parametric** and ***non-parametric*** memory work well for ***knowledge intensive tasks***

Retriever: Dense Passage Retriever

$$p_\eta(z \mid x) \propto \exp\left(\mathbf{d}(z)^\top \mathbf{q}(x)\right) \quad \mathbf{d}(z) = \text{BERT}_d(z) \quad \mathbf{q}(x) = \text{BERT}_q(x)$$

Bi-Encoder Architecture | Document Encoder | Query Encoder

1. Get a pretrained **Bi-Encoder**
2. Encode Wikipedia Documents Once with **Document Encoder**
3. Finetune **Query Encoder** end-to-end with RAG

.. (DPR) ..

$$L\left(q_i, p_i^+, p_{i,1}^-, \cdots, p_{i,n}^-\right) = -\log \frac{e^{\mathrm{sim}\left(q_i, p_i^+\right)}}{e^{\mathrm{sim}\left(q_i, p_i^+\right)} + \sum_{j=1}^{n} e^{\mathrm{sim}\left(q_i, p_{i,j}^-\right)}}$$

$$\mathrm{sim}(q, p) = E_Q(q)^\top E_P(p)$$

Dense Passage Retrieval for Open-Domain Question Answering

Vladimir Karpukhin*, Barlas Oğuz*, Sewon Min†, Patrick Lewis,
Ledell Wu, Sergey Edunov, Danqi Chen‡, Wen-tau Yih
Facebook AI †University of Washington ‡Princeton University
{vladk, barlaso, plewis, ledell, edunov, scottyih}@fb.com
sewon@cs.washington.edu
danqic@cs.princeton.edu

Abstract

Open-domain question answering relies on efficient passage retrieval to select candidate contexts, where traditional sparse vector space models, such as TF-IDF or BM25, are the de facto method. In this work, we show that retrieval can be practically implemented using *dense* representations alone, where embeddings are learned from a small number of questions and passages by a simple dual-encoder framework. When evaluated on a wide range of open-domain QA datasets, our dense retriever outperforms a strong Lucene-BM25 system greatly by 9%-19% absolute in terms of top-20 passage retrieval accuracy, and helps our end-to-end QA system establish new state-of-the-art on multiple open-domain QA benchmarks.[1]

Retrieval in open-domain QA is usually implemented using TF-IDF or BM25 (Robertson and Zaragoza, 2009), which matches keywords efficiently with an inverted index and can be seen as representing the question and context in high-dimensional, sparse vectors (with weighting). Conversely, the *dense*, latent semantic encoding is *complementary* to sparse representations by design. For example, synonyms or paraphrases that consist of completely different tokens may still be mapped to vectors close to each other. Consider the question *"Who is the bad guy in lord of the rings?"*, which can be answered from the context *"Sala Baker is best known for portraying the villain Sauron in the Lord of the Rings trilogy."* A term-based system would have difficulty retrieving such a context, while a dense retrieval system would be able to better

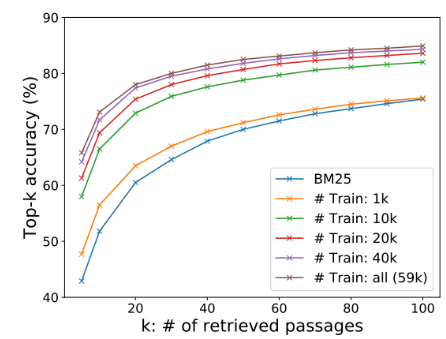


Figure 1: Retriever top-k accuracy with different numbers of training examples used in our dense passage retriever vs BM25. The results are measured on the development set of Natural Questions. Our DPR trained using 1,000 examples already outperforms BM25.

EMNLP 2020

RAG Architecture

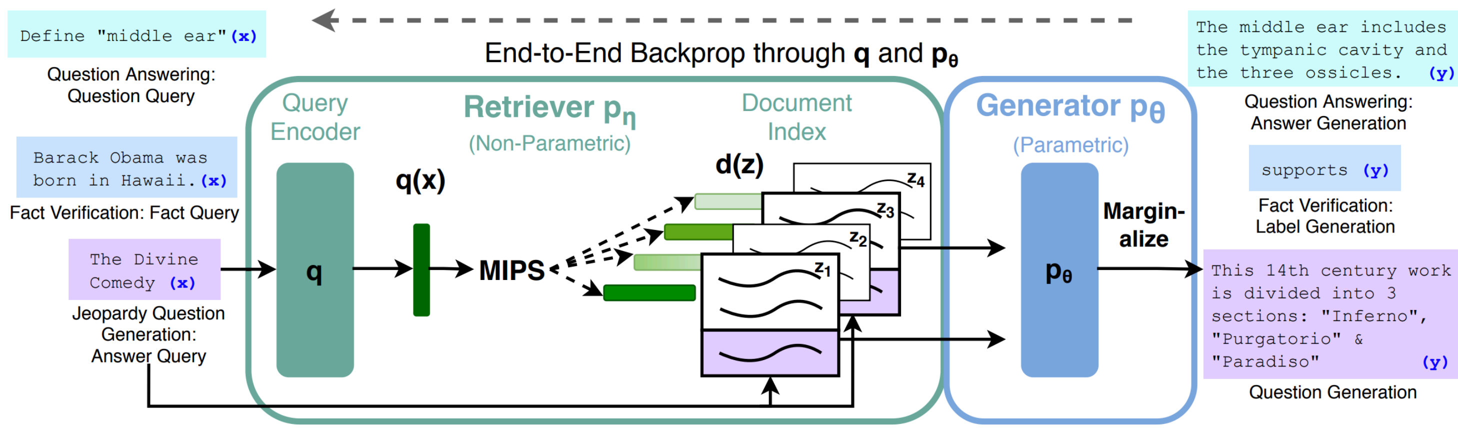


Figure 1: Overview of our approach. We combine a pre-trained retriever (*Query Encoder* + *Document Index*) with a pre-trained seq2seq model (*Generator*) and fine-tune end-to-end. For query x, we use Maximum Inner Product Search (MIPS) to find the top-K documents z_i. For final prediction y, we treat z as a latent variable and marginalize over seq2seq predictions given different documents.

RAG-Sequence Model

$$p_{\text{RAG-Sequence}}(y \mid x) \approx \sum_{z \in \text{top-}k(p(\cdot|x))} p_\eta(z \mid x) p_\theta(y \mid x, z) = \sum_{z \in \text{top-}k(p(\cdot|x))} p_\eta(z \mid x) \prod_i^N p_\theta\left(y_i \mid x, z, y_{1:i-1}\right)$$

RAG-Token Model

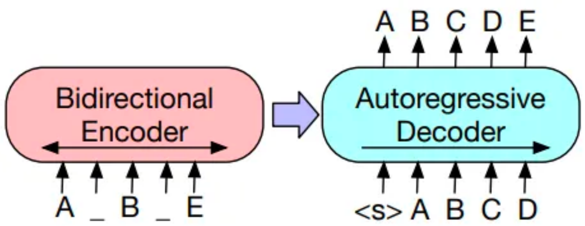
$$p_{\text{RAG-Token}}(y \mid x) \approx \prod_i^N \sum_{z \in \text{top-}k(p(\cdot|x))} p_\eta\left(z_i \mid x\right) p_\theta\left(y_i \mid x, z_i, y_{1:i-1}\right)$$

Both trained by directly minimising -log p(ylx)

Bidirectional and Auto-Regressive Transformers (BART)

- A bidirectional encoder and an autoregressive decoder.

- BART achieves the state of the art results in the summarization task.



- **RAG Simply concatenates Latent Document z to Input x**

Experiments

- RAG can be applied to **any task with input and output sequences.**

- Focus on tasks with a clear need for precisely accessing knowledge

Open-domain QA:

Natural Questions, TriviaQA, WebQuestions, CuratedTREC

Abstractive open-domain QA:

"Open" MS MARCO

Question Generation:

Jeopardy questions

Fact Verification:

FEVER

Open–Domain QA

Table 1: Open-Domain QA Test Scores. For TQA, left column uses the standard test set for Open-Domain QA, right column uses the TQA-Wiki test set. See Appendix D for further details.

	Model	NQ	TQA	WQ	CT
Closed Book	T5-11B [52]	34.5	- /50.1	37.4	-
	T5-11B+SSM[52]	36.6	- /60.5	44.7	-
Open Book	REALM [20]	40.4	- / -	40.7	46.8
	DPR [26]	41.5	**57.9**/ -	41.1	50.6
	RAG-Token	44.1	55.2/66.1	**45.5**	50.0
	RAG-Seq.	**44.5**	56.8/**68.0**	45.2	**52.2**

- Strongly outperform “closed-book" models with specialized pretraining
- No span extraction required
- Docs that don’t contain exact answer still contribute to generating correct answer
- Answer questions correctly even when correct answer is not in retrieved docs

Abstractive Open-Domain QA

- Some questions unanswerable without gold passages
- RAG strongly outperforms BART baseline
- . Not so far from SoTA models which use gold passages

Top Retrieved doc:
`A typical apple serving weighs 242 grams and provides 126 calories with a moderate content of dietary fiber (table). Otherwise, there is ... is usually not eaten and is discarded.`

Input: `how many calories in average apple`

BART: `The average apple contains 1,000 calories in an average apple and 1,200 calories in a medium apple`

RAG: `There are 126 calories in apple, while an extra large size apple has 172 calories.`

`GOLD: apple has 80 calories`

Table 2: Generation and classification Test Scores. MS-MARCO SotA is [4], FEVER-3 is [68] and FEVER-2 is [57] *Uses gold context/evidence. Best model without gold access underlined.

Model	Jeopardy B-1	Jeopardy QB-1	MSMARCO R-L	MSMARCO B-1	FVR3 Label Acc.	FVR2 Label Acc.
SotA	-	-	**49.8***	**49.9***	**76.8**	**92.2***
BART	15.1	19.7	38.2	41.6	64.0	81.1
RAG-Tok.	**17.3**	**22.2**	40.1	41.5	72.5	<u>89.5</u>
RAG-Seq.	14.7	21.4	<u>40.8</u>	<u>44.2</u>		

Jeopardy Question Generation

Input: `Washington`

Gold: `Florida's in the southeast corner of the 48 contiguous states; this state is in the northwest corner`

BART: `This state has the largest number of counties in the U.S.`

RAG: `Its the only U.S. state named for a U.S. President`

Input: `The Divine Comedy`

BART: `This epic poem by Dante is divided into three parts: the Inferno, The Purgatorio & the Purgatorio`

RAG: `This 14th Century work is divided into 3 sections: "inferno", "Purgatorio" & "Paradiso"`

Jeopardy Question Generation

- Challenging knowledge intensive generation task
- Unlike other tasks **RAG-Token performs best** here
- Task requires integrating facts from different documents

Table 2: Generation and classification Test Scores. MS-MARCO SotA is [4], FEVER-3 is [68] and FEVER-2 is [57] *Uses gold context/evidence. Best model without gold access underlined.

Model	Jeopardy B-1	Jeopardy QB-1	MSMARCO R-L	MSMARCO B-1	FVR3 Label Acc.	FVR2 Label Acc.
SotA	-	-	**49.8***	**49.9***	**76.8**	**92.2***
BART	15.1	19.7	38.2	41.6	64.0	81.1
RAG-Tok	**17.3**	**22.2**	40.1	41.5	72.5	<u>89.5</u>
RAG-Seq	14.7	21.4	<u>40.8</u>	<u>44.2</u>		

Table 4: Human assessments for the Jeopardy Question Generation Task.

	Factuality	Specificity
BART better	7.1%	16.8%
RAG better	**42.7%**	**37.4%**
Both good	11.7%	11.8%
Both poor	17.7%	6.9%
No majority	20.8%	20.1%

Generation Diversity

Table 5: Ratio of distinct to total tri-grams for generation tasks.

	MSMARCO	Jeopardy QGen
Gold	89.6%	90.0%
BART	70.7%	32.4%
RAG-Token	77.8%	46.8%
RAG-Seq.	83.5%	53.8%

Reinforcement Learning with Human Feedback

Motivation

- LLMs should possess three properties to be applicable in real-world:
- Helpful: should help the user solve their task according to the instructions.
- Honest: should give accurate information;
  - should express uncertainty when the model doesn't know the answer, instead of hallucinating a wrong answer.
- Harmless: should not cause physical, psychological, or social harm to people or the environment.

Motivation (cont.)

- Misalignment: When the training objective does not capture the desiderata we want from models
- Predicting the next token on a webpage from the internet—is different from the objective “follow the user’s instructions helpfully and safely”

$$p(x) = \prod_{i=1}^{n} p(s_n | s_1, ..., s_{n-1})$$

Training: Predict the next token

$\not\Rightarrow$

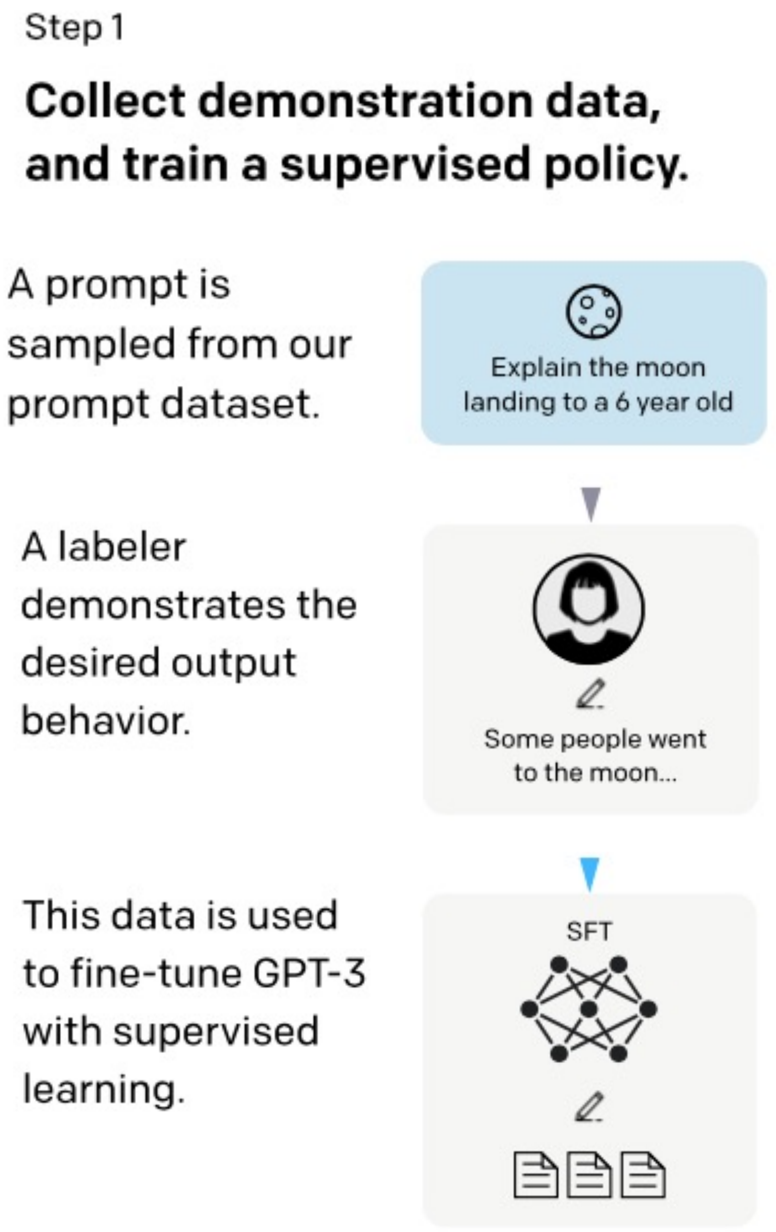
The three H’s of Model Desiderata

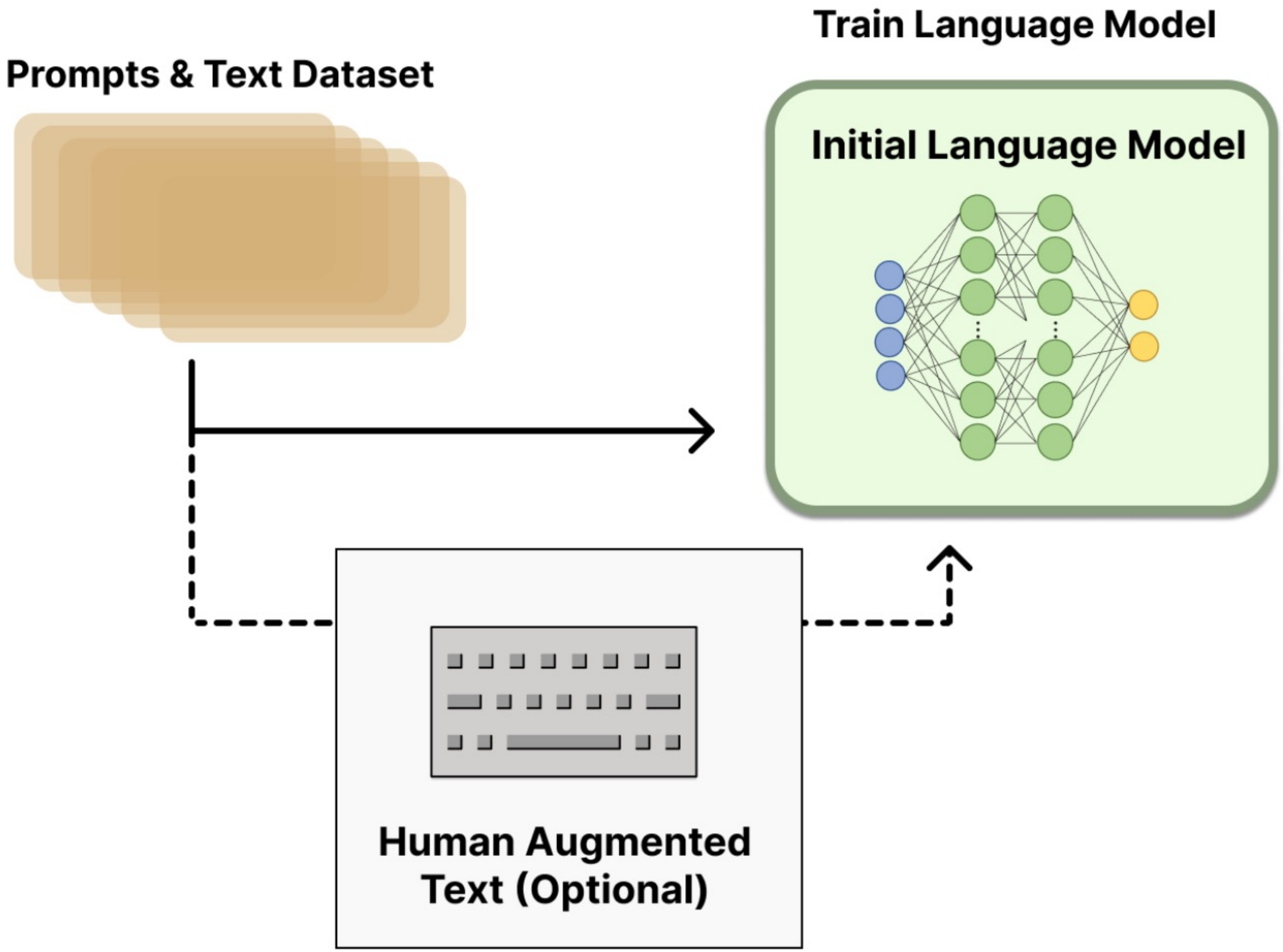
How to go about this?

- Modifying the loss?
- Supervised instruction tuning?
- Use reward signal and Reinforcement Learning to fine tune the model.

Reinforcement Learning with Human Feedback (RLHF) : Step 1

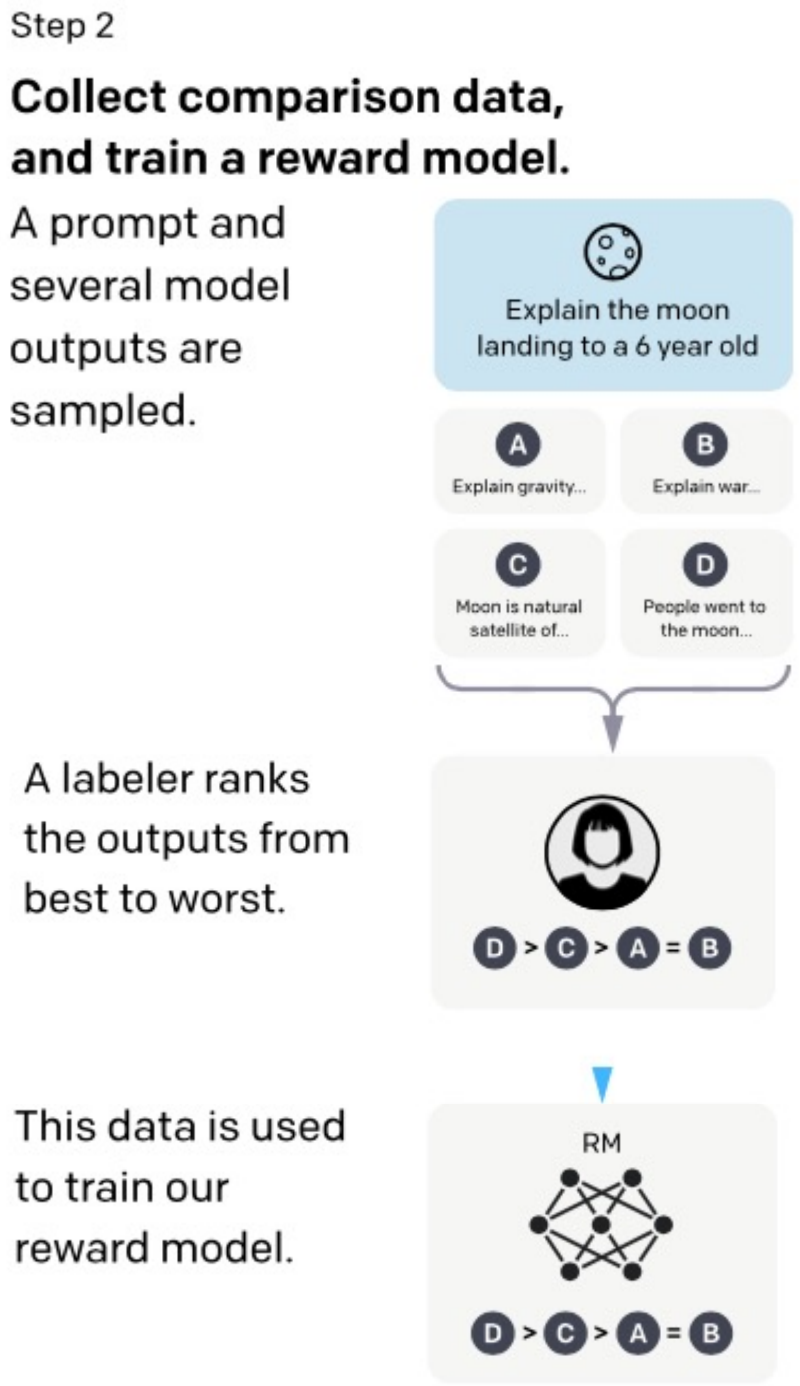
- Need a good policy to start with.
- Supervised training with a set of instructions is a good start.
- Essentially the same idea as FLAN and T0.
  - What's the difference here?

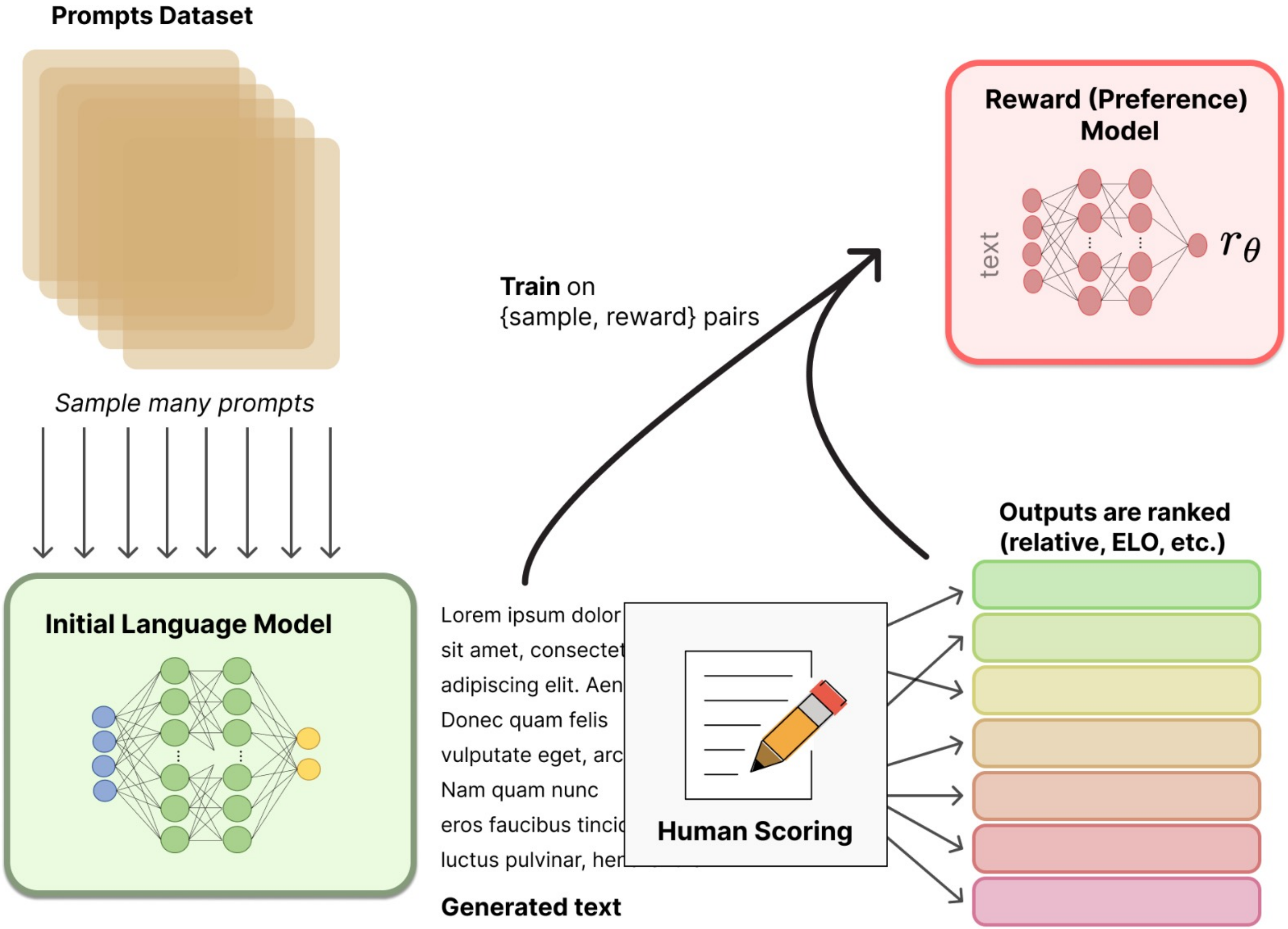



Train Language Model
Prompts & Text Dataset
Initial Language Model
Human Augmented
Text (Optional)

RLHF : Step 2

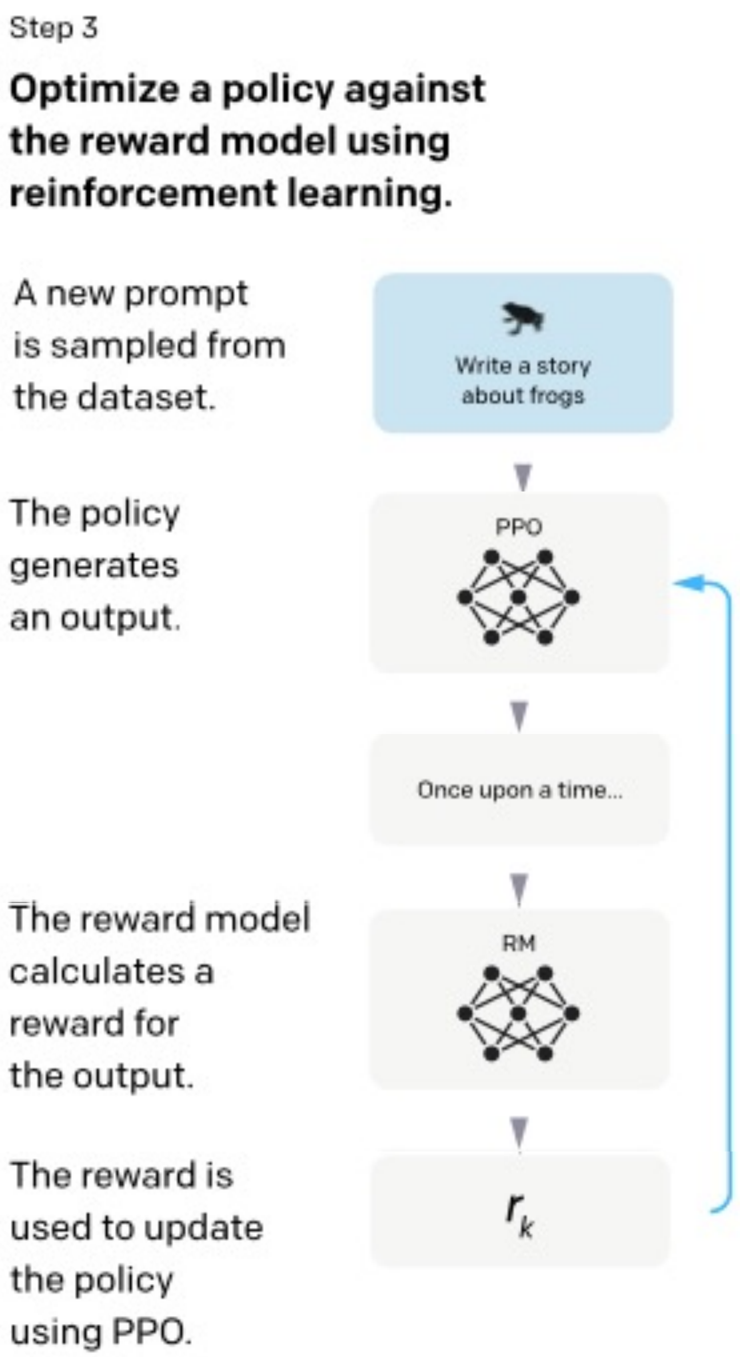
- Need a reward function in order to be able run RL.
- Is the previous data format (instruction, answer) sufficient?
- Need scored data: (instruction, answer, score)
- What are the challenges of score?

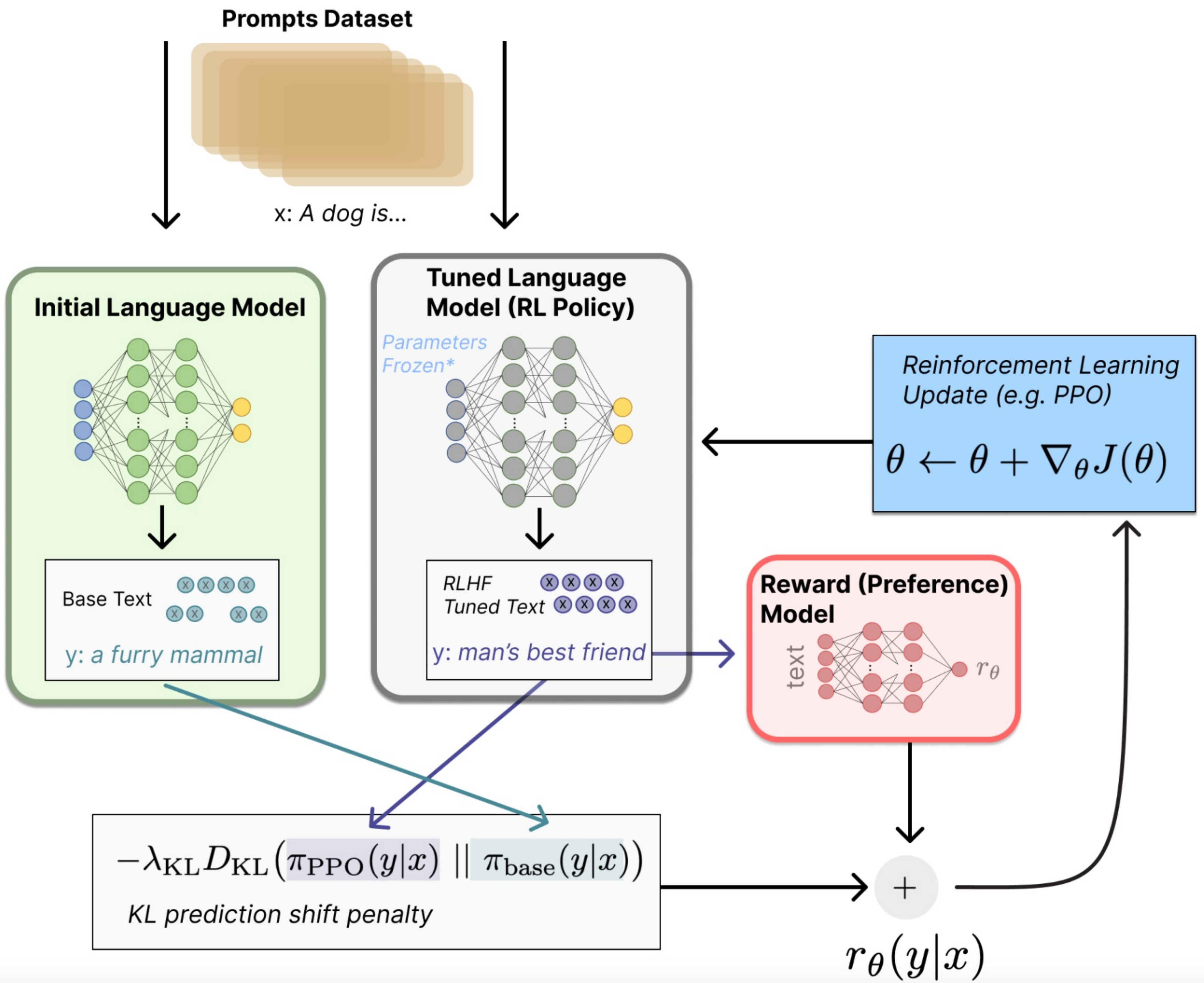



Prompts Dataset
Sample many prompts
Initial Language Model
Train on
{sample, reward} pairs
Reward (Preference) Model
text
r_θ
Lorem ipsum dolor
sit amet, consectet
adipiscing elit. Aen
Donec quam felis
vulputate eget, arc
Nam quam nunc
eros faucibus tincid
luctus pulvinar, her
Generated text
Human Scoring
Outputs are ranked (relative, ELO, etc.)

RLHF : Step 3

- Proximal Policy Optimization (PPO) is applied.
- It is an on-policy RL algorithm
  - The policy that is optimized is the same as the policy that is used to gather the data.
- The core idea: In improving the policy based on the current data, do NOT change the policy overly. Why?




Prompts Dataset
x: A dog is...
Initial Language Model
Tuned Language Model (RL Policy)
Parameters Frozen*
Base Text
y: a furry mammal
RLHF Tuned Text
y: man's best friend
Reinforcement Learning Update (e.g. PPO)
$\theta \leftarrow \theta + \nabla_\theta J(\theta)$
Reward (Preference) Model
text
r_θ
$-\lambda_{KL} D_{KL}(\pi_{PPO}(y|x) \,||\, \pi_{base}(y|x))$
KL prediction shift penalty
+
$r_\theta(y|x)$

Details of Step 1

- Text prompts submitted to the OpenAI API of earlier InstructGPT.
- Deduplicate prompts (long common prefix).
- < 200 prompts per user ID.
- A team of 40 labelers provided desired demonstrations (outputs).
- Train/val./test splits are based on the user ID.

Details of Step 2

- Start from the SFT model, removed the last unembedding layer.
- Takes in the (prompt, response); outputs: scalar reward
- 6B model is fine, and is more stable
- K = 4 to K = 9 possible responses are given to the labelers.
  - Multiple model outputs constitute the responses
- The data is converted to $\binom{K}{2}$ pairwise samples.
- All such comparisons are provided in a single batch. Why?
  - Avoids overfitting.
  - Computationally more efficient.

Details of Step 2 (cont.)

- The loss function:

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

- x = input prompt ; y_l : worse response ; y_w : better response
- 33k training prompts

Number of Prompts

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

Details of Step 3

- Bandit problem: single step episodes.
- 31k prompts for training.
- Potential danger: over-optimization or the reward model.
  - Let's discuss why.
- Penalize the model for drifting from the SFT model:

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

- Mixing pretraining gradient with PPO. Why?
  - Avoid ruining the performance on public NLP datasets.
  - PPO-ptx