Reinforcement learning
and
Large Language Models

Lize Pirenne (lize.pirenne@uliege.be)
May 7, 2024

Outline

Large Language Models

Reinforcement Learning

Learning from humans

RL Methods for LLMs

Current Challenges

Inspired by EWRL: RL & Languages, Olivier Pietquin

Notations

In this course, we use the classic reinforcement learning notations:

- $s \in \mathcal{S}$ for the states (instead of $x \in \mathcal{X}$),
- $a \in \mathcal{A}$ for the actions (instead of $u \in \mathcal{U}$),
- $V(s)$ for the state value function (instead of $J(s)$),
- $Q(s, a)$ for the state-action value function,
- $\pi(a|s)$ for the stationary stochastic policy,

In addition, we use the following abbreviations:

- MDP: Markov decision process
- (L)LM: (Large) language model

Large Language Models

Language Models

Let w_i denote the i-th word in a sentence. A language model M estimates the probability of the next word w_{i+1} given the previous words $w_1, \ldots, w_i$.

Causal language modelling

$$P(w_{i+1}|w_1, \ldots, w_i) \approx M(w_1, \ldots, w_i)$$

This probability can be learnt in an unsupervised manner; there is no need to label the data.

M uses production rules (Context-Free Grammar), n-grams (MDP), or neural networks (Recurrent, Graph or Transformer-based) as the underlying mechanism.

Decoder only architecture

Based on the Transformer architecture of Vaswani et al., 2017, the decoder-only architecture is the most common for generative language models.

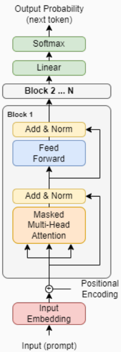

Figure 1: Decoder-only architecture[1].

[1] Image source

Tokens

Language models operate on tokens rather than words in order to compress the size of the input. All possible tokens constitute the vocabulary $\mathcal{V}$.

A common tokenization method is byte-pair encoding (BPE) which, starting from all possible bytes, merges the most frequent pairs and adds the whole to the vocabulary until a desired size is reached.

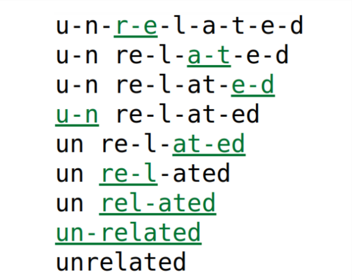

Figure 2: Byte-pair encoding (BPE) vocabulary construction Provilkov, Emelianenko, and Voita, 2019.

Generation

The model produces for each token a distribution over the vocabulary that represents the probability of appearance of the next token.

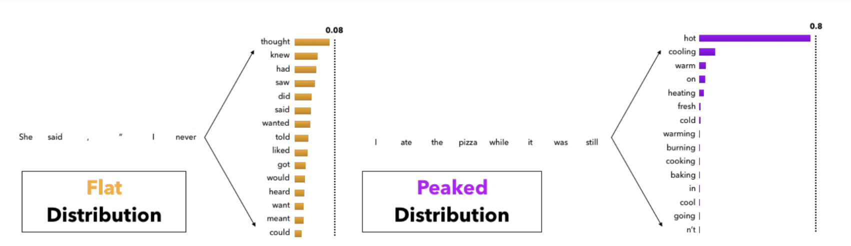

Figure 3: Token distributions [2]

[2] Holtzman et al., 2020

Sampling

By repeatedly adding back the most likely token (greedy decoding), the model generates a sentence in an auto-regressive manner.

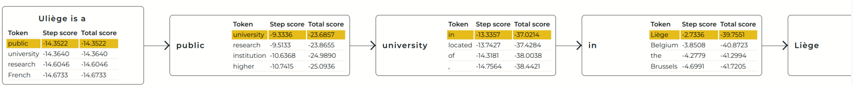

Figure 4: Greedy decoding [3]

By sampling from the probability instead of taking the most likely every step, the auto-regressive generation can be non-deterministic.

[3] m-ric/beam_search_visualizer

Decoding

A decoding strategy chooses which token to pick next to form a likely sentence.

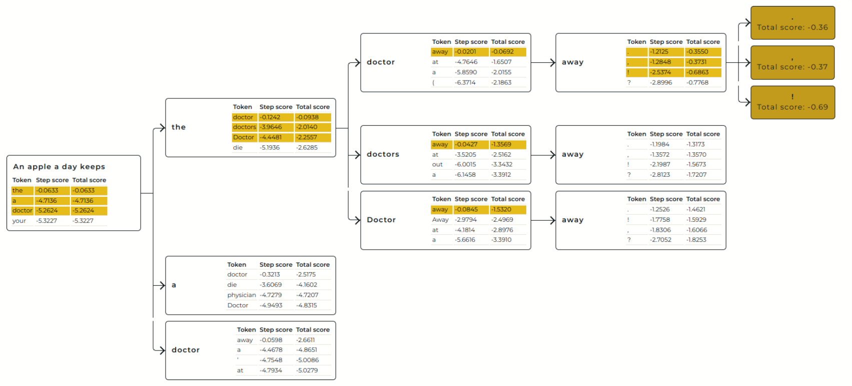

Figure 5: Beam decoding [4]

[4] m-ric/beam_search_visualizer

More than words

Transformers (Vaswani et al., 2017), the architecture behind most LLMs today, work for any sequential data, meaning they can use and produce code, images, sounds, ...

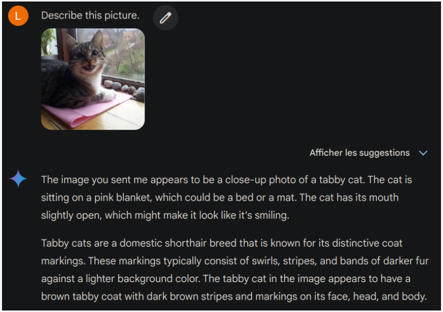

Figure 6: Example of a multi-modal interaction with Gemini[5]

[5] Team et al., 2023, cat picture (not original)

Multi-modality in practice

Use a specific encoder and a pre-trained model and learn the projector linking the two.

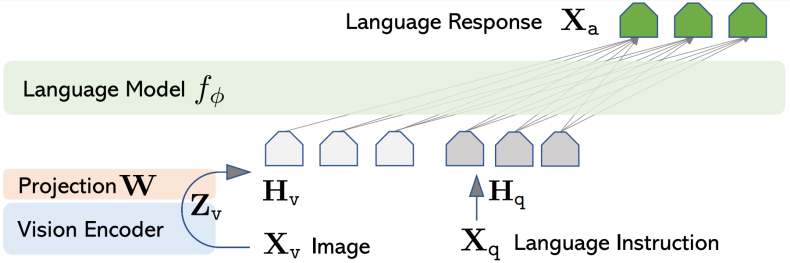

Figure 7: Multi-modal generation encoders [6]

[6] LLaVa: Liu, C. Li, Wu, et al., 2023; Liu, C. Li, Y. Li, et al., 2023

Reinforcement Learning

In reinforcement learning, an agent interacts with an environment by taking actions a_t in states s_t according to a policy π. The goal is to find the optimal policy π^* that maximizes the return of a reward function $R(s, a)$, which is the expected return $V(s)$.

Optimal policy

$$V_\pi(s) = \underset{s_{t+1} \sim p(.|a_t, s_t)}{E} \left(\sum_t \gamma^t R(s_t, \pi(a_t|s_t)) | s_0 = s \right)$$
$$\pi^*(s) = \arg\max_\pi \; V_\pi(s)$$

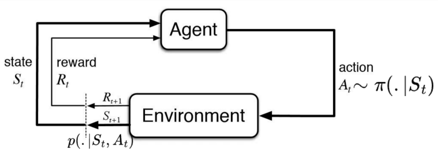

Figure 8: Agent / Environment interaction loop.

Value-based methods

Value-based methods define the value function $Q(s, a)$ as the expected cumulative reward from taking action a in state s and then following the optimal policy according to the value function.

Policy construction

$$Q(s, a) = \underset{s_{t+1} \sim p(.|a_t, s_t)}{E} \left(R(s_t, a_t) + \gamma \underset{b \in A}{max} Q(s_{t+1}, b) | s_0 = s, a_0 = a \right)$$

$$\pi^*(s) = \arg\max_b \; Q(s, b)$$

While a bit faster thanks to bootstrapping (use estimates), value-based methods can be biased and offer only an indirect access to the policy.

Policy-based methods

Policy-based methods directly optimize the policy $\pi(a|s)$ by maximizing the expected cumulative reward using a gradient based approach.

Policy construction

$$\pi^k(s) = \pi^{k-1}(s) + \alpha \frac{\delta V_\pi(s)}{\delta \pi}$$

Policy-based methods often use Monte-Carlo (use only observations) and thus are unbiased and offer a direct access to the policy, but they can be slow and high-variance.

Choice of approach

Due to the large action space, taking the max function becomes dangerous: the differences of values become too small compared to the noise.

Policy based methods are also preferred because they are more sample efficient and lead to more stable training.

Learning from humans

Imitation learning[7] is a method to learn a policy π from a set of demonstrations $\mathcal{D}$.

Behavioral cloning

$$\pi^* = \arg\min_{\pi} \sum_{(s,a)\in\mathcal{D}} loss(\pi(a|s), a)$$

The policy is trained to mimic the expert's actions, but it can be brittle (sensitive to the proximity of the training distribution) and biased (expert does not provide π^*).

[7] Useful imitation library

Next token prediction is already behavior cloning with the LLM as the agent. We can draw a parallel between the two fields:

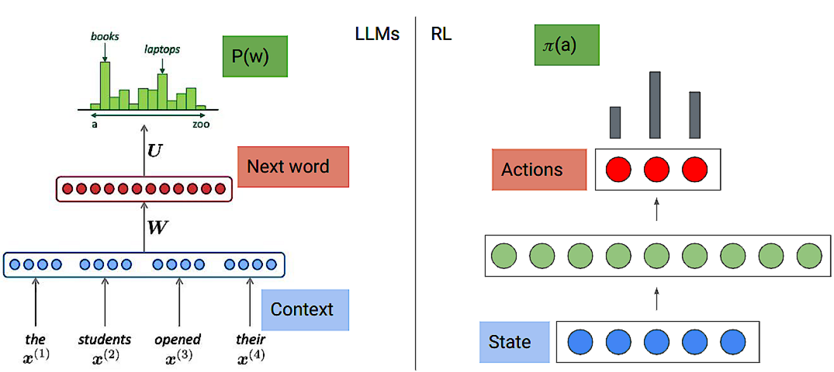

Figure 9: RL and LLM parallel[8]

[8] EWRL: RL & Languages, Olivier Pietquin.

Drifting

Behavior cloning is subject to the open-loop drifting problem: the model accumulates errors over time and diverts too far from the learnt policy.

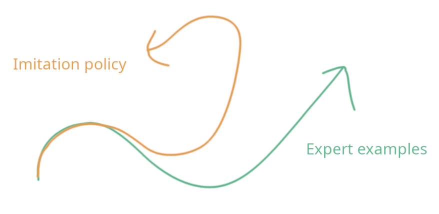

Figure 10: Open-loop drifting problem.

Methods like DAgger[9], which asks experts to annotate some observations, GAIL[10], which discriminates expert and agent trajectories, or IRL, which aims to learn the higher concept of reward, can alleviate this problem.

[9] Ross, Gordon, and Bagnell, 2010
[10] Ho and Ermon, 2016

LLM thus suffer from the same drifting problem, named hallucinations.

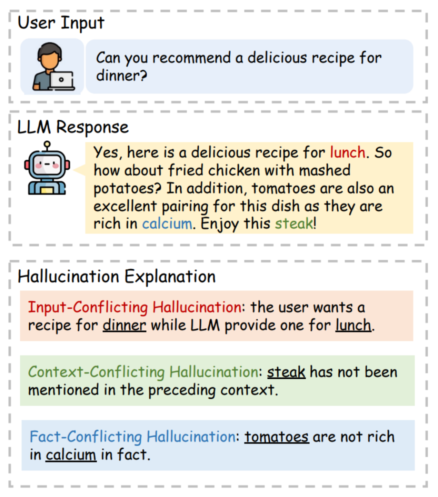

Figure 11: Different hallucinations[11].

[11]Y. Zhang et al., 2023

Short-sightedness

LLMs are able to measure the "quality" of a sentence through the perplexity but they cannot target a specific one.

Perplexity

$$PPL(w_1 : w_N) = exp\left(-\sum_{i}^{t} log\left(p_\theta(w_i | w_{<i})\right)\right)$$

Heuristics help guide the generation based on the distributions.

- Temperature Sampling
  Modifies the distribution ($0 \rightarrow argmax$, $\infty \rightarrow uniform$).
- Beam search
  Explores multiple paths and keeps the best ones.
- Nucleus sampling (top-p)
  Selects tokens until cum-sum p is reached.
- Top-k sampling
  Keeps only the k most likely tokens.

Metrics

Most useful metrics in Natural Language Processing (NLP) are non-differentiable, and thus cannot be used as a loss function.

NLP metrics

- 🔵 BLEU
- 🔴 ROUGE
- 🌠 METEOR
- 🍾 CIDEr
- ...

LLM metrics

- 🤷‍♀️ Truthfulness
- ✅ Factuality
- 🤐 Verbosity
- 🥶 Toxicity
- ⚖️ Neutrality
- 🗣️ Personna
- ...

Benefits of RL

Why use RL in LLMs?

- RL can optimize for any scalar score (even NLP metrics)
- RL can provide the sequence-level optimization that LLMs lack.
- RL improves over behavior cloning

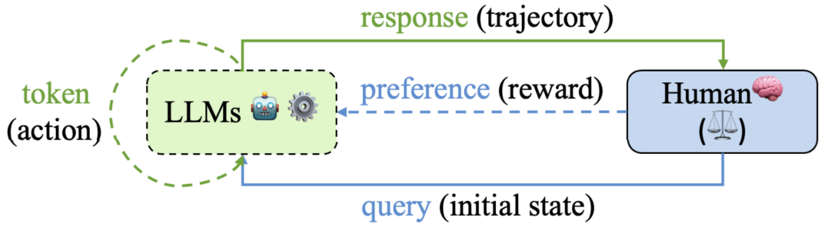

Figure 12: RL for LLMs[12]

[12] H. Sun, 2023

RL Methods for LLMs

Inverse reinforcement learning (IRL) is a method to learn a reward function $R(s, a)$ from a set of demonstrations $\mathcal{D}$. To do this, we learn the vector w in the expression $R(s, a) = w^T \phi(s, a)$, where ϕ is a feature map.

Valid reward function[13]

$$V_\pi(s) = w^T \mu(\pi, s) = w^T \underset{s_{t+1} \sim p(.|a_t, s_t)}{E} \left(\sum_t \gamma^t \phi(s_t, \pi(s_t)) | s_0 = s \right)$$

$$\text{Find } {w^*}^T \text{ satisfying } {w^*}^T \mu(\pi^*, s) >= {w^*}^T \mu(\pi, s)$$

IRL needs an access to the environment, and methods to alleviate the reward ambiguity (existence of trivial solutions)[14].

[13] Ng, Russell, et al., 2000
[14] Stanford.edu

Human feedback

On top of demonstrations, the feedback can take the form of :

- Preferences
  - Ranking
  - Pairwise comparison
    - The sequences y_0, y_1 generated from x are compared by the expert and given a preference index $\mu \in \{0, 1\}$.

Reward assumption under Bradley-Terry model[15]

$$p[y_0 \succ y_1 | x] := \frac{\exp(r(x,y_0))}{\exp(r(x,y_0)) + \exp(r(x,y_1))}$$

Pairwise reward function loss[16]

$$loss(r_\phi) = - \underset{(x,y_0,y_1,\mu)\sim\mathcal{D}}{E} \left[\log \sigma \left(r_\phi(x, y_\mu) - r_\phi(x, y_{1-\mu}) \right)\right]$$

- Rewards
  - Scores
  - Ratings
  - Thumbs up / down

- Advice
  - Corrections

[15] Bradley and Terry, 1952

[16] Christiano et al., 2017; Rafailov et al., 2024

Policy Gradient Theorem

Given a sentence τ, the likelihood of the sentence according to the LLM π_θ being $p_{\pi_\theta}(\tau)$, the expected return is $V_{\pi_\theta} = \int p_{\pi_\theta}(\tau) R(\tau) d\tau$.

$$
\begin{aligned}
\nabla_\theta V_{\pi_\theta} &= \int \nabla_\theta p_{\pi_\theta}(\tau) R(\tau) d\tau \\
&= \int p_{\pi_\theta}(\tau) \frac{\nabla_\theta p_{\pi_\theta}(\tau)}{p_{\pi_\theta}(\tau)} R(\tau) d\tau \\
&= E\left[\frac{\nabla_\theta p_{\pi_\theta}(\tau)}{p_{\pi_\theta}(\tau)} R(\tau) \right] \\
&= E\left[\nabla_\theta \log p_{\pi_\theta}(\tau) R(\tau) \right]
\end{aligned}
$$

Policy Gradient Theorem (cont'd)

We can decompose τ into a sequence of tokens $w_1, \ldots, w_N$ and, since the policy defines w_t given $w_{<t}$, we can write the likelihood of the sentence as follows.

$$p_{\pi_\theta}(\tau) = p(w_1) \prod_{t=2}^{N} \pi_\theta(w_t | w_{<t})$$

The gradient of the log-likelihood is then:

$$\nabla_\theta \log p_{\pi_\theta}(\tau) = \sum_{t=1}^{N} \nabla_\theta \log \pi_\theta(w_t | w_{<t})$$

Policy Gradient Theorem[17]

$$\nabla_\theta V_{\pi_\theta} = E\left[\sum_{t=1}^{N} \nabla_\theta \log \pi_\theta(w_t | w_{<t}) R(\tau)\right]$$

[17] Sutton et al., 1999

Reinforce applied to LLMs

The REINFORCE algorithm[18] uses the policy gradient theorem to update the policy π_θ.

REINFORCE

$$\hat{\nabla}_\theta V_{\pi_\theta} = \frac{1}{D}\sum_{i=1}^{D}\left[\left(\sum_{t=1}^{N}\nabla_\theta \log \pi_\theta(w_t^i|w_{<t}^i)\right)\left(\sum_{t=1}^{N} r_t^i\right)\right]$$

We can use a baseline b to reduce the variance of the estimator.

REINFORCE with baseline

$$\hat{\nabla}_\theta V_{\pi_\theta} = \frac{1}{D}\sum_{i=1}^{D}\left[\left(\sum_{t=1}^{N}\nabla_\theta \log \pi_\theta(w_t^i|w_{<t}^i)\right)\left(\sum_{t=1}^{N} r_t^i - b\right)\right]$$

[18] Williams, 1992

Reinforcement Learning from Human Feedback

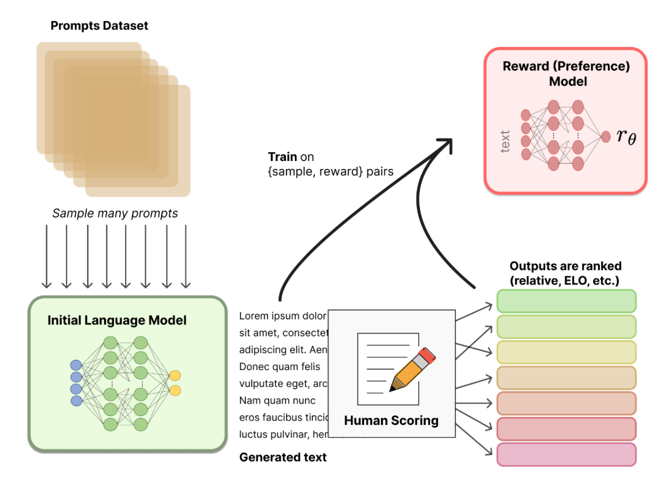

Figure 13: Simple RLHF paradigm [19]

[19] HuggingFace RLHF

Reward hacking

Continued training leads to decrease in performance due to reward hacking: the model finds a way to maximize the reward without actually solving the task.

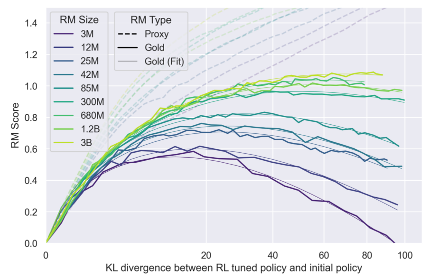

Figure 14: Reward hacking[20]

Adding a Kullback-Leibler divergence (KL) term to the loss function can help alleviate this problem.

[20] Gao, Schulman, and Hilton, 2023

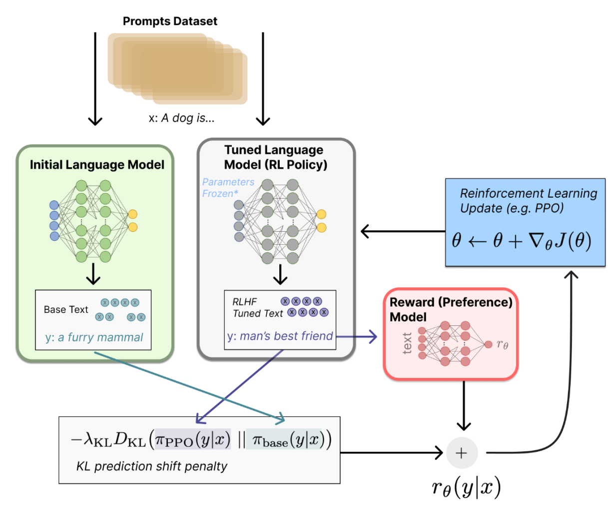

Figure 15: Better RLHF paradigm [21]

[21] HuggingFace RLHF

Examples of RLHF

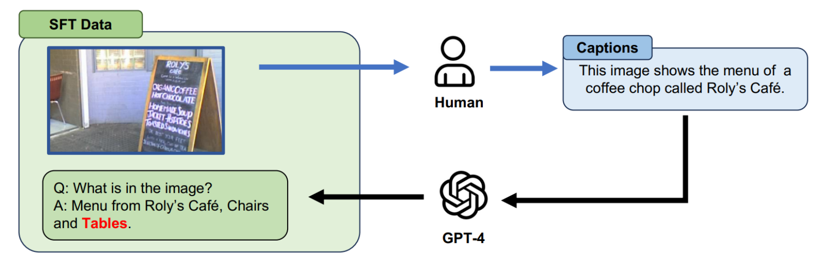

Figure 16: Misaligned Supervised Fine-Tuning (SFT) Data contains Hallucination[22]

[22] Z. Sun et al., 2023

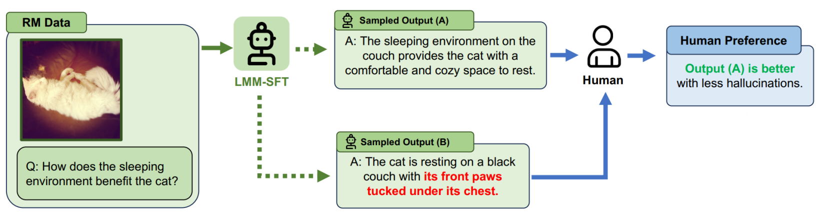

Figure 17: Collect Human Preference (More Helpful & Less Hallucinated) Data for Reward Models (RM)[23]

Train a reward model with pairwise loss

$$loss(r_\phi) = - \underset{(x,\sigma_1,\sigma_2,\mu)\sim\mathcal{D}_{RM}}{E} \left[\log \sigma\left(r_\phi(x, y_\mu) - r_\phi(x, y_{1-\mu})\right)\right]$$

[23] Z. Sun et al., 2023

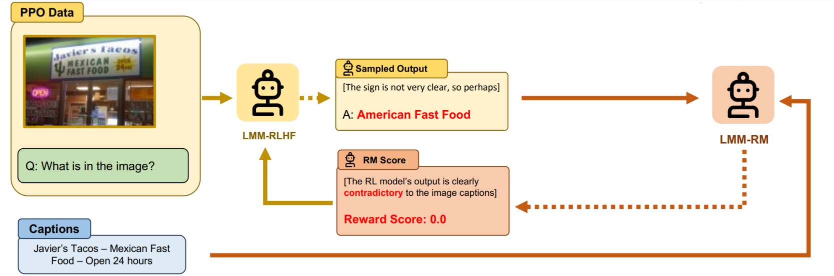

Figure 18: Factually Augmented Reinforcement Learning from Human Feedback (Fact-RLHF)[24]

Train the LLM with the reward model

$$L(\pi_\theta) = -\underset{x\sim\mathcal{D}_{RL}, y\sim\pi_\theta(y|x)}{E}\left[r_\phi(x,y) - \beta\mathbf{D}_{KL}(\,\pi_\theta(y|x)||\pi^{REF}(y|x)\,)\right]$$

[24] Z. Sun et al., 2023

The LLM can be used as its own reward model.

DPO[25]

$$L_{DPO}(\pi_\theta) = -\underset{(x,y_0,y_1,\mu)\sim\mathcal{D}}{E}\left[\log\sigma\left(\beta\log\frac{\pi_\theta(y_\mu|x)}{\pi^{REF}(y_\mu|x)} - \beta\log\frac{\pi_\theta(y_{1-\mu}|x)}{\pi^{REF}(y_{1-\mu}|x)}\right)\right]$$

We can add the assumption that y_{win} is ideal rather than simply better by adding a regularization term, granted λ large enough.

DPO-Positive [26]

$$L_{DPO}(\pi_\theta) = -\underset{(x,y_0,y_1,\mu)\sim\mathcal{D}}{E}\left[\ldots - \lambda\max\left(\log\frac{\pi^{REF}(y_\mu|x)}{\pi_\theta(y_\mu|x)}, 0\right)\right]$$

[25] Rafailov et al., 2024
[26] Z. Sun et al., 2023

By doing successive rounds of training, we can vastly improve the performance of the model.

Algorithm `DNO-Prct`: Practical Implementation of `DNO` via Iterative Contrastive Self-Improvement

input: General preference function $\mathcal{P}$, learning rate $\widetilde{\eta}$, iterations T, reference policy π_{ref}, prompt distribution ρ.

Initialize $\pi_1 \leftarrow \pi_{\text{ref}}$.
for iteration $t = 1, 2, \ldots, T$ **do**
  Construct $\mathcal{D}_t = \{(x, y^{\text{gold}})\}$ where $x \sim \rho$ and $y \sim \pi_{\text{gold}}(\cdot \mid x)$.
  Sample *batched on-policy* responses: Sample K outputs per prompt using the current π_t: $\{y_t^1, y_t^2, \ldots, y_t^K\} \sim \pi_t(\cdot \mid x), \forall x \in \mathcal{D}_t$.
  Rank responses: For each $x \in \mathcal{D}_t$, rank the corresponding $\{y_t^1, y_t^2, \ldots, y_t^K, y^{\text{gold}}\}$ using the pair-wise win-rate by sampling from the general preference function $\mathcal{P}$.
  Filter preference pairs: Construct $\mathcal{D}_{t+1} = \{(x, y_t^+, y_t^-)\}$, for all $x \in \mathcal{D}_{t+1}$, and (y_t^+, y_t^-) are large-margin pairs (based on the win-rate rank) within the responses for x from the previous step.
  Contrastive learning: Obtain π_{t+1} by,

$$\pi_{t+1} \leftarrow \underset{\pi \in \Pi}{\operatorname{argmax}} \, \mathbb{E}_{(x, y_t^+, y_t^-) \sim \mathcal{D}_{t+1}} \log \left[\sigma \left(\widetilde{\eta} \log \frac{\pi(y_t^+ \mid x)}{\pi_t(y_t^+ \mid x)} - \widetilde{\eta} \log \frac{\pi(y_t^- \mid x)}{\pi_t(y_t^- \mid x)} \right) \right].$$

end for
return best of $\pi_{1:(T+1)}$ on the validation data.

Figure 19: Iterative DPO[27]

[27] Rosset et al., 2024

Current Challenges

Task complexity evaluation

Choosing the right LLM for the right task while keeping the cost reasonable is difficult.

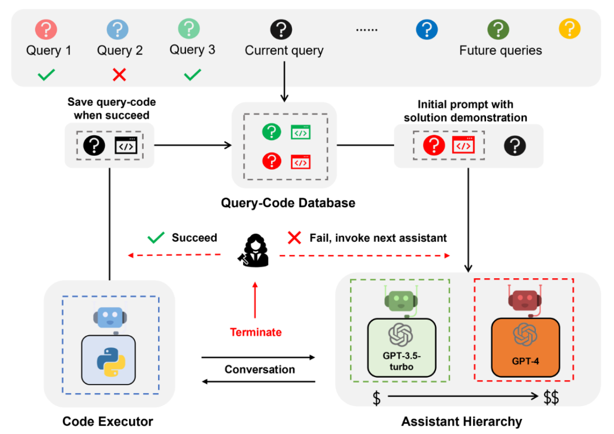

Figure 20: Eco-Assistant framework[28]

[28] J. Zhang et al., 2023

Tool usage

LLMs can learn to use tools but choosing the right one is a challenge.

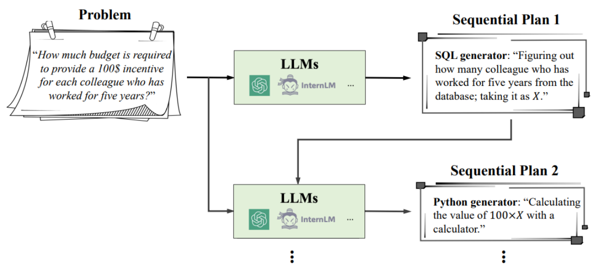

Figure 21: Tool usage[29]

[29]Ruan et al., 2023

Multi-Step Planning

The decoding heuristics are not enough to guide the generation towards a specific goal.

Task	Instances correct	
	GPT-4	I-GPT3
Plan Generation We showcase an instance and the respective plan as an example and prompt the machine with a new instance.	206/600 (34.3%)	41/600 (6.8%)
Cost-Optimal Planning We showcase an instance, the respective optimal plan and the associated cost as an example and prompt the machine with a new instance.	198/600 (33%)	35/600 (5.8%)

Figure 22: LLMs still can't plan[30]

[30]Valmeekam et al., 2022

And many more...

- Life long learning
  Gather continual feedback, avoid forgetting, ...
- Personalization
  Adapt to the user, to a certain task, ...
- Evaluation
  How to evaluate the quality of the generated text?
- Efficient data usage
  Learn on its own (goal of RL), reuse data, compress LLMs, ...

Take-home message

You
Give me the take home message of the presentation

ChatGPT
The key take-home message of the presentation is that while Large Language Models (LLMs) have revolutionized natural language processing, integrating reinforcement learning (RL) techniques can significantly enhance their capabilities. By incorporating RL, LLMs can optimize for specific objectives, learn from human feedback, and mitigate issues such as hallucinations and short-sightedness. RL techniques like Reinforce, Inverse Reinforcement Learning (IRL), and Reinforcement Learning from Human Feedback (RLHF) offer powerful tools for training LLMs to better understand context, generate more accurate and contextually appropriate responses, and ultimately improve their overall performance in various tasks. However, challenges such as task complexity evaluation, tool usage, and multi-step planning remain, indicating the need for further research and development in this area.

Figure 23: Message by ChatGPT[31]

Use it for what it's good at.

[31] chatgpt.com

Thank you!

Questions?

References

Bradley, Ralph Allan and Milton E Terry (1952). **"Rank analysis of incomplete block designs: I. The method of paired comparisons".** In: *Biometrika* 39.3/4, pp. 324–345.

Christiano, Paul F, Jan Leike, Tom Brown, et al. (2017). **"Deep reinforcement learning from human preferences".** In: *Advances in neural information processing systems* 30.

Gao, Leo, John Schulman, and Jacob Hilton (2023). **"Scaling laws for reward model overoptimization".** In: *International Conference on Machine Learning*. PMLR, pp. 10835–10866.

Ho, Jonathan and Stefano Ermon (2016). **"Generative adversarial imitation learning".** In: *Advances in neural information processing systems* 29.

Holtzman, Ari, Jan Buys, Li Du, et al. (2020). **"The Curious Case of Neural Text Degeneration".** In: *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020.* OpenReview.net. URL: https://openreview.net/forum?id=rygGQyrFvH.

Liu, Haotian, Chunyuan Li, Yuheng Li, et al. (2023). **"Improved Baselines with Visual Instruction Tuning".** In: *arXiv preprint arXiv: 2310.03744.*

Liu, Haotian, Chunyuan Li, Qingyang Wu, et al. (2023). **"Visual Instruction Tuning".** In: *Neural Information Processing Systems*. DOI: 10.48550/arXiv.2304.08485.

Ng, Andrew Y, Stuart Russell, et al. (2000). **"Algorithms for inverse reinforcement learning.".** In: *Icml*. Vol. 1. 2, p. 2.

Provilkov, Ivan, Dmitrii Emelianenko, and Elena Voita (2019). **"BPE-dropout: Simple and effective subword regularization".** In: *arXiv preprint arXiv:1910.13267*.

Rafailov, Rafael, Archit Sharma, Eric Mitchell, et al. (2024). **"Direct preference optimization: Your language model is secretly a reward model".** In: *Advances in Neural Information Processing Systems* 36.

Ross, Stephane, Geoffrey J. Gordon, and J. Andrew Bagnell (2010). **"A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning".** In: *arXiv preprint arXiv: 1011.0686*.

Rosset, Corby, Ching-An Cheng, Arindam Mitra, et al. (2024). **"Direct Nash Optimization: Teaching Language Models to Self-Improve with General Preferences".** In: *arXiv preprint arXiv: 2404.03715*.

Ruan, Jingqing, Yihong Chen, Bin Zhang, et al. (2023). **"TPTU: Large Language Model-based AI Agents for Task Planning and Tool Usage".** In: *arXiv preprint arXiv: 2308.03427.*

Sun, Hao (2023). **"Reinforcement Learning in the Era of LLMs: What is Essential? What is needed? An RL Perspective on RLHF, Prompting, and Beyond".** In: *arXiv preprint arXiv: 2310.06147.*

Sun, Zhiqing, Sheng Shen, Shengcao Cao, et al. (2023). **"Aligning Large Multimodal Models with Factually Augmented RLHF".** In: *arXiv preprint arXiv: 2309.14525.*

Sutton, Richard S, David McAllester, Satinder Singh, et al. (1999). **"Policy gradient methods for reinforcement learning with function approximation".** In: *Advances in neural information processing systems* 12.

Team, Gemini, Rohan Anil, Sebastian Borgeaud, et al. (Dec. 2023). ***Gemini: A Family of Highly Capable Multimodal Models.*** eprint: 2312.11805 (cs). (Visited on 12/20/2023).

Valmeekam, Karthik, Alberto Olmo, S. Sreedharan, et al. (2022). **"PlanBench: An Extensible Benchmark for Evaluating Large Language Models on Planning and Reasoning about Change".** In: *Neural Information Processing Systems.*

Vaswani, Ashish, Noam Shazeer, Niki Parmar, et al. (2017). **“Attention is all you need”.** In: *Advances in neural information processing systems* 30.

Williams, Ronald J (1992). **“Simple statistical gradient-following algorithms for connectionist reinforcement learning”.** In: *Machine learning* 8, pp. 229–256.

Zhang, Jieyu, Ranjay Krishna, Ahmed H. Awadallah, et al. (2023). **“EcoAssistant: Using LLM Assistant More Affordably and Accurately”.** In: *arXiv preprint arXiv: 2310.03046.*

Zhang, Yue, Yafu Li, Leyang Cui, et al. (2023). **“Siren’s Song in the AI Ocean: A Survey on Hallucination in Large Language Models”.** In: *arXiv preprint arXiv: 2309.01219.*