Self Correction in LLMs

Self Correction in LLMs

As part of our aim to understand how self correction in LLMs work, we first decided to gain some knowledge about the problem statement and existing methods.

With that in mind, we found and read the following paper -

Automatically Correcting Large Language Models: Surveying the landscape of diverse self-correction strategies

Liangming Pan, Michael Saxon, Wenda Xu, Deepak Nathani, Xinyi Wang, William Yang Wang

Agenda

- An issue with LLMs
- Taxonomy for correcting LLMs with feedback
- Overview of different methods corresponding to when correction is applied
- Applications
- Future plans for RE

An Issue with LLMs

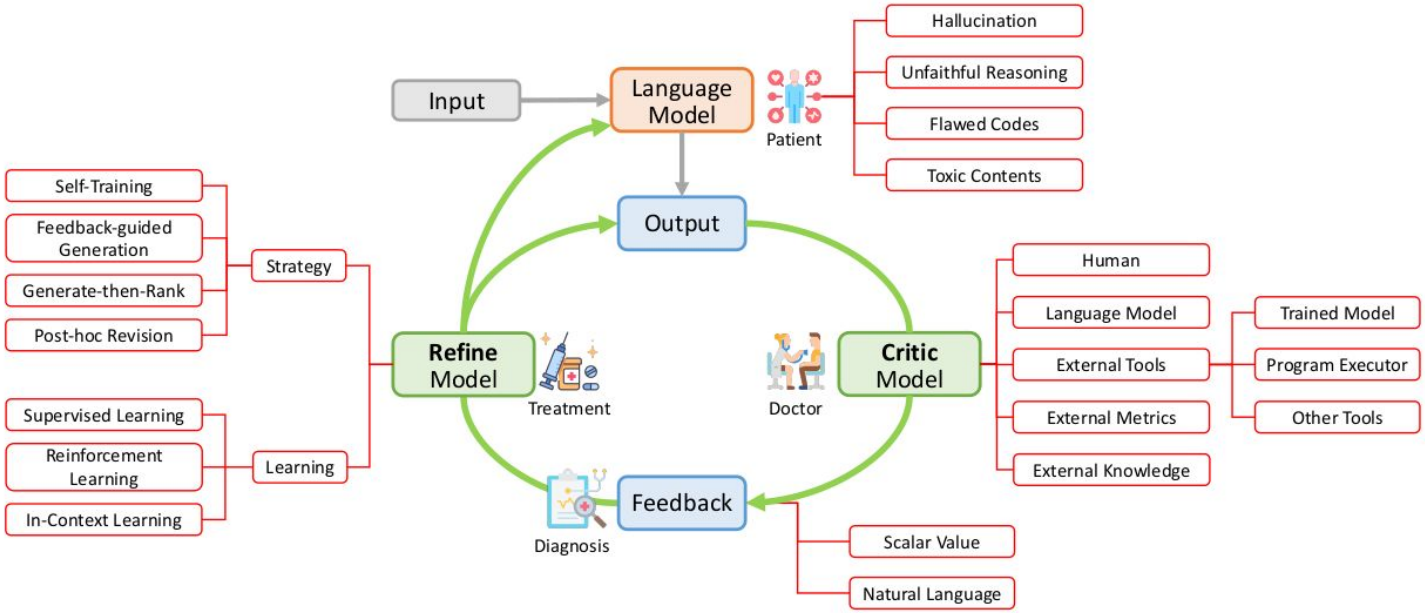
There are some cases where LLMs have provided wrong outputs-

- Seemingly accurate but inaccurate hallucinations.
- Conducting unfaithful reasoning
- Generating harmful content
- Failing to understand and follow rules and constraints.

This questions the amount of trust that can be put in the output coming from an LLM.

Taxonomy for Correcting LLMs with Feedback

Conceptual framework



Taxonomy for Correcting LLMs with Feedback

---

5 Primary Classification criteria for Existing Works:-

- What gets Corrected.
- Source of the Feedback.
- Format of the Feedback.
- When the Feedback is used.
- How to Correct the Model with Feedback.

What gets corrected

The method of self correction generally deals with overcoming these shortcomings in LLMs:

- Hallucination
- Unfaithful reasoning
- Toxic, Biased, and Harmful contents
- Flawed code

What is the Source of feedback

- Human Feedback
- Automated Feedback
  - Self feedback
  - External feedback

Format of the Feedback

- Scalar Value Feedback
- Natural Language Feedback

When/How to correct the model with feedback

- Training-time correction
- Generation-time correction
- Post-hoc correction

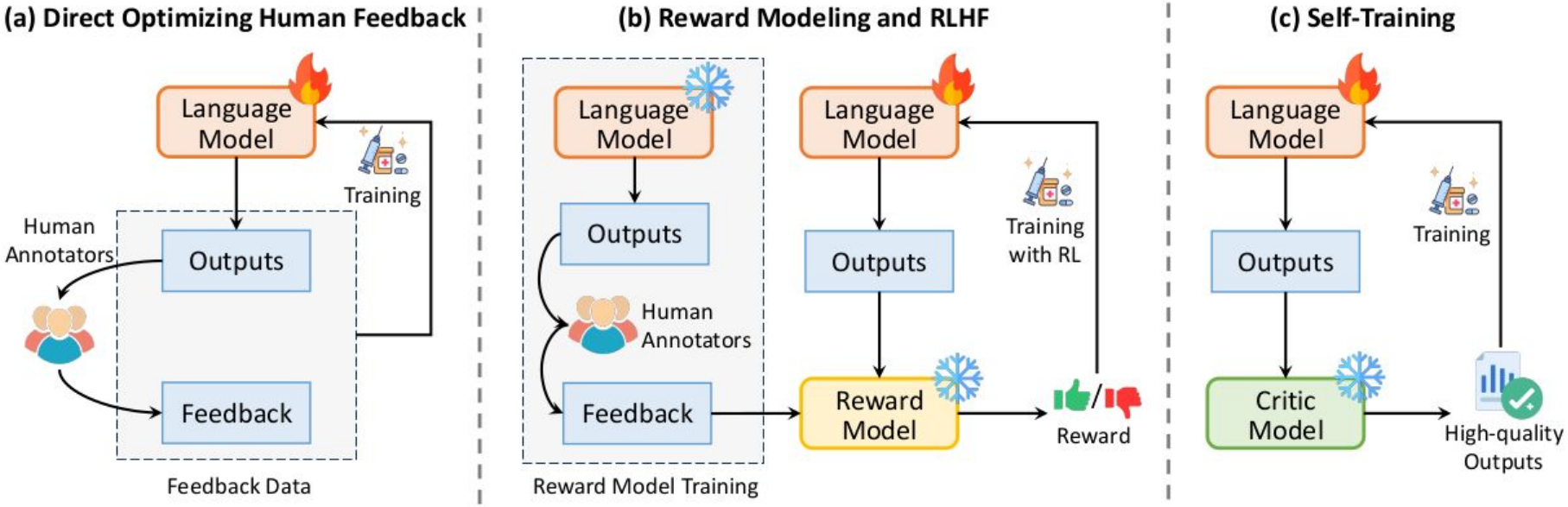
Training-Time Correction

- Exploration of methods that seek to correct the LLM during the training phase of the model.
- Tweaks the model parameters in the training phase to give the best possible output for a task.

Training-time Correction

- Learning from Human feedback
  - Direct Optimization with human feedback
- Reward Modelling and RLHF
- Learning with Automated feedback
  - External Metric Guidance
  - Self-training

Diagrammatic representation of the methods -



Training-time Correction

Drawbacks of training-time correction:

- Infeasibility of fine-tuning giant closed-source LLMs, such as GPT-4
- The Potential unavailability of feedback during model training
- Requirement for the feedback to be optimizable

Generation-Time Correction

- Training-time correction would not be able to generalise well to unseen cases.
- Exploration of methods that seek to correct the LLM during the generation of the output.

Generation-time Correction

- Generate-then-rank
- Feedback guided decoding
  - Reward model from human feedback
  - Training verifier with synthetic data
  - Feedback from external metric
  - Self-evaluation

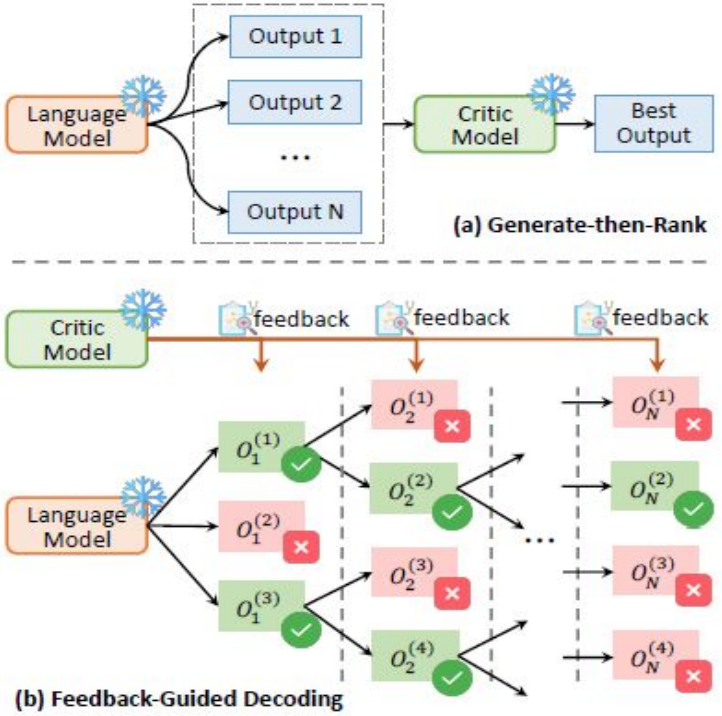


Figure 3: The illustrations of the two typical strategies of *generation-time correction*: (a) Generate-then-Rank, and (b) Feedback-Guided Decoding.

Post-Hoc Correction

- Exploration of methods that correct the LLM by intervening only after the output is generated.
- The feedback can be used to locate the exact point of the occurrence of an error.

Post-Hoc Correction - Methods

- Self-correction - The use of one large-scale LLM to generate output and provide feedback.
- Correction with External tools providing feedback
  - Code Interpreter
  - Logic Reasoner
  - External Knowledge
  - Trained Model
  - Integrating Multiple tools
- Multi-agent Debate - The use of multiple LLMs to “discuss” and arrive to a conclusion.

Diagrammatic Representation of the Methods

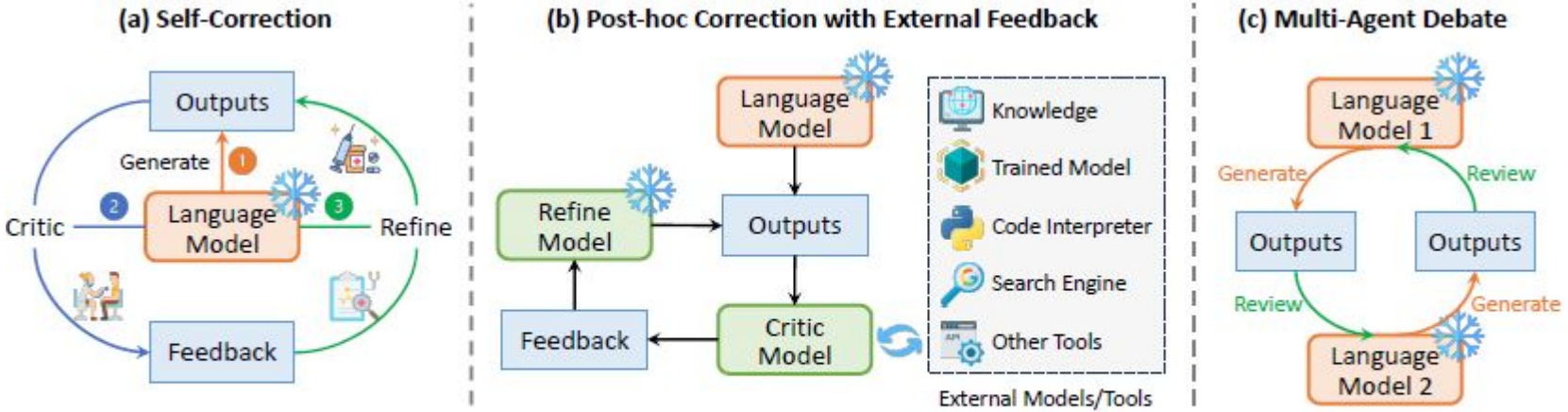


Figure 4: Three typical strategies of *post-hoc correction*: self-correction (a), post-hoc correction with external feedback (b), and multi-agent debate (c).

General Applications -

- Factual Correction - Use of multi-agent debate models like LM vs LM.
- Reasoning Tasks - A question of “how to find the right prompt which would enable the LLM to identify and correct intermediate steps?”
- Code Synthesis - The use of external tool based feedback. Performance is enhanced when there is an expert in the loop.

Future Plan

- Look into, in depth, some of the methods mentioned above like self-correcting and multi-agent debate.
- Study prompt engineering to get a better idea of how it helps critic models provide feedback.

Thank You