Genetic Pareto Prompt Optimizer

Notes taken by D. Gueorguiev, 7/3/2025

GEPA stands for **Ge**netic-**Pa**reto Prompt Optimizer, a prompt optimizer that incorporates natural language reflection to learn high level rules from trial and error.

References

[1] [GEPA: Reflective Prompt Evolution can Outperform Reinforcement Learning, Lakshya A. Agrawal et al, 2025](https://github.com/dimitarpg13/large_language_models/blob/main/articles/GePa/GEPA-Reflective_Prompt_Evolution_Can_Outperform_Reinforcement_Learning_Agrawal_Berkeley_2025.pdf)

[2] [Reflection Agents, LangChain blog, Feb 21, 2024](https://blog.langchain.com/reflection-agents/)

[3] [Reflection, Prompt Engineering Guide, online article at promptingguide.ai](https://www.promptingguide.ai/techniques/reflexion)

[4] [Thinking Fast and Slow, Daniel Kahneman, 2011](https://github.com/dimitarpg13/aiconcepts/blob/master/literature/kahneman/ThinkingFastAndSlow.pdf)

[5] [Reflection: Language Agents with Verbal Reinforcement Learning, Noah Shinn et al, 2023](https://github.com/dimitarpg13/large_language_models/blob/main/articles/Reflexion/Reflexion-Language_Agents_with_Verbal_Reinforcement_Learning_Shinn_2023.pdf)

[6] [LLMs can learn self-restraint through iterative self-reflection, Alexandre Piche et al, 2024](https://github.com/dimitarpg13/large_language_models/blob/main/articles/Reflexion/LLMs_can_learn_self-restraint_through_iterative_self-reflection_Piche_2024.pdf)

[7] [Self-Reflection in LLM Agents: Effects on Problem-Solving Performance, Matthew Renze et al, 2024](https://github.com/dimitarpg13/large_language_models/blob/main/articles/Reflexion/Self-Reflection_in_LLM_Agents-Effects_on_Problem-Solving_Performance_Renze_2024.pdf)

[8] [LangGraph: reflection tutorial on langchain-ai.github.io](https://langchain-ai.github.io/langgraph/tutorials/reflection/reflection/)

[9] [Policy Gradient Method, Wiki](https://en.wikipedia.org/wiki/Policy_gradient_method)

[10] [DSPy: Compiling Declarative Language Model Calls into Self-Improving Pipelines, O. Khattab et al, 2023](https://github.com/dimitarpg13/large_language_models/blob/main/articles/DSPy/DSPy-Compiling_Declarative_Language_Model_Calls_into_Self-Improving_Pipelines_Khattab_2023.pdf)

[11] [DSPy Assertions: Computational Constraints for Self-Refining Language Model Pipelines, A. Singhvi et al, 2023](https://github.com/dimitarpg13/large_language_models/blob/main/articles/DSPy/DSPy_Assertions-Computational_Constraints_for_Self-Refining_Language_Model_Pipelines_Singhvi_2024.pdf)

[12] [ReAct: Synergizing Reasoning and Acting in Language Models, S. Yao et al, Princeton U., 2023](https://github.com/dimitarpg13/large_language_models/blob/main/articles/ReAct/ReAct-Synergizing_Reasoning_and_Acting_in_Language_Models_Yao_2023.pdf)

[13] [Toolformer: Language Models Can Teach Themselves to Use Tools, Timo Schick et al, 2023](https://github.com/dimitarpg13/large_language_models/blob/main/articles/Toolformer/Toolformer-Language_Models_Can_Teach_Themselves_to_Use_Tools_Schick_2023.pdf)

Appendix

Reflection in LLM-based frameworks

The paper by Noah Shinn et al [5] introduces the notion of Reflection in LLM context.

A recent framework such as DSPy ([10], [11]), ReAct ([12]), Toolformer ([13]) have demonstrated the feasibility of autonomous decision-making agents built on top of LLM instances. These frameworks use LLMs to generate text and actions that can be used in API calls and executed in a suitable environment. Most of these frameworks are limited to in-context learning examples for teaching the agents since RL-based training schemes require substantial amount of compute and time.

The alternative to in-context learning is *Reflexion* and it uses verbal reinforcement to help agents learn from prior negative experiences. Reflexion converts numerical feedback from the environment into verbal feedback in the form of textual representation (summary) which is then added as an additional context to the LLM agent for the next episode.

This self-reflective

Reflection in DSPy

DSPy with self-reflection: Enabling smarter and more reliable language model pipelines

DSPy is a powerful framework that allows developers to programmatically build and optimize pipelines that leverage Large Language Models (LLMs). Its goal is to move beyond brittle prompt engineering and towards a more robust and systematic approach to LLM-powered applications. One key way DSPy achieves this is through the integration of **LM Assertions**, which enable self-reflection and self-correction within LLM pipelines.

How DSPy uses assertions for self-reflection

* **Assertions as constraints:** LM Assertions act as programmable elements that specify conditions or rules the LLM outputs must satisfy.
* **Two types of assertions:**
  + **Hard Assertions:** Represent critical conditions. If violated even after retries, they halt the pipeline, indicating a non-negotiable breach of requirements.
  + **Soft Assertions (Suggestions):** Denote desirable properties. Violation triggers self-refinement and retry mechanisms, but doesn't necessarily halt the pipeline if violated after retries.
* **Self-refinement through backtracking:** When an assertion fails, DSPy can backtrack and retry the failing module. It incorporates the error message and the violating output into the new prompt, enabling the LLM to self-reflect and refine its generation based on the feedback.

Example of DSPy self-reflection in action

Consider a multi-hop question-answering task using a retriever. DSPy can use assertions to guide the generation of queries to the retriever:

* **Suggestion 1:** dspy.Suggest(len(query) < 100, "Query should be less than 100 characters")
* **Suggestion 2:** dspy.Suggest(is\_query\_distinct(query, queries), f"Query should be distinct from {queries}")

If the second suggestion fails, DSPy will construct a new prompt for the generate\_query module, highlighting the previous query and the error message to help the LM generate a distinct query.

Benefits of using DSPy with self-reflection

* **Increased Reliability:** Ensures LLM outputs adhere to specific guidelines and constraints, making the pipeline more robust.
* **Improved Accuracy:** Self-refinement and backtracking allow the LLM to learn from its mistakes and generate more accurate outputs.
* **Simplified Debugging:** Provides clearer insights into LLM behavior and helps identify areas for improvement within complex pipelines.
* **Reduced Manual Prompt Engineering:** Automates the optimization and refinement of prompts, allowing developers to focus on higher-level logic.

Key takeaway

DSPy's integration of LM Assertions and self-reflection mechanisms marks a significant step towards building more intelligent and dependable LLM-powered applications. By allowing LLMs to monitor their own outputs and refine their behavior, DSPy empowers developers to build more reliable and accurate AI systems.

//TODO: finish the section on Reflection in LLM

A Refresher on Policy Gradient Methods

//TODO: refresher on PGM