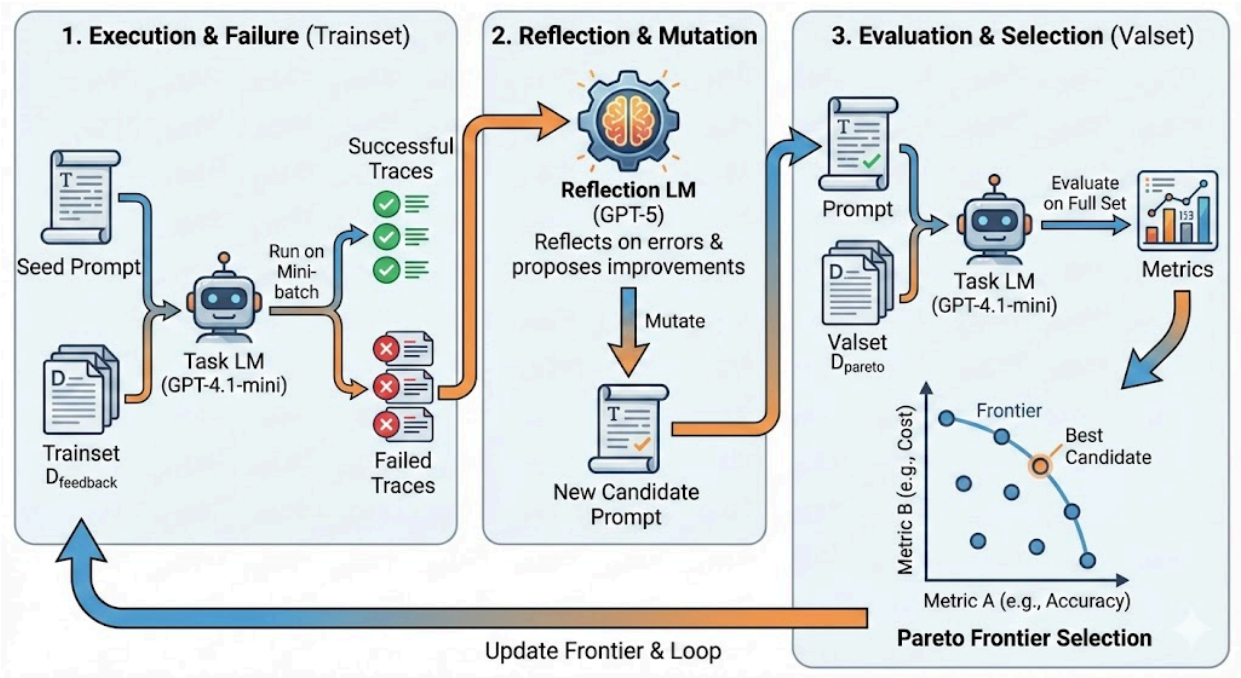
GEPA: Genetic-Pareto Optimization

GEPA is a sophisticated prompt optimization algorithm (popularized by the **DSPy** framework) that treats prompt engineering as an evolutionary process. Instead of simply picking the "best" prompt, it maintains a diverse pool of high-performing candidates.

1. The Process Diagram



2. Core Components

A. The Two Models

- **Task LM (e.g., GPT-4.1-mini):** The model you are actually trying to optimize. It performs the work and its performance is what the metric measures.
- **Reflection LM (e.g., GPT-5):** The "Intelligence" behind the optimization. It looks at errors made by the Task LM and suggests new instructions to fix them.

B. The Two Datasets

- **Trainset ($D_{feedback}$):** Used to generate "traces" (execution logs). When the Task LM fails here, those logs are sent to the Reflection LM to prompt a **Mutation**.
- **Valset (D_{pareto}):** Used to evaluate every new candidate prompt. The results here determine if a prompt moves onto the **Pareto Frontier**.

3. How It Evolves

The optimization loop follows three main genetic operations:

1. Mutation (Learning from Failure)

The Reflection LM looks at a "minibatch" of failures from the Task LM. It reflects on *why* the model failed and generates a new prompt intended to prevent those specific errors.

2. Crossover (Combining Strengths)

GEPA identifies two "parents" from the Pareto Frontier—prompts that are successful in different ways. The Reflection LM then synthesizes a new "child" prompt that inherits the logical structure of one and the formatting precision of the other.

3. Pareto Frontier Selection

Instead of keeping only the single prompt with the highest average score, GEPA keeps a **Frontier** of non-dominated candidates.

- **What is a Frontier?** It is a set of prompts where no prompt is strictly better than another.
- **Why use it?** It preserves "specialists." If one prompt is 100% accurate on "Hard Logic" but 20% on "Creative Writing," it stays on the frontier alongside a prompt that is the opposite. This diversity prevents the algorithm from getting stuck in a mediocre "average" solution.

4. Key Parameters

- **max_metric_calls:** The total budget for Task LM executions. Each time a prompt is tested against an example, it counts as one call.
- **reflection_minibatch_size:** How many failed examples the Reflection LM looks at before it tries to "mutate" a new prompt.
- **max_bootstrapped_demos:** If using Few-Shot prompting, this controls how many successful examples GEPA can "harvest" from its runs to include in the final prompt.

5. Summary of the Approach

GEPA is essentially a **Self-Correction Loop**. It automates the "Trial and Error" that human prompt engineers do manually. By using a strong model (GPT-5) to reflect on the mistakes of a faster/cheaper model (GPT-4.1-mini), it scales high-quality prompt engineering without requiring human intervention for every iteration.