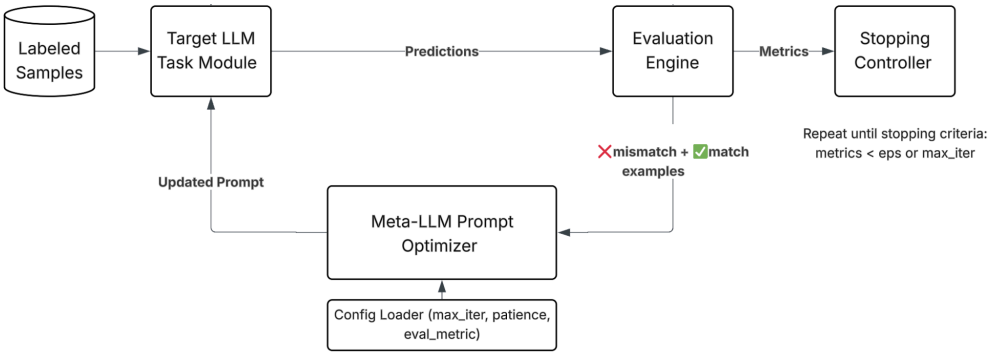
Auto-Prompt-Tuner: A Framework for LLM-driven Prompt Optimization

The performance of Large Language Models (LLMs) is highly sensitive to small changes in prompts, making **manual prompt engineering an unreliable and difficult-to-reproduce process**. This ad-hoc approach often fails on edge cases and lacks a systematic methodology. To address this, we introduce the Auto-Prompt-Tuner, a framework that automates prompt optimization using a data-driven, meta-LLM feedback loop.

The Auto-Tuner Framework



Our framework iteratively refines prompts by using a powerful "meta-LLM" to analyze the performance of a "target LLM" on a given task.

Core Process: The framework operates in a continuous loop:

1. The target LLM makes predictions on labeled data using the current prompt.
2. An Evaluation Engine assesses these predictions, identifying successful matches and mismatches.
3. A Meta-LLM Prompt Optimizer receives this feedback and generates an improved prompt.

The process repeats with the updated prompt.

Convergence: The loop terminates when performance stops improving by a predefined threshold or a maximum number of iterations is reached.

Case Study: Complex Name Matching

We applied the Auto-Prompt-Tuner to a challenging name-matching task, which is complicated by issues like poor image quality, OCR errors, nicknames, and cultural name variations.

Results: The framework automatically evolved the prompt over several iterations, achieving a 12% absolute increase in accuracy, from a baseline of 68% to a peak of 80%. In the production setting, this led to a 10% coverage gain for automatic approval, improving onboarding time.

Non-Linear Optimization: The path to improvement was not straightforward. An early iteration that added case-insensitive matching initially lowered accuracy from 68% to 64%. However, the framework self-corrected in the next step by introducing Levenshtein fuzzy matching, demonstrating its ability to navigate temporary setbacks to find a better overall solution.

Iteration	Accuracy	Change Introduced by Meta-LLM
0	0.68	Baseline: Strict first and last name matching.
1	0.64	Added case-insensitive matching, which hurt precision.
2	0.72	Introduced Levenshtein fuzzy matching to handle typos.
3	0.74	Added logic for nicknames and initials (e.g., "J. Smith" ↔ "John Smith").
4	0.70	Handled suffixes and placeholder names (e.g., "FNU","LNU").
5	0.80	Added rules for multi-token first names and compound surnames.

Table 1: Summary of Iterative Prompt Evolution and Performance

The optimization process shows a clear pattern: initial iterations involve larger, more exploratory edits to the prompt, while later stages consist of smaller, fine-tuning adjustments. This dynamic suggests that a systematic, automated approach can effectively discover complex logic that manual tuning might miss, ultimately leading to more robust and accurate prompts.