GEPA: Optimización Automática de Prompts con Reflexión

Cómo un LLM 'profesor' mejora automáticamente tus prompts mediante análisis de errores

DEMO TÉCNICA

REFLEXIO LAB

El Desafío: Optimizar Prompts es Lento y Manual

Antes de GEPA

- Escribes un prompt genérico basado en intuición
- Pruebas manualmente con casos aislados
- Ves errores, ajustas por prueba y error
- Repites durante horas para mejoras marginales
- Resultado depende completamente de tu experiencia

El proceso es artesanal, subjetivo y no escalable.

Con GEPA

- Defines prompt inicial y casos de prueba
- GEPA ejecuta y analiza errores automáticamente
- LLM "profesor" genera variantes optimizadas
- Búsqueda evolutiva identifica el mejor candidato
- Resultado: prompt especializado en minutos

El proceso es sistemático, medible y reproducible.

Qué Hace GEPA: Test-Driven Development para Prompts

01

Ejecutar

El prompt actual procesa micro-lotes de 3 casos para observar errores y aciertos. Evaluación ágil que permite iteración ultra-rápida sin consumir todo el dataset.

02

Reflexionar

El LLM Profesor analiza los errores detectados, razona sobre las causas raíz y propone mejoras específicas. Ejemplo: "El prompt olvidó mencionar límites monetarios, debería enfatizar precisión numérica".

03

Evolucionar

Genera variantes mejoradas del prompt, las evalúa sistemáticamente y mantiene las mejores. Se repite hasta alcanzar el score objetivo definido.

Búsqueda evolutiva Pareto-eficiente: mantiene candidatos complementarios, no solo el mejor

El Cambio de Paradigma: De la Artesanía a la Ingeniería

GEPA es TDD para Prompts

1

El Problema Actual

Vibe-based prompting: escribimos, probamos un caso, y "creemos" que funciona. Sin métricas, sin validación sistemática.

2

La Solución GEPA

Aplicamos **Test-Driven Development** al lenguaje natural. Definimos tests, medimos métricas, iteramos automáticamente.

3

El Resultado

Profesionalización del desarrollo de IA Generativa con procesos reproducibles y resultados medibles.

GEPA no solo valida el prompt; **lo escribe por ti** hasta que los tests pasan en verde

TDD Clásico vs. Flujo GEPA

La analogía perfecta entre desarrollo de software y prompts

Fase TDD (Software)	Fase GEPA (Prompt Engineering)
1. Write Test	Definir Dataset de Entrenamiento y Validación (Inputs + Salidas Esperadas)
2. Red (Falla)	Baseline Run (Prompt inicial con baja precisión, ej: 40%)
3. Code & Refactor	Optimización Automática (Modelo Profesor reescribe basándose en errores)
4. Green (Pasa)	Prompt Optimizado (Alcanza métrica objetivo, ej: >90%)
5. Integration Test	Prueba de Robustez (Validar generalización en Test Set)

El Superpoder

En TDD tradicional, **tú** escribes el código. En GEPA, **la IA** analiza el error y reescribe el prompt. Es TDD Autónomo.

Estrategia Profesor-Estudiante

El secreto: desacoplar inteligencia de inferencia

1

Fase 1: Optimización

Modelo PROFESOR (GPT-4o): Alta capacidad analítica para analizar errores y redactar prompt maestro. Inversión inicial controlada que se realiza una sola vez.

2

Resultado Intermedio

Se genera un **Prompt Especializado** que captura la inteligencia del modelo profesor en forma de instrucciones precisas.

3

Fase 2: Producción

Modelo EFICIENTE (GPT-4.1-mini): Ejecuta miles de llamadas con el prompt optimizado. Calidad de nivel GPT-4o al costo de GPT-4.1-mini

Reducción del 90% en costos de producción

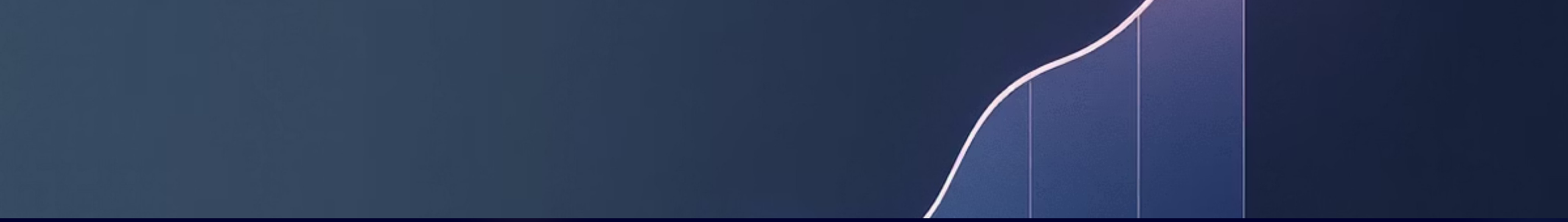
Inteligencia de GPT-4o al costo de GPT-4.1-mini

Resultados Reales: Impacto Medido en el Laboratorio

Mejoras consistentes con estrategia estudiante GPT-4.1-mini

Caso de Uso	Mejora	Detalle
Email Urgency	**+38%**	59.1% → 85.5% precisión
CV Extraction	**+24%**	60.0% → 84.3% precisión
Text-to-SQL	**+41%**	32.5% → 59.2% precisión
RAG Optimization	**+42%**	51.7% → 85.2% precisión

Mejoras consistentes de **+25% a +42%** en todos los casos sin cambiar de modelo, solo optimizando el prompt.



ROI: Inversión Mínima, Retorno Masivo

Análisis de costo-beneficio con datos reales

$0.12

Costo Total de Optimización

Inversión única: 510 llamadas Task Model ($0.04) + 25 llamadas Reflection Model ($0.08)

101

Punto de Equilibrio

Llamadas necesarias para recuperar la inversión inicial

91,950%

ROI a 100K llamadas

Ahorro de $109.88 sobre inversión de $0.12

Ejemplo: 1,000 Llamadas

Sin GEPA: Modelo potente (GPT-4o) = $1.25

Con GEPA: Optimización ($0.12) + Mini ($0.15) = $0.27

Ahorro: $0.98 USD (78% reducción)

ROI: 820% de retorno

Escalabilidad del Ahorro

500 llamadas: ROI 360%

5,000 llamadas: ROI 4,502%

50,000 llamadas: ROI 45,925%

A mayor escala, mayor ahorro proporcional.

Arquitectura del Sistema

Componentes y flujo de datos de GEPA

Dataset de Entrenamiento, Validación, Test(Robustez)

Casos con inputs y salidas esperadas en formato CSV o JSON. Define el criterio de éxito objetivo.

Configuración YAML

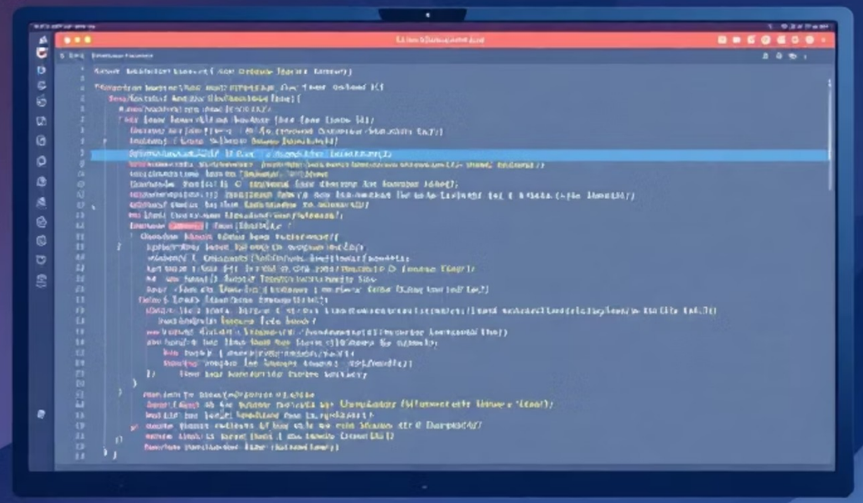
Define modelos (profesor/estudiante), métricas, límites de iteración y parámetros de búsqueda evolutiva.

Motor de Optimización GEPA

Ejecuta ciclos de evaluación, reflexión y evolución. Mantiene leaderboard de candidatos Pareto-óptimos.

Artefactos de Salida

Prompts inicial/final, resultados JSON, métricas CSV y reportes de ROI y performance.



Demo en Vivo: Qué Veremos

01

Ejecutar Optimización

Comando: `python universal_optimizer.py --config experiments/configs/email_urgency.yaml --verbose`

Modo Verbose: Veremos en tiempo real cómo el "Profesor" analiza errores. Tiempo: ~3-5 minutos.

02

Comparar Antes vs Después

Prompt inicial genérico vs prompt optimizado especializado. Comparación de scores: Baseline vs Optimized vs Test.

03

Revisar Artefactos

Entrada: `email_urgency.csv`, `email_urgency_v1.json`

Salida: `initial_prompt.txt`, `final_prompt.txt`, `results.json`, `metricas_optimizacion.csv`

04

Visualizar Leaderboard y ROI

Comando: `python utils/leaderboard.py`

Gráficos: `gepa_roi_analysis.png` y `gepa_performance_improvement.png` con punto de equilibrio y proyección de ahorros.

¿Ahora vemos GEPA en acció