 OPTIMIZACIÓN DE AGENTES RAG

De Funcional a Productivo

- Hace una semana construimos un agente RAG que FUNCIONA
- Hoy lo convertiremos en una herramienta PROFESIONAL
- Enfoque: Prompts, Parámetros, Costos
- Resultado: Mejor calidad, 70% menos dinero

📍 Recapitulación: Semana 1

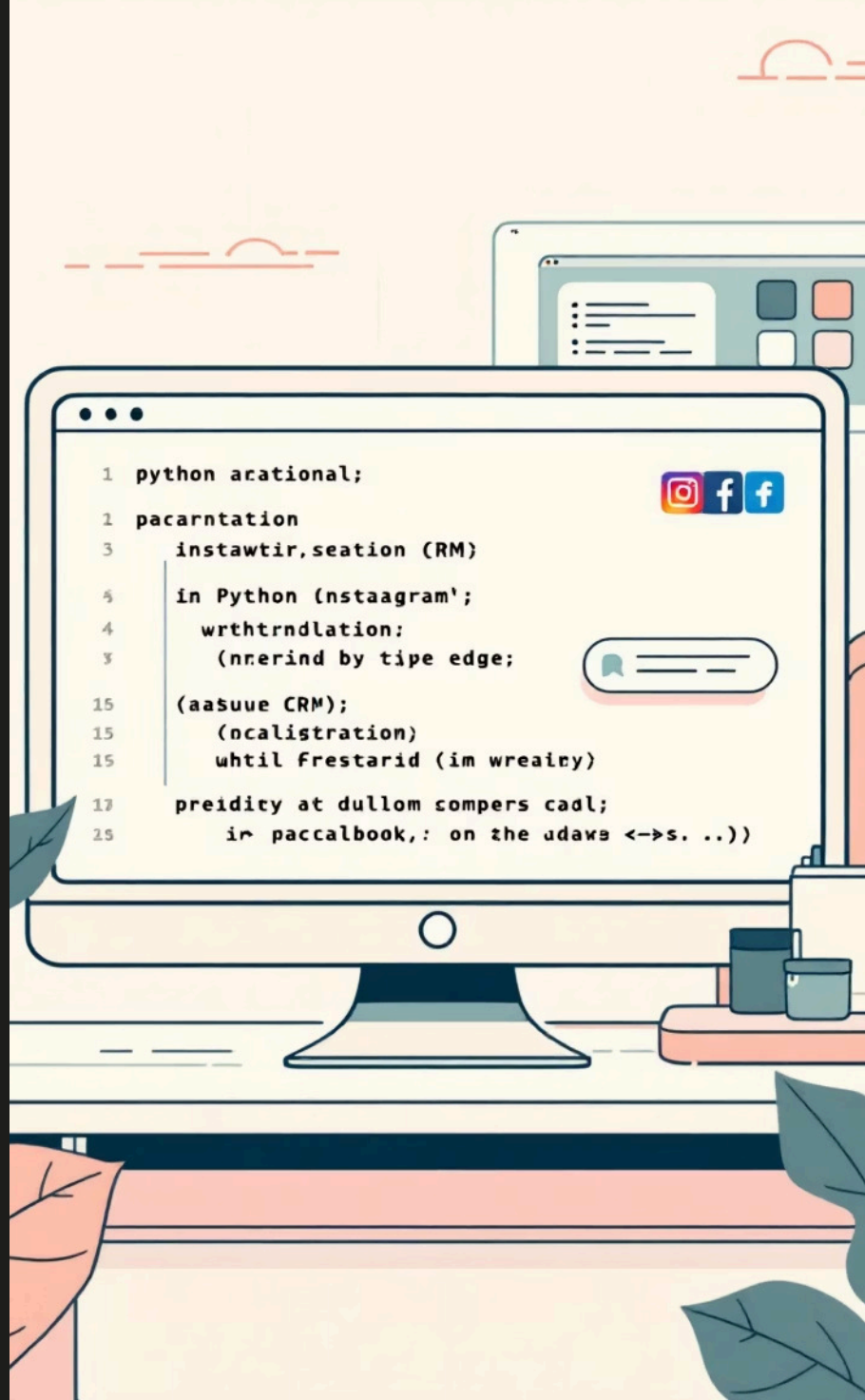
Semana 1:

- Aprendimos qué es RAG, vimos demo en vivo
- Construimos agente con: Document Loader + Splitter + Embeddings + FAISS + RetrievalQA
- Agente está FUNCIONAL: responde preguntas sobre productos Lazarus
- ✅ Sabe de ADMIX, HILTI, proyectos

PERO HAY PROBLEMAS:

- Respuestas a veces genéricas
- Costos no optimizados
- Parámetros por defecto
- ¿Cómo mejorarlo?

HOY RESPONDEMOS: Cómo pasar de funcional a PRODUCTIVO



❓ El Reto de Esta Semana

Tenemos un agente que funciona... pero **¿funciona BIEN?**

1. **¿Es lo suficientemente rápido?**
2. **¿Es lo suficientemente barato?**
3. **¿Responde correctamente siempre?**
4. **¿Respeta los documentos o alucina?**

SEMANA 2 RESPONDE:

Si cambias 3 cosas = Mejor calidad + 70% menos costo

🎯 3 Cambios = Agente Profesional

CAMBIO 1: PROMPTS

- Cómo le preguntas al LLM importa MUCHO
- Prompt malo = respuesta genérica
- Prompt bueno = respuesta específica y verificada
- Técnicas: Few-shot learning, Chain of Thought, restricciones
- **Impacto: +25% calidad sin gastar más**

CAMBIO 2: PARÁMETROS

- Temperature: cuánta creatividad
- Max_tokens: cuánto puede responder
- Top_p: cuánta diversidad
- Ajustar = Respuestas mejor + más rápido
- **Impacto: -40% tokens, -40% tiempo**

CAMBIO 3: RETRIEVER

- K: cuántos documentos consultar (3 vs 5 vs 1)
- Tipo de búsqueda: similarity vs semantic
- Re-ranking: validar resultados
- Ajustar = Búsqueda más eficiente
- **Impacto: Menos datos innecesarios = más rápido**

Flujo RAG

01

Usuario pregunta

02

Buscar documentos relevantes

03

Pasar documentos + pregunta al LLM

04

LLM genera respuesta basada en los documentos

📝 Prompt Engineering: El Arte de Preguntar

PROMPT MALO:

"¿Qué producto me recomiendas?"

→ Resultado: Respuesta genérica

PROMPT BUENO:

"Soy contratista en zona tropical con lluvia extrema. Necesito impermeabilizar cimentación de concreto. Presupuesto máximo L50,000. ¿Cuál es el mejor ADMIX?"

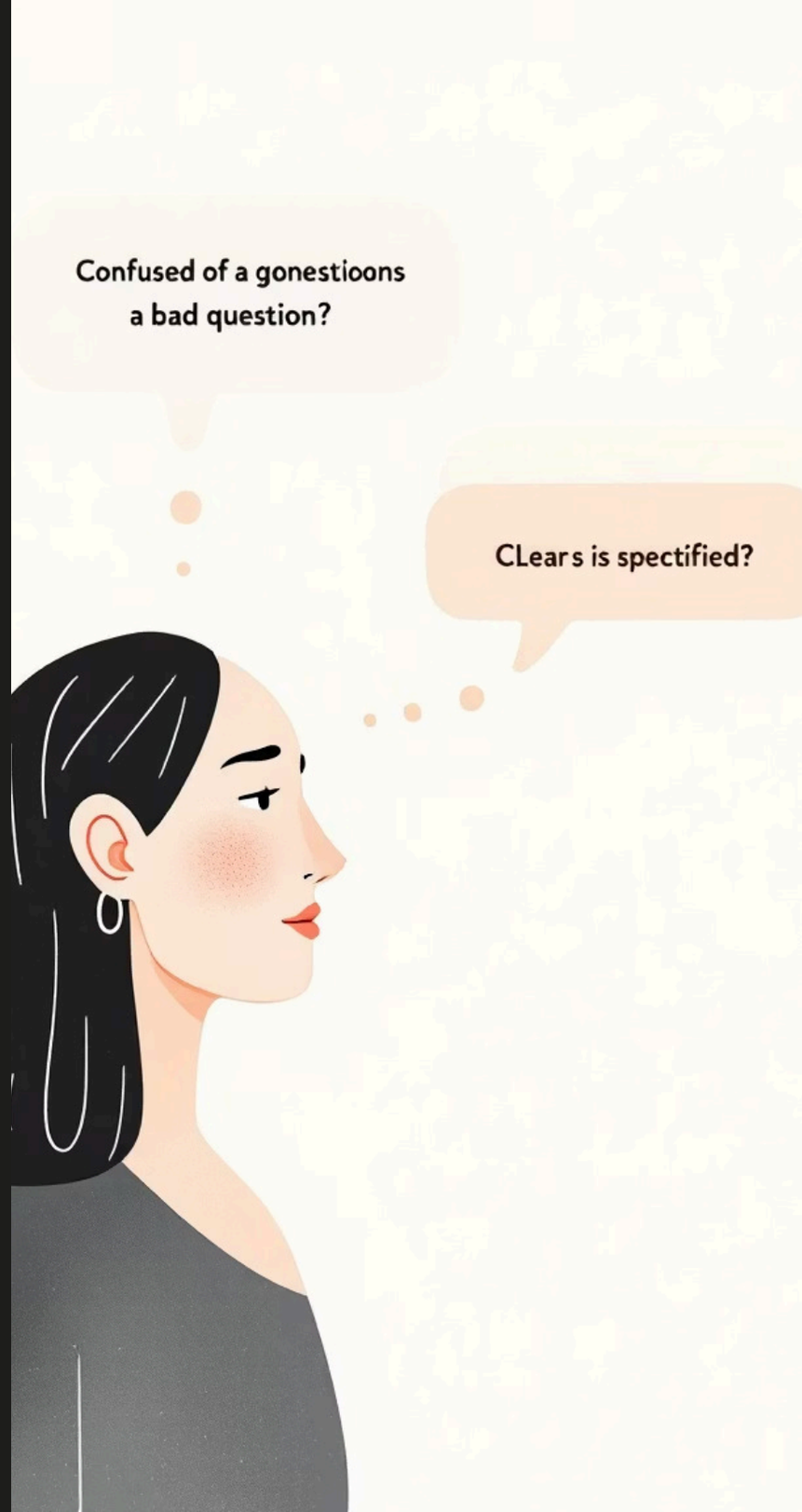
→ Resultado: Respuesta específica

LA DIFERENCIA:

- Prompt malo: 5 palabras
- Prompt bueno: Contexto + Problema + Restricciones + Objetivo

FÓRMULA MÁGICA:

Contexto + Problema + Restricciones + Objetivo + Formato = Buena respuesta



Componentes Principales

Chains	Secuencias de operaciones	Traducir → Resumir → Formatear
Memory	Recordar conversaciones	Chatbot que recuerda tu nombre
Tools	Extender capacidades del LLM	Buscar Google, calcular, leer PDFs
Output Parsers	Estructurar respuestas	Convertir texto a JSON

🎯 5 Técnicas Que Funcionan

01

TÉCNICA 1: FEW-SHOT LEARNING

- Dale ejemplos al LLM
- "Aquí hay ejemplos de buenas recomendaciones..."
- LLM aprende del patrón

Resultado: Respuestas consistentes

02

TÉCNICA 2: CHAIN OF THOUGHT

- Pide que explique paso a paso
- "Analiza esto en 3 pasos..."
- LLM razona en lugar de adivinar

Resultado: Respuestas verificables

03

TÉCNICA 3: ROLE PLAYING

- "Eres un especialista técnico de Lazarus con 20 años"
- LLM adopta personalidad y expertise

Resultado: Respuestas profesionales

04

TÉCNICA 4: RESTRICCIONES CLARAS

- "SOLO usa documentos proporcionados"
- "Si no sabes, di: 'No tengo información'"
- "NUNCA inventes especificaciones"

Resultado: Sin alucinaciones

05

TÉCNICA 5: ESTRUCTURA DE SALIDA

- Especifica el formato exacto
- "Responde en este formato: Nombre / Usos / Especificaciones"

Resultado: Respuestas consistentes y parseables

Ejemplo Rápido

Imagina un asistente virtual que no solo entiende tus preguntas, sino que también puede buscar en internet para encontrar la respuesta más actual. Eso es LangChain en acción:

- Combina Modelos de Lenguaje Grandes (LLMs) con herramientas externas.
- Permite al LLM interactuar con el mundo exterior (ej. búsqueda web, bases de datos).
- Resultados: respuestas más precisas, actualizadas y contextuales.

⚙️ Parámetros: Controlando el Comportamiento

PARÁMETRO 1: TEMPERATURE (Creatividad)

- 0.0 = Determinístico, sin creatividad → Perfectamente reproducible
- 0.5 = Balanceado (RECOMENDADO PARA LAZARUS) → Exactitud + claridad
- 1.5 = Muy creativo → Impredecible

PARÁMETRO 2: MAX_TOKENS (Longitud)

- 200 = Muy corto → Puede quedar incompleto
- 300-500 = RECOMENDADO → Balance costo/calidad
- 2000 = Muy largo → Caro y lento

PARÁMETRO 3: TOP_P (Diversidad)

- 0.5 = Muy restrictivo
- 0.9 = RECOMENDADO → Mantiene calidad
- 1.0 = Todo permitido → Más alucinaciones

PARÁMETRO 4: FREQUENCY_PENALTY (Repetición)

- 0 = Sin penalización → Puede repetir palabras
- 0.5 = RECOMENDADO → Respuestas más diversas
- 1.0 = Muy restrictivo → Fuerza palabras raras

Conclusión

LangChain se establece como un marco indispensable para el desarrollo de aplicaciones de inteligencia artificial de nueva generación. Su arquitectura permite a los Modelos de Lenguaje Grandes (LLMs) ir más allá de su entrenamiento inicial, conectándolos con fuentes de datos externas y herramientas del mundo real.

Esto resulta en:

- **Mayor precisión:** Respuestas basadas en información actualizada.
- **Interacción contextual:** La IA puede buscar y procesar datos específicos para cada consulta.
- **Potencial ilimitado:** La base para construir asistentes, agentes y aplicaciones de IA verdaderamente inteligentes y dinámicas.

📊 Resultados Medibles

MÉTRICA 1: COSTO	$0.0008 por pregunta	$0.00025 por pregunta	**70% MENOS ↓**
MÉTRICA 2: TOKENS USADOS	850 tokens promedio	500 tokens promedio	**40% MENOS ↓**
MÉTRICA 3: TIEMPO DE RESPUESTA	2.5 segundos	1.5 segundos	**40% MÁS RÁPIDO ↓**
MÉTRICA 4: CALIDAD/SATISFACCIÓN	70% respuestas buenas	95% respuestas buenas	**25% MEJOR ↑**

RESULTADO FINAL:

- ✅ Semana 2 = Mejor en TODOS los aspectos
- ✅ Más rápido, más barato, mejor calidad
- ✅ Esto es lo que pasa cuando optimizas correctamente

💰 Costos: El Factor Crítico

DESGLOSE ACTUAL (Semana 1):

- Por pregunta: $0.0008 (0.8 centavos)
- Embeddings: $0 (ya cacheado)
- 1000 preguntas: $0.80
- 100,000 preguntas/año: $80

DESPUÉS DE OPTIMIZAR (Semana 2):

- Por pregunta: $0.00025 (0.25 centavos) = **70% MENOS**
- 1000 preguntas: $0.25
- 100,000 preguntas/año: $25

CÓMO LO LOGRAREMOS:

→ Mejores prompts = menos tokens innecesarios (-30%)

→ Reducir max_tokens de 500 a 300 (-40%)

→ Reducir K de 5 a 3 (-20%)

→ Caché de preguntas frecuentes (-20%)

MODELO QUE USAMOS:

gpt-4o-mini: $0.15 input, $0.60 output per 1M tokens (BARATO Y BUENO)

🔍 Retriever: Mejorando la Búsqueda

PARÁMETRO K (Documentos a recuperar):

K=1

Solo 1 documento

- Rápido pero arriesgado: si ese doc es malo, respuesta es mala

K=3 (RECOMENDADO)

Top-3 documentos

- Si uno es malo, los otros lo corrigen
- Balance perfecto velocidad/información

K=5

5 documentos

- Más contexto pero LLM se pierde
- Más tokens = más caro

K=10 (NO RECOMENDADO)

Demasiado

TIPO DE BÚSQUEDA:

- **SIMILARITY:** Rápido y efectivo (USA ESTO)
- **SEMANTIC:** Más preciso pero lento
- **MMR:** Evita duplicados