Universidad de Manizales

Facultad de Ingeniería
Programa de Ingeniería de Sistemas y Telecomunicaciones

Profundización I en Ingeniería Artificial

Informe

Profesor:
Carlos Betancour

Integrantes:
Juan David Villaneda Herrera
Isabella Díaz Galindo

22 de noviembre de 2025

REFLEXIÓN METACOGNITIVA

1. Reflexión sobre el razonamiento analítico y técnico

¿Qué tareas exigieron más razonamiento analítico o verificación técnica?

Para nosotros, las actividades que más exigieron razonamiento analítico fueron:

- La comparación y ejecución de modelos en **Hugging Face** (NLP, visión y audio), porque tuvimos que interpretar métricas, revisar arquitectura y evaluar limitaciones reales.

- La construcción del pipeline **RAG con LLaMaIndex**, donde debimos verificar manualmente la indexación, ajustar el chunking y comprobar que las respuestas realmente estuvieran basadas en nuestros documentos.

- La replicación de un experimento desde **Papers With Code**, ya que nos obligó a comparar nuestras métricas con las del paper original y analizar por qué había diferencias.

- El despliegue del **Hugging Face Space**, que nos hizo depurar errores del entorno, revisar dependencias y validar la inferencia en producción.

¿Cuándo tuvimos que ir más allá de seguir instrucciones?

Tuvimos que diseñar y deducir por nuestra cuenta cuando:

- Los modelos no arrojaron los resultados esperados y tocó modificar prompts, parámetros o procesos.

- Hubo que integrar varias plataformas en el caso integrador y tomar decisiones de arquitectura sin un paso a paso definido.

- Comparábamos herramientas como Copilot y Codeium, porque no bastó ejecutarlas: debimos evaluar calidad, estilo, errores y utilidad real.

2. Reflexión sobre corrección de supuestos y evolución conceptual

Ideas iniciales que resultaron incompletas o erróneas

Como equipo comenzamos con varias suposiciones que luego resultaron parciales:

- Creíamos que “el modelo más grande siempre sería mejor”. Después comprobamos que muchos modelos medianos tenían mejor desempeño en nuestro caso por costo y velocidad.

- Pensábamos que un pipeline RAG siempre daría respuestas completamente verificables. Aprendimos que depende mucho del indexador, la segmentación y la calidad de los documentos.

- Esperábamos que los copilotos resolvieran la mayor parte del código sin supervisión; luego vimos que requieren análisis y revisión humana constante.

Cómo cambió nuestra comprensión

A lo largo del laboratorio entendimos que:

- La ingeniería de prompts es un proceso iterativo y experimental, no una receta fija.

- La reproducibilidad (con Papers With Code) exige controlar versiones, seeds, dependencias y métricas.

- El “ecosistema” completo importa tanto como el modelo: datos, limpieza, preparación, despliegue, hardware y documentación.

3. Reflexión sobre estrategias de resolución de problemas

Estrategias que utilizamos

Durante el laboratorio aplicamos varias estrategias cognitivas, entre ellas:

- **Descomposición:** dividir problemas grandes en módulos (indexación, modelo, API, interfaz).

- **Experimentación iterativa:** modificar una variable a la vez y comparar resultados.

- **Contrastación de fuentes:** especialmente usando Perplexity AI vs Google Scholar.

- **Lectura y análisis de documentación:** para entender fallos en modelos y APIs.

- **Comparación de outputs:** entre diferentes modelos y entre ejecuciones propias vs papers.

Estrategia más útil

La estrategia más útil para nosotros fue la **experimentación iterativa combinada con la descomposición del problema**, porque nos permitió identificar causas raíz y mejorar el rendimiento sin perder el control de lo que estábamos cambiando.

4. Reflexión sobre integración de plataformas y transferencia de conocimiento

Qué aprendimos sobre integrar plataformas

Aprendimos que:

- Cada plataforma tiene sus propios formatos, límites y estilos de autenticación.

- La integración requiere transformar datos, normalizar textos y coordinar tokenizadores.

- Muchos modelos funcionan mejor cuando los combinamos con APIs externas o pipelines adicionales (RAG, embeddings, evaluadores).

Decisiones que tomamos para que el caso integrador funcionara

Tuvimos que decidir:

- El tamaño del chunk y la estrategia de embedding en LLaMaIndex para balancear precisión y costo.

- Qué modelo usar en Hugging Face para garantizar desempeño sin saturar recursos.

- Cómo consumir la API de Replicate desde un microservicio.

- Cómo manejar tokens y variables de entorno sin exponerlos en GitHub.

Conocimientos que transferimos entre herramientas

- Lo aprendido en ingeniería de prompts en Google AI Studio nos sirvió para mejorar preguntas y análisis en NotebookLM.

- El flujo experimental de Hugging Face lo aplicamos también en Modelscope.

- Las prácticas de documentación de Papers With Code las usamos en el README y en la estructura del repositorio final.

5. Reflexión sobre nuestro crecimiento profesional

Cómo fortalecer nuestra identidad profesional

Este laboratorio nos ayudó a vernos más como **ingenieros que integran sistemas y toman decisiones informadas**, no solo como programadores que siguen instrucciones.

Además:

- Aprendimos a documentar mejor.

- A evaluar trade-offs técnicos.

- A analizar resultados con criterio y no solo aceptar lo que dice el modelo.

Lo que descubrimos sobre nuestras capacidades

Ambos descubrimos que:

- Podemos aprender herramientas nuevas muy rápido.

- Somos capaces de integrar sistemas distintos y hacerlos trabajar juntos.

- Tenemos criterio para evaluar tecnologías y decidir qué conviene o no en un proyecto.

Fortalezas actuales y lo que debemos reforzar

Fortalezas:

- Análisis técnico

- Experimentación
- Integración de modelos con APIs
- Documentación y diseño de prototipos

A reforzar:

- MLOps (CI/CD para modelos)
- Optimización de modelos (quantization, pruning)
- Seguridad en IA
- Pruebas automatizadas para pipelines ML

6. Reflexión sobre la profesión en la era de la IA

Cambios en el rol del desarrollador

Percibimos que el rol cambia así:

- De escribir código desde cero → a **orquestar modelos, servicios y APIs**.
- De implementar funciones → a **evaluar, supervisar y mejorar sistemas inteligentes**.
- De solo programar → a diseñar arquitecturas híbridas humano-IA.

Tareas que se van a automatizar

- Código repetitivo
- Documentación inicial
- Limpieza básica de datos
- Pruebas y refactorizaciones simples

Nuevas responsabilidades del ingeniero

- Supervisión de modelos
- Identificación y mitigación de sesgos
- Garantizar privacidad y cumplimiento de licencias
- Observabilidad y métricas de rendimiento
- Diseño de flujos donde modelo y humano cooperan

Cómo nos vemos en este nuevo contexto

Nos vemos como ingenieros capaces de:

- Integrar IA en proyectos reales
- Evaluar tecnología con criterio
- Diseñar soluciones híbridas
- Seguir aprendiendo constantemente

Y sabemos que debemos fortalecer conocimientos en automatización, MLOps, seguridad y gobernanza de modelos para mantenernos actualizados y aportar valor en esta nueva era.