Deep Learning para Business Analytics

Tema B — Sistemas sobre el agente

RAG, MCP, multi-agente, adaptación y el presupuesto completo.



Eduardo C. Garrido-Merchán

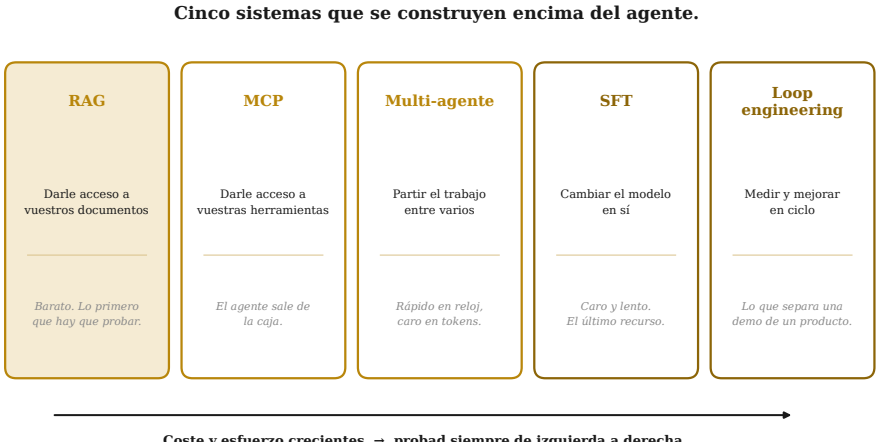
Profesor Propio Adjunto, Métodos Cuantitativos, ICADE

ecgarrido@comillas.edu

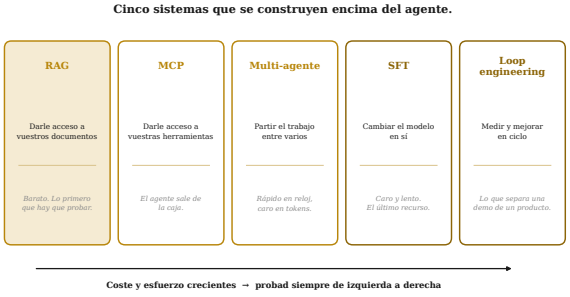
Madrid, 2026

herramientas del tema

Cinco sistemas que se construyen encima del agente.



Cómo se lee esta hoja de ruta.



El orden importa. Los cinco bloques están ordenados por coste y por esfuerzo. Casi todos los proyectos que fracasan empezaron por el cuarto.

La pregunta que los separa. ¿Al modelo le falta *información* (RAG), le faltan *acciones* (MCP), le falta *capacidad de trabajo* (multi-agente) o le falta *estilo* (SFT)?

El quinto no es opcional. Sin un ciclo de medida, los otros cuatro son decoración: nadie sabe si el cambio mejoró algo.

Idea clave. Este tema cierra con el presupuesto completo, que es el entregable del proyecto final.

Parte 1 / 6

RAG: darle acceso a vuestros documentos



Un agente que no ha leído vuestros papeles falla de tres maneras.

Tres formas de equivocarse que tiene un agente que no ha leído vuestros documentos.

Un empleado de Banco Iberia le pregunta al asistente: **«¿Con cuánta antelación tengo que pedir una excedencia?»**

1 · Alucina

Contesta igual de seguro sin tener ningún documento delante.

> *«Quince días naturales, conforme al artículo 46 del Estatuto de los Trabajadores.»*

Lo que pasa de verdad:

El artículo 46 del Estatuto regula la excedencia, pero no fija ningún plazo de preaviso. La cifra y la cita son invención del modelo.

Y la respuesta correcta, treinta días, está escrita en vuestro convenio.

2 · No sabe cuál de ellos manda

La misma pregunta tiene tres respuestas dentro de la propia empresa.

Convenio de empresa 2019, art. 14 — *vigente*
dice: treinta días naturales

Política de RR.HH. 2024, §3.2
dice: quince días laborables

Circular 03/2026 de Personas
dice: deroga la política de 2024

Los tres existen. Sólo uno está vigente. El modelo no ha visto ninguno.

3 · No caben en el prompt

Pegarle la normativa entera no es una opción, ni cara ni barata.

4.000 documentos internos son

38.000.000 tokens de normativa

y la ventana de contexto del modelo es

1.000.000 de ventana

Cabe el 2,6 % del corpus.

Y si cupiera, cada pregunta:

$114,01 por consulta

Con RAG, cinco fragmentos: $0,0154
7.422 veces más barato.

Los tres fallos tienen la misma raíz: el modelo no tiene delante el documento que contesta la pregunta. Precios a 2026-10-02.

Los tres fallos son el mismo fallo.



Los tres fallos tienen la misma raíz: el modelo no tiene delante el documento que contesta la pregunta. Precios a 2026-10-02.

Alucina. El modelo contesta siempre, y contesta con el mismo tono de seguridad cuando sabe y cuando no. Quince días y el artículo 46 suenan exactos; los dos son inventados.

No distingue cuál de vuestros documentos manda. El convenio, la política de 2024 y la circular de 2026 dicen cosas distintas. Un humano sabe cuál está vigente porque conoce la casa. El modelo no ha visto ninguno de los tres.

No caben. Cuatro mil documentos son 38 millones de tokens. En una ventana de contexto de un millón cabe el 2,6 %, y esa consulta costaría $114.

Idea clave. Las tres cosas se arreglan igual: poniéndole delante el documento que contesta la pregunta, y sólo ese.

RAG son tres palabras, y las tres hacen falta.

RAG, letra a letra, sobre la misma pregunta.

Retrieval-Augmented Generation · se lee al revés: generación aumentada por recuperación

R — Retrieval
Recuperación

Busco en los documentos de Banco Iberia los trozos que más se parecen a la pregunta.

Lo que sale:

Convenio 2019, art. 14 · sim 0,81
Circular 03/2026 · sim 0,74
Protocolo de bajas por IT · sim 0,52
Permiso por nacimiento · sim 0,49
Plan de pensiones · sim 0,31

Cinco trozos, no cuatro mil documentos.

A — Augmented
Aumentada

Pego esos trozos en el prompt, delante de la pregunta. El prompt crece; el modelo no cambia.

Lo que se le manda:

Responde usando SOLO estos textos.
Cita el documento de cada dato.
Si no está aquí, di que no lo sabes.
[Convenio 2019, art. 14] «…con una antelación mínima de treinta días…»

3.200 tokens de prompt, no 38 millones.

G — Generation
Generación

El modelo redacta leyendo sólo lo que tiene delante, y dice de qué documento ha salido cada dato.

Lo que contesta:

«Treinta días naturales de antelación, salvo que sea por cuidado de un familiar, en cuyo caso son quince. Fuente: Convenio de empresa 2019, artículo 14.2 y 14.3.»

Con la cita, alguien puede ir al documento y comprobarlo.

El modelo es el mismo de la diapositiva anterior. Lo único que ha cambiado es lo que tiene delante cuando contesta.

Qué significa cada letra, y cuál es la que cuesta dinero.



***Retrieval*, recuperación.** Buscar en vuestros documentos los fragmentos que se parecen a la pregunta. Es una búsqueda, no una llamada al modelo: cuesta milisegundos y céntimos.

***Augmented*, aumentada.** Pegar esos fragmentos en el prompt. Aquí es donde crece la factura: cada fragmento que inyectáis se paga en cada consulta.

***Generation*, generación.** El modelo redacta con eso delante. El modelo no ha cambiado; sólo ha cambiado lo que lee.

Idea clave. RAG convierte «el modelo sabe» en «el modelo tiene delante», y eso sí se puede auditar: el fragmento viene de un documento con nombre y fecha.

Un RAG contestando, y luego negándose a contestar.

Demo en directo · 6 min

jupyter — ~/demo_rapida

```
$ cd ~/demo_rapida
$ jupyter lab 01_rag.ipynb

>>> responder(pregunta, con_contexto=False)
>>> responder(pregunta, k=3)
>>> responder('¿Cuántas oficinas tiene
    Banco Iberia en Galicia?', k=3)
```

Lanzo. El cuaderno 01 de la demo rápida, sobre la normativa de Banco Iberia: hipotecas, tarjetas, fraude y protección de datos. Ya está troceada e indexada; arrancamos en la consulta.

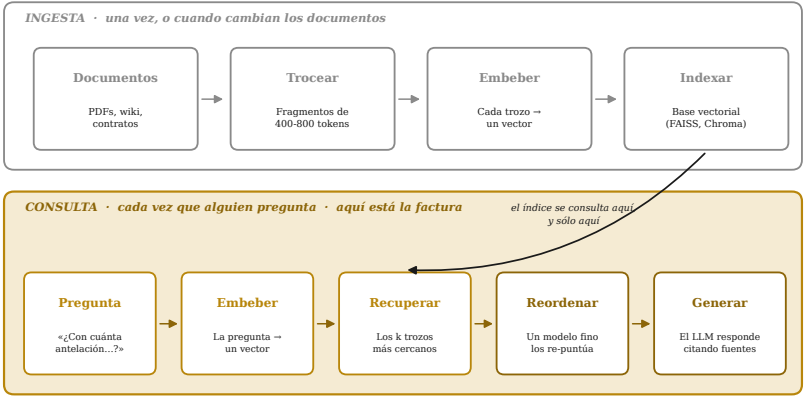
Miráis. La misma pregunta sin contexto y con contexto. Luego, los cinco fragmentos recuperados con su similitud, y qué hace con una pregunta cuya respuesta no está en ningún documento.

Si falla. Si Jupyter no arranca, el mismo cuaderno ya ejecutado está publicado en la web del curso, con la salida de todas las celdas.

Idea clave. Un RAG bien montado dice «no está en la documentación que manejo». El mismo modelo sin RAG se inventa el número de oficinas.

RAG completo: una fase que se hace una vez y otra que se paga siempre.

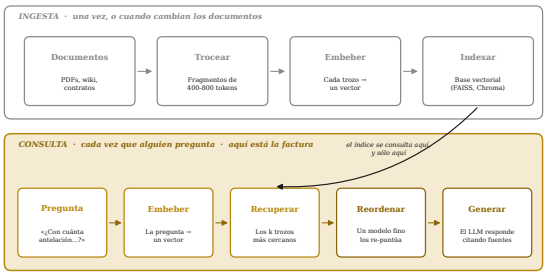
RAG completo: una fase que se hace una vez y otra que se paga en cada consulta.



El 90 % de los tutoriales acaban en «Recuperar». El reordenado y las citas son lo que separa un juguete de algo que se puede enseñar a un cliente.

Las dos fases cuestan en sitios distintos.

RAG completo: una fase que se hace una vez y otra que se paga en cada consulta.



El 90 % de los tutoriales acaban en «Recuperar». El reordenado y las citas son lo que separa un juguete de algo que se puede enseñar a un cliente.

La ingesta se paga una vez. Trocear, embeber e indexar los 4.000 documentos. Se repite sólo cuando cambian, y embeber un millón de tokens cuesta céntimos.

La consulta se paga siempre. Y de los cinco pasos, el que cuesta es inyectar los fragmentos en el prompt del generador.

El índice sólo lo toca «Recuperar». Es el único punto donde la fase de consulta habla con lo que construyó la ingesta. Los otros cuatro pasos no saben que existe una base vectorial.

Idea clave. El 90 % de los tutoriales acaban en «Recuperar». El reordenado y las citas son lo que separa un juguete de algo que se le puede enseñar a un cliente.

El mismo artículo troceado de tres maneras da tres respuestas.

El mismo artículo troceado de tres maneras, y la respuesta que sale de cada una.

Convenio de Banco Iberia, artículo 14 · pregunta: «¿Con cuánta antelación tengo que pedir una excedencia?»



Trocear por la estructura del documento (por artículo, por apartado, por fila de la tabla) gana casi siempre a cualquier número fijo de tokens.

Trocear: el parámetro que más se descuida.



Qué se rompió con 150 tokens. El trozo recuperado empieza por «El plazo del apartado anterior». El apartado anterior se quedó en el trozo de al lado, así que el sistema contesta quince donde la respuesta es treinta.

Ejemplo a mano: El trozo que lo arregla

«Artículo 14. Excedencia voluntaria. 14.1 El personal con al menos un año de antigüedad… 14.2 …con una antelación mínima de treinta días naturales. 14.3 El plazo del apartado anterior se reduce a quince días cuando sea por cuidado de un familiar.» Leído solo, se entiende entero.

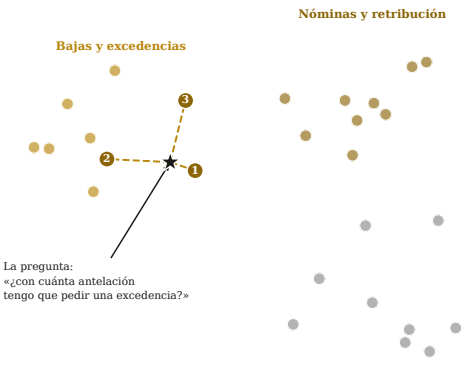
Punto de partida razonable. Entre 400 y 800 tokens, con un solapamiento del 10 al 15 % para no cortar una idea por la mitad.

Mejor que afinar el número. Trocear por la estructura: por artículo, por apartado, por fila de la tabla.

Idea clave. Si un fragmento no se entiende leído solo, el modelo tampoco lo va a entender.

Un embedding coloca cada fragmento en un mapa donde la cercanía es significado.

Un embedding coloca cada fragmento en un mapa donde la cercanía es significado.



El mapa real tiene entre 384 y 3.072 dimensiones; aquí lo vemos en dos.

Recuperar = buscar los *k* vectores más cercanos a la pregunta.

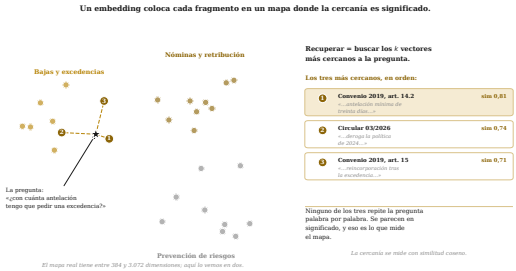
Los tres más cercanos, en orden:

1. **Convenio 2019, art. 14.2** — sim 0,81
   «…antelación mínima de treinta días…»
2. **Circular 03/2026** — sim 0,74
   «…deroga la política de 2024…»
3. **Convenio 2019, art. 15** — sim 0,71
   «…reincorporación tras la excedencia…»

Ninguno de los tres repite la pregunta palabra por palabra. Se parecen en significado, y eso es lo que mide el mapa.

La cercanía se mide con similitud coseno.

Similitud coseno: la cuenta que hace la recuperación.



$$\mathrm{sim}(\mathbf{q}, \mathbf{d}) = \frac{\mathbf{q} \cdot \mathbf{d}}{\|\mathbf{q}\| \, \|\mathbf{d}\|}$$

q vector de la pregunta **d** vector de un fragmento

Resultado en $[-1, 1]$: 1 es misma dirección, 0 sin relación.

Por qué el coseno. Mide el ángulo, no la magnitud: un documento largo y uno corto sobre lo mismo apuntan en la misma dirección.

Ejemplo a mano: A mano, en dos dimensiones

Con $\mathbf{q} = (3, 4)$ y $\mathbf{d} = (6, 8)$: producto escalar = 50, normas 5 y 10, luego $\mathrm{sim} = 50/50 = 1$.

Idea clave. El fragmento que gana, el artículo 14.2, no contiene la palabra «excedencia» en la frase que responde. Por eso no basta con buscar palabras.

Una métrica cuenta preguntas; la otra cuenta frases.

Las dos métricas, contadas a mano sobre la misma pregunta.

Una cuenta preguntas y mira el buscador. La otra cuenta frases y mira al que redacta.

Recall@5 · mide el BUSCADOR

¿Estaba el trozo que contesta entre los cinco que traje?

Los cinco trozos recuperados, por similitud:

1.	Convenio 2019, art. 15 — reincorporación	*sim 0,83*
2.	**Convenio 2019, art. 14.2 — antelación mínima**	*sim 0,81* ✓
3.	Circular 03/2026 — deroga la política de 2024	*sim 0,74*
4.	Protocolo de bajas por IT	*sim 0,52*
5.	Permiso por nacimiento	*sim 0,49*

El trozo bueno llegó en el puesto 2 de 5: esta pregunta cuenta como acierto.

Se repite con las 30 preguntas del conjunto de prueba:

$$\text{Recall@5} = \frac{25 \text{ preguntas con acierto}}{30 \text{ preguntas}} = 83\,\%$$

Si sale bajo: el troceado, el modelo de embedding o un *k* demasiado pequeño.

Faithfulness · mide al que REDACTA

De lo que ha afirmado, ¿cuánto se sostiene en esos cinco trozos?

La respuesta, partida en afirmaciones:

✓	*«Treinta días naturales de antelación.»* art. 14.2, literal
✓	*«Quince si es por cuidado de un familiar.»* art. 14.3, literal
✓	*«Hay que tener un año de antigüedad.»* art. 14.1, literal
✗	*«Y se puede prorrogar una vez.»* no aparece en ningún trozo

La cuarta es cierta en muchas empresas, y aquí nadie se la había dicho.

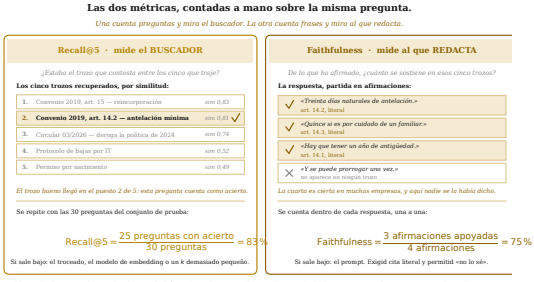
Se cuenta dentro de cada respuesta, una a una:

$$\text{Faithfulness} = \frac{3 \text{ afirmaciones apoyadas}}{4 \text{ afirmaciones}} = 75\,\%$$

Si sale bajo: el prompt. Exigid cita literal y permitid «no lo sé».

El buscador hizo su trabajo y el redactor añadió una frase de su cosecha: recall 83 %, faithfulness 75 %. Las dos cuentas son independientes.

Cómo se mide un RAG sin autoengañarse.



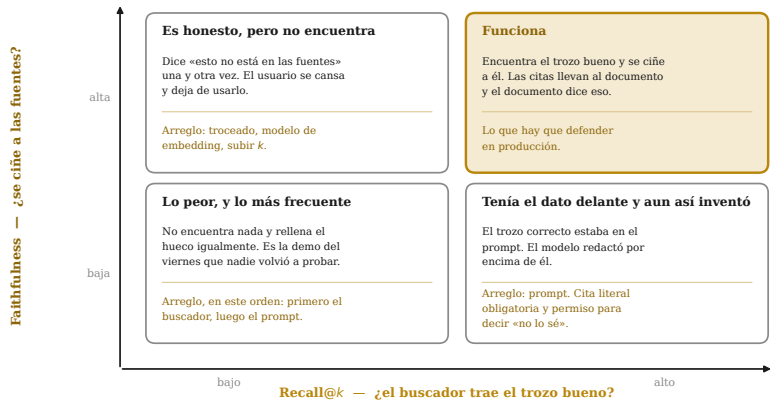
Recall@*k* se calcula una vez por conjunto. Pasáis las 30 preguntas de prueba, miráis si el fragmento correcto apareció entre los *k* recuperados, y dividís aciertos entre preguntas. No mira la respuesta: el generador ni siquiera ha intervenido.

Faithfulness se calcula una vez por respuesta. Partís la respuesta en afirmaciones, comprobáis cada una contra los fragmentos que se le pasaron, y dividís las apoyadas entre el total. No mira si la respuesta es cierta en el mundo: mira si sale de las fuentes.

Distinto denominador, distinto culpable. 25 de 30 *preguntas* frente a 3 de 4 *frases*. La primera acusa al buscador; la segunda, al que redacta.

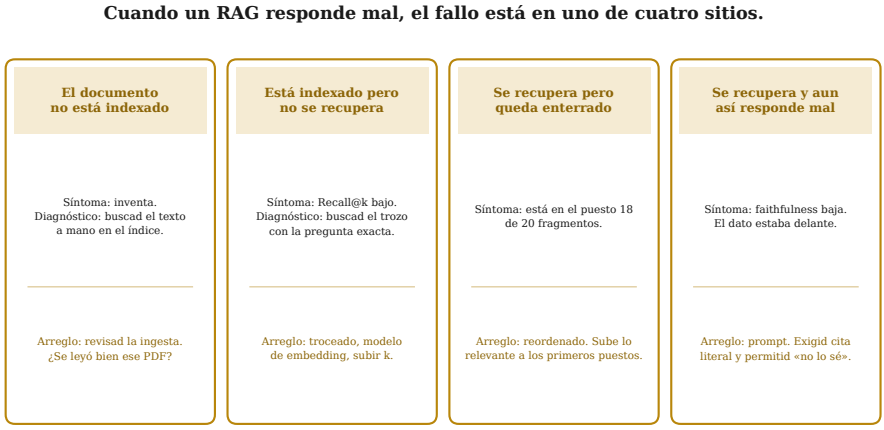
Idea clave. Treinta preguntas reales con su respuesta acordada bastan para empezar. Sin ese conjunto, «va mejor» es una impresión.

Las dos métricas se mueven por separado: por eso hacen falta las dos.



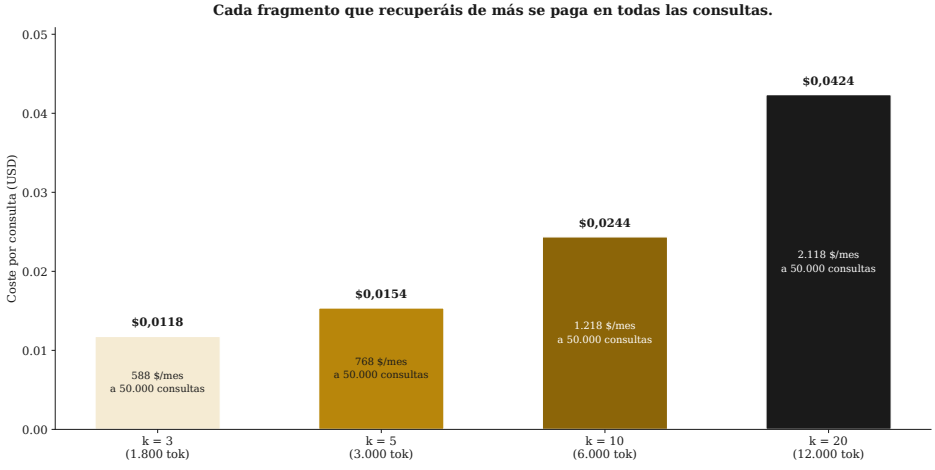
Una sola de las dos cifras no distingue el cuadrante de arriba a la derecha del de abajo a la derecha, y son problemas opuestos.

Cuando un RAG responde mal, el fallo está en uno de cuatro sitios.



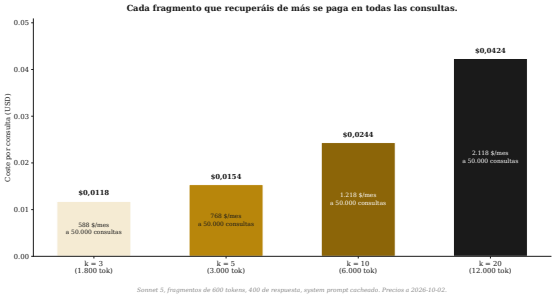
Los cuatro se diagnostican en ese orden, y cada uno tiene un arreglo distinto. Cambiar de modelo casi nunca arregla ninguno.

Cada fragmento que recuperáis de más se paga en todas las consultas.



Sonnet 5, fragmentos de 600 tokens, 400 de respuesta, system prompt cacheado. Precios a 2026-10-02.

El coste de un RAG, partida a partida.



Una vez. Embeber el corpus. Un millón de tokens de documentos cuesta céntimos con un modelo de embeddings: es la partida que nadie debería discutir.

Cada consulta. Embeber la pregunta (despreciable) y, sobre todo, los *k* fragmentos que inyectáis en el prompt. Ahí está el 90 % de la factura.

Mensual fijo. La base vectorial alojada, si no la ponéis vosotros.

La palanca. Subir *k* de 5 a 20 cuadruplica los tokens de contexto y sube el recall unos 12 puntos. Reordenar con un modelo barato os da ese recall con *k* pequeño.

Idea clave. El reordenado es el cambio con mejor relación calidad/coste de todo el pipeline.

Tres sitios donde podéis tocar un RAG esta misma tarde.

Tres sitios donde podéis tocar un RAG esta misma tarde.

El cuaderno de RAG, ya ejecutado

`demo-rapida/01_rag.html`

El corpus de Banco Iberia troceado, embebido y consultado, con la salida de cada celda a la vista.

Se lee en el navegador, sin instalar nada.

Estas mismas ideas, pero manipulables

`interactivas/tema_8/index.html`

Las slides interactivas del tema. Movéis *k*, el tamaño del trozo y el precio del modelo, y la factura se recalcula delante de vosotros.

También en el navegador.

La demo entera, para ejecutarla

`demo-rapida/index.html`

El zip con los cuatro cuadernos y el corpus.

Corre en un portátil sin GPU: el de RAG tarda unos tres minutos y pide 4,6 GB de RAM.

Los tres cuelgan de eduardogarrido90.github.io/docencia/dl/diapositivas-segunda-parte/ · el cuaderno de RAG es el que ejecutamos en clase.

Qué hacer con cada uno de los tres enlaces.



Los tres cuelgan de eduardogarrido90.github.io/docencia/dl/diapositivas-segundo-parte/ · el cuaderno de RAG es el que ejecutamos en clase.

Hoy, en diez minutos. Abrid el cuaderno ya ejecutado y leed dos celdas: la que recupera los fragmentos con su similitud, y la que compara la respuesta con contexto y sin contexto. Es el mismo pipeline de la figura, con números reales.

Esta semana, con el portátil. Descargad el zip y ejecutadlo. Cambiad k, cambiad el modelo de *embeddings* y mirad cómo se mueve el acierto. Corre sin GPU.

Para el proyecto. Las slides interactivas tienen la calculadora de coste por consulta que vais a necesitar en el presupuesto.

Idea clave. El corpus del cuaderno está inventado a propósito: ningún modelo lo ha visto en el preentrenamiento, así que cuando acierta una cifra es porque la ha leído.

Qué está pasando ya en las empresas de vuestro sector.

Cinco Días · Empresas — CincoDías

Las eléctricas montan buscadores internos sobre su normativa

El patrón se repite en banca, energía y seguros: indexar la normativa interna y dejar que el empleado pregunte en lenguaje natural, con la cita al documento original.

Expansión · Tecnología — Expansión

El 80 % de los proyectos de IA se quedan en la prueba piloto

La causa que más se repite no es técnica: no hay forma acordada de medir si el sistema funciona, así que nadie se atreve a ponerlo delante de un cliente.

Lo que separa el piloto del producto

Un conjunto de evaluación con casos reales y respuesta acordada.

Una métrica por cada tipo de fallo, medida antes y después.

Un responsable que decide si el número es suficiente para salir.

Parte 2 / 6

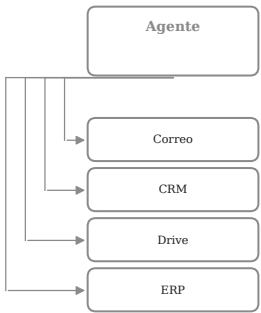
MCP: darle acceso a vuestras herramientas



MCP: un enchufe estándar entre el agente y vuestros sistemas.

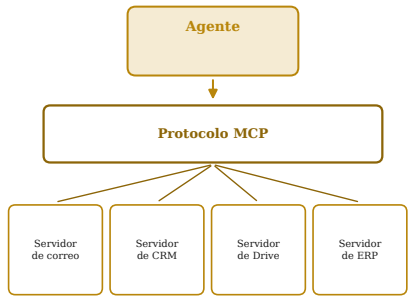
MCP: un enchufe estándar entre el agente y vuestros sistemas.

Antes: un conector a medida por sistema



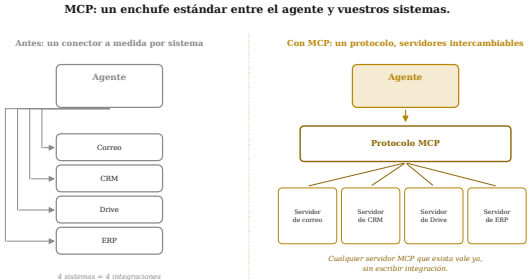
4 sistemas = 4 integraciones que mantener

Con MCP: un protocolo, servidores intercambiables



Cualquier servidor MCP que exista vale ya, sin escribir integración.

Qué problema resuelve el protocolo.



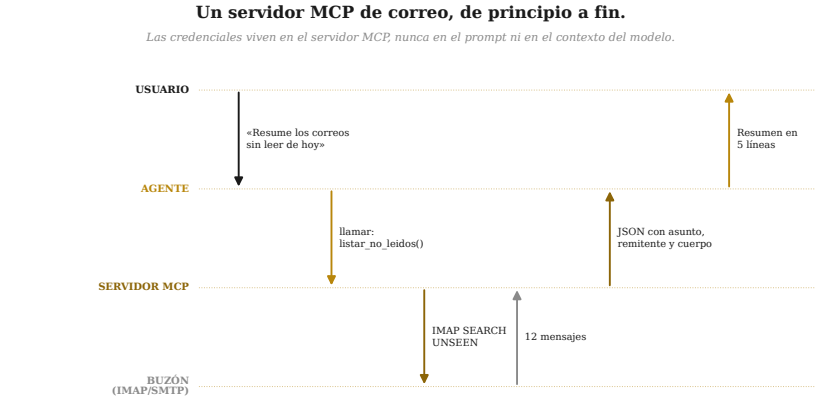
Antes. Cada equipo escribía su propio puente entre su agente y cada sistema. Con cuatro sistemas y tres agentes, doce integraciones que mantener.

Con MCP. El sistema expone un *servidor* que declara qué sabe hacer. Cualquier agente que hable el protocolo lo usa sin escribir código.

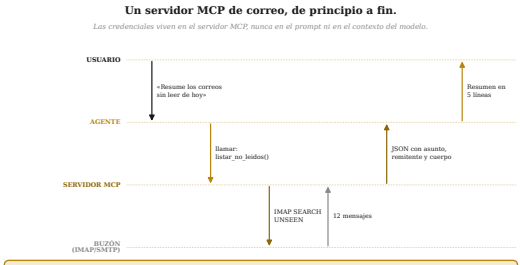
Las tres cosas que expone un servidor. *Herramientas* (acciones que ejecuta), *recursos* (datos que puede leer) y *prompts* (plantillas preparadas).

Idea clave. Lo valioso es el ecosistema: hay servidores ya escritos para Drive, GitHub, Slack, Postgres o correo. Conectar cuesta minutos.

Un servidor MCP de correo, de principio a fin.



El caso del correo, con sus riesgos encima de la mesa.



Qué expone el servidor. `listar_no_leidos()`, `leer(id)`, `buscar(consulta)`, `redactar_borrador(...)`, `enviar(...)`.

Dónde viven las credenciales. En el servidor MCP, nunca en el prompt ni en el contexto del modelo. El modelo pide «enviar»; quien tiene la contraseña es el servidor.

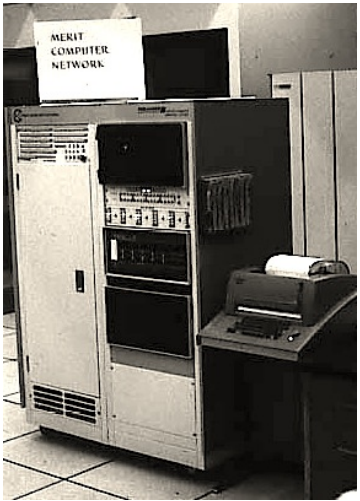
Permisos por método. Los cuatro primeros van solos. `enviar` pide confirmación siempre, porque un correo a un cliente no se deshace.

El riesgo nuevo: inyección de prompt. Un correo entrante puede contener texto diseñado para dar órdenes al agente que lo lea. Si el agente puede leer correo *y* enviarlo, un desconocido acaba de adquirir capacidad de escribir en vuestro nombre.

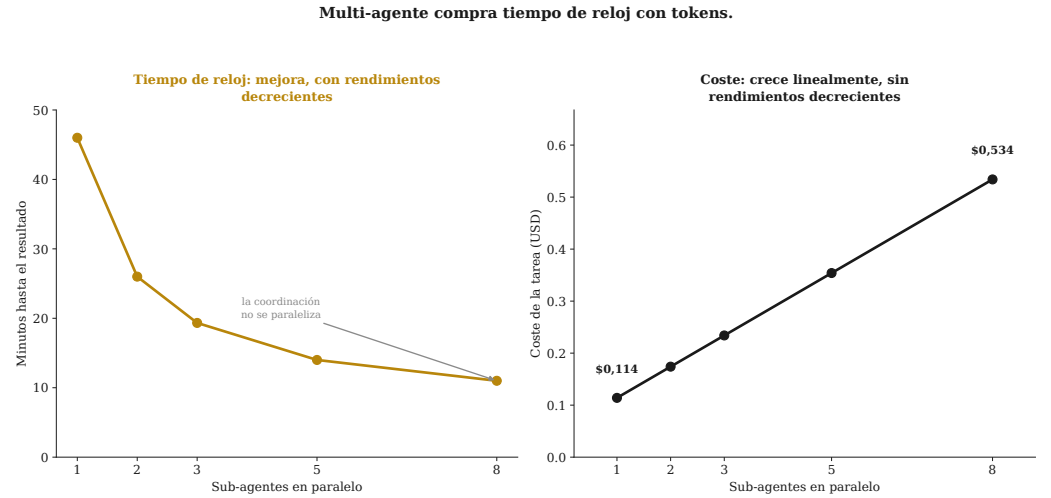
Idea clave. Leer datos no fiables y poder actuar sobre el mundo es la combinación peligrosa. Separad las dos cosas o poned un humano entre ellas.

Parte 3 / 6

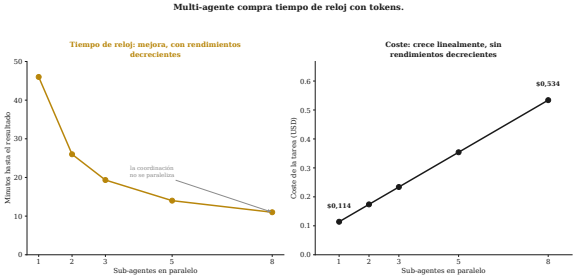
Sistemas multi-agente



Multi-agente compra tiempo de reloj con tokens.



Por qué el coste crece y el tiempo no baja igual.



El tiempo tiene suelo. Repartir el trabajo entre más agentes lo acelera, pero coordinar, repartir y juntar los resultados no se paraleliza. A partir de cierto punto, un agente más apenas aporta.

El coste no tiene techo. Cada sub-agente arranca con su propio contexto base: instrucciones, herramientas, el encargo. Ese contexto se paga entero, tantas veces como sub-agentes lancéis.

Ejemplo a mano: Ocho sub-agentes

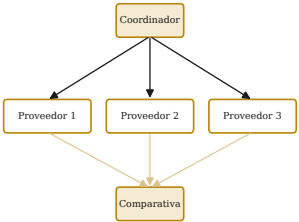
Con 8.000 tokens de contexto base por agente, ocho sub-agentes pagan 64.000 tokens sólo en arrancar, antes de hacer nada útil.

Idea clave. Lanzad sub-agentes cuando el trabajo sea de verdad independiente y voluminoso; nunca para partir una tarea pequeña.

Tres patrones que compensan el sobrecoste.

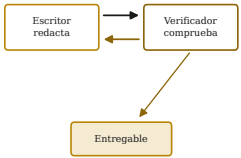
Tres patrones que compensan el sobrecoste.

Abanico



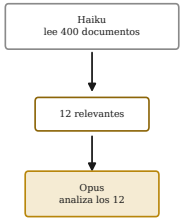
Trabajo independiente que se junta al final.

Escritor y verificador



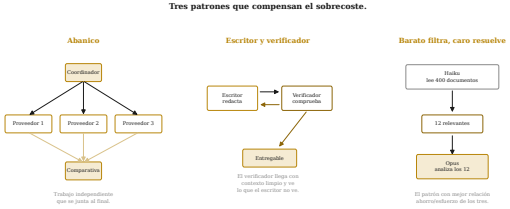
El verificador llega con contexto limpio y ve lo que el escritor no ve.

Barato filtra, caro resuelve



El patrón con mejor relación ahorro/esfuerzo de los tres.

Cuándo multi-agente merece la pena, y cuándo no.



Sí, cuando el trabajo se abre en abanico. Analizar 30 ofertas de proveedor, revisar 50 ficheros, investigar 12 competidores. Cada rama es independiente y voluminosa.

Sí, para separar escribir de revisar. El verificador llega con el contexto limpio y detecta lo que el escritor ya no ve, igual que un revisor humano.

Sí, para filtrar barato antes de analizar caro. Haiku descarta 388 documentos de 400; Opus analiza los 12 que quedan. Es el patrón de mejor relación ahorro/esfuerzo.

No, para una tarea secuencial. Si el paso 2 necesita el resultado del paso 1, no hay nada que paralelizar: estaréis pagando contextos de más para hacer lo mismo.

Idea clave. Regla práctica: si no sabéis enumerar las ramas independientes antes de lanzar, no es un caso multi-agente.

Parte 4 / 6

Adaptar el modelo: SFT



Prompt, RAG o SFT: tres formas de cambiar lo que hace el modelo.



Sólo RAG mete información nueva de forma fiable. SFT le enseña a sonar como vosotros, no a saber lo que sabéis.

Qué cambia de verdad cada una.

Prompt, RAG o SFT: tres formas de cambiar lo que hace el modelo.



Sólo RAG mete información nueva de forma fiable. SFT le enseña a sonar como vosotros, no a saber lo que sabéis.

El prompt. Le escribís en las instrucciones qué tono, qué formato y qué no hacer, con tres o cuatro ejemplos. Cuesta una tarde y se deshace en otra.

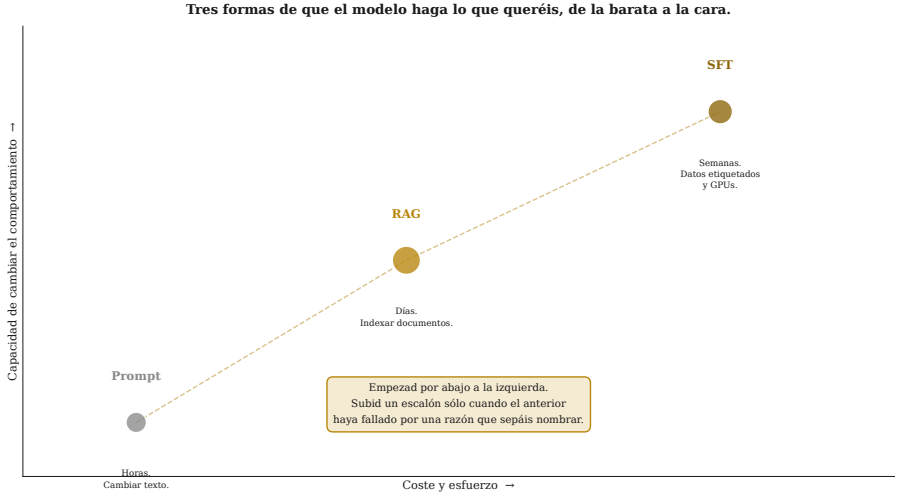
RAG. Le dais los documentos en el momento de responder. Es lo único de los tres que mete información que el modelo no tenía.

SFT. Le dais pares «entrada, salida correcta» y ajustáis los pesos para que reproduzca ese estilo. Sirve para formato, tono y estructura, y queda grabado en el modelo.

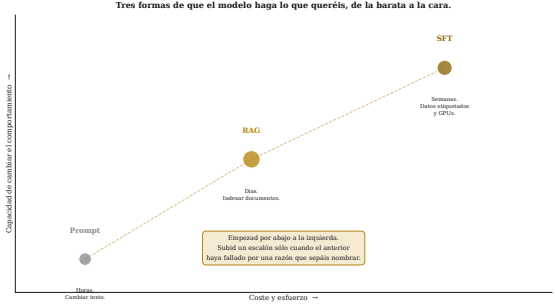
Lo que SFT no hace. Meter conocimiento nuevo. Si entrenáis con vuestra normativa, el modelo aprenderá a *sonar* a vuestra normativa y seguirá inventando artículos.

Idea clave. Conocimiento nuevo → RAG. Estilo → el prompt, y sólo si no basta, SFT.

De la barata a la cara: subid un escalón sólo cuando sepáis por qué.



El árbol de decisión, en serio.



¿Le falta información? → RAG. Siempre. Entrenar para meter datos es caro, lento y poco fiable.

¿Responde bien pero con el formato equivocado? → Primero el prompt, con tres o cuatro ejemplos. Es gratis y se prueba en una tarde.

¿Sigue fallando el formato después del prompt, y tenéis cientos de ejemplos resueltos? → Entonces SFT, y no antes.

¿Necesitáis que no salga de vuestra red? → Modelo libre servido por vosotros, con la cuenta de VRAM del Tema A.

Idea clave. En el proyecto final, justificad por qué NO habéis elegido el escalón anterior. Esa es la pregunta que os haré en la defensa.

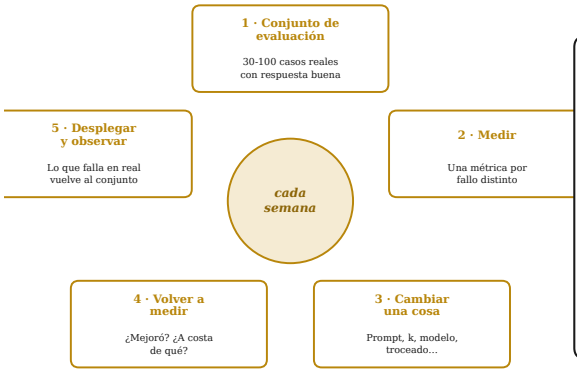
Parte 5 / 6

Loop engineering



El ciclo que convierte una demo en un producto.

Loop engineering: el ciclo que convierte una demo en un producto.



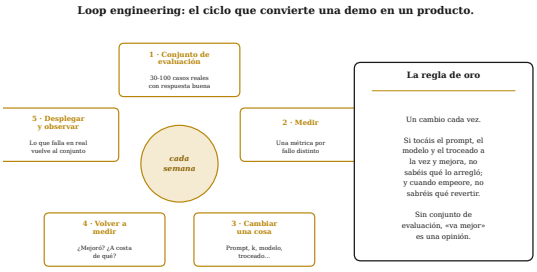
La regla de oro

Un cambio cada vez.

Si tocáis el prompt, el modelo y el troceado a la vez y mejora, no sabéis qué lo arregló; y cuando empeore, no sabréis qué revertir.

Sin conjunto de evaluación, «va mejor» es una opinión.

Cómo se monta el conjunto de evaluación.



Tamaño. Entre 30 y 100 casos. Menos no distingue señal de ruido; más no os lo vais a mantener.

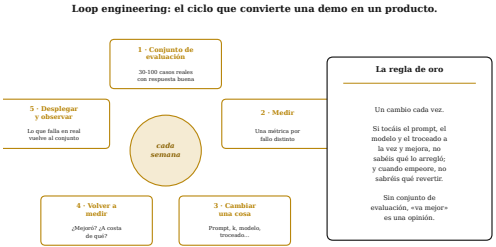
Procedencia. Preguntas reales de usuarios reales. Las que inventéis en una sala se parecerán a lo que el sistema ya hace bien.

Qué guardáis por caso. La pregunta, la respuesta que consideráis buena, y el documento o dato del que debería salir.

Quién decide la respuesta buena. Alguien del negocio, no del equipo técnico. Si no hay nadie que firme cuál es la respuesta correcta, el proyecto no está listo para empezar.

Idea clave. Lo que falle en producción se añade al conjunto. Así el conjunto mejora con el tiempo en vez de envejecer.

Las trampas que hacen que un ciclo de evaluación no sirva.



Cambiar varias cosas a la vez. Si tocáis prompt, modelo y troceado y mejora, no sabéis qué lo arregló; cuando empeore, no sabréis qué revertir.

Evaluar con los casos que usasteis para ajustar. Es el sobreajuste de toda la vida, con otro nombre. Reservad una parte que no se toca.

Usar un modelo como juez sin validarlo. Un LLM puede puntuar respuestas, y es cómodo. Antes comprobad que su puntuación coincide con la de una persona en una muestra.

Medir sólo la media. Un sistema con 85 % de acierto y fallos catastróficos en el 2 % restante es peor para el negocio que uno con 80 % y fallos suaves.

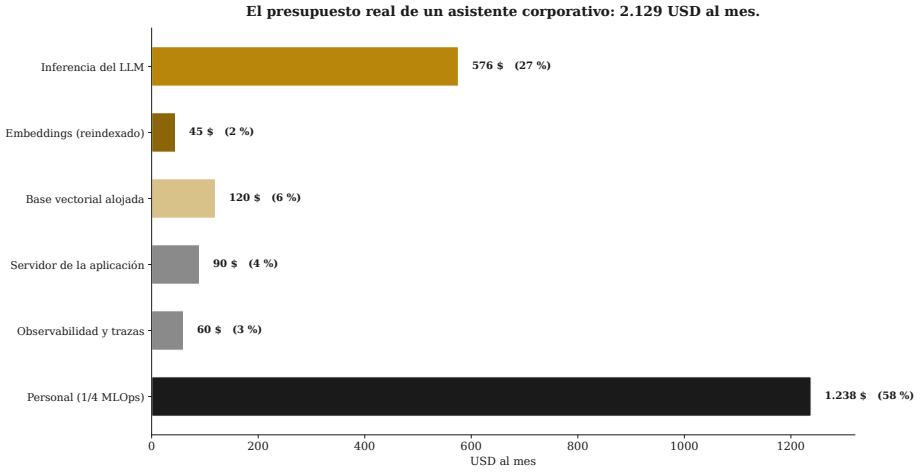
Idea clave. Observabilidad: guardad cada traza con su coste en tokens. Sin eso, el primer susto en la factura os pilla sin datos.

Parte 6 / 6

El presupuesto completo

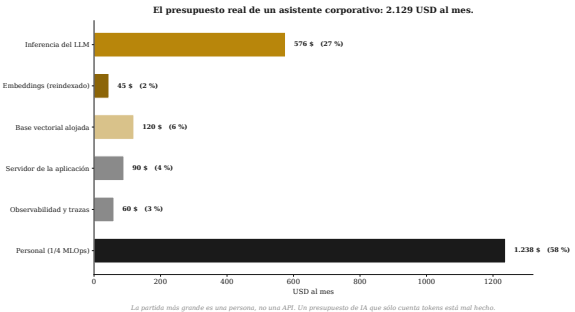


El presupuesto real de un asistente corporativo.



La partida más grande es una persona, no una API. Un presupuesto de IA que sólo cuenta tokens está mal hecho.

Cómo se construye esa tabla para vuestro caso.



Paso 1. Estimad una consulta típica: tokens de entrada, de salida y fracción cacheable. Medidlo, no lo adivinéis.

Paso 2. Multiplicad por vuestra previsión de consultas al mes. Ese número sale del plan de negocio.

Paso 3. Añadid lo que no son tokens: base vectorial, servidor, observabilidad, reindexado.

Paso 4. Añadid personas. Es la partida más grande y la que todos los grupos olvidan.

Paso 5. Dividid entre usuarios activos. Ese es el número que el inversor quiere oír.

Idea clave. Usad `calculadora_costes.py` del repositorio: hace los cinco pasos y cita la fecha de los precios.

Las tres preguntas que decide vuestro modelo de costes.

Pregunta del inversor	Qué está midiendo	Dónde lo sacáis
¿Cuánto cuesta atender a un usuario al mes?	Si el precio que cobráis deja margen.	Coste total ÷ usuarios activos.
¿Qué pasa si crecéis 10×?	Si vuestro coste es lineal o se dispara.	Repetir el cálculo con $N \times 10$.
¿Y si el proveedor sube el precio?	Vuestra dependencia de un tercero.	Rehacer con el precio ×1,5.
¿Cuánto del coste es fijo?	Cuánto perdéis en un mes malo.	Servidor + personal ÷ total.
¿Cuándo os compensa el servidor propio?	Si habéis pensado el largo plazo.	Punto de equilibrio del Tema A.

Idea clave. Un grupo que llegue al pitch con estas tres respuestas calculadas tiene media defensa hecha.

Lo que os lleváis de este tema.



Concepto 1. RAG da información; MCP da acciones; multi-agente da capacidad; SFT da estilo. Elegid por lo que falta, no por lo que suena mejor.

Concepto 2. Un RAG se mide con dos números: recall@k para el buscador y faithfulness para el generador. Fallan por motivos distintos y se arreglan en sitios distintos.

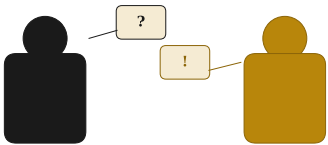
Concepto 3. MCP estandariza la conexión con vuestros sistemas. La combinación peligrosa es leer datos no fiables y poder actuar sobre el mundo.

Concepto 4. SFT enseña al modelo a sonar como vosotros, no a saber lo que sabéis. Para conocimiento, RAG; y antes de entrenar nada, probad el prompt.

Concepto 5. Sin conjunto de evaluación, «va mejor» es una opinión.

Para el proyecto. El presupuesto completo con las cinco respuestas de la tabla anterior.

Pausa para debatir: ¿a quién le toca responder cuando el RAG se equivoca?



El caso. Vuestro asistente de RR.HH. dice a un empleado que le corresponden 15 días de permiso. Eran 5. El empleado se los coge.

Posición A. Responde la empresa: puso la herramienta, y el empleado hizo lo que le dijeron sus sistemas internos.

Posición B. Responde el empleado: el aviso decía «verifique con RR.HH.», y la cita al convenio estaba ahí.

La pregunta de verdad. ¿Cambia vuestra respuesta si el sistema citaba el artículo correcto y lo resumió mal?

Reglas. Cinco minutos, dos turnos por bando. Yo modero y no participo.

Gracias.

¿Preguntas?

RAG **MCP** **SFT**

Eduardo C. Garrido-Merchán

ecgarrido@comillas.edu

