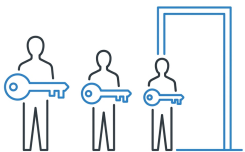
IA local para PyMEs · Módulo 12

Evaluación y observabilidad



Un set de evals desde cero, métricas por tarea, trazas multipaso y canarios nocturnos que avisan cuando algo se degrada.

Qué se construye en este módulo

Un set de evals desde cero

Escribir 60 casos con respuesta esperada y criterio

Métricas por tipo de tarea

Elegir la métrica según la tarea

Trazado multipaso y el bucle traza → eval → corrección

Encontrar el paso que falló en una traza

Qué se construye en este módulo

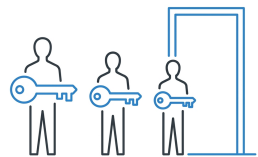
📈 Evals canario nocturnos y degradación

Programar un canario que corra cada noche

12.1 · Un set de evals desde cero

Un set de evals desde cero

Un **eval** es un caso con entrada, respuesta esperada y criterio de corrección. Un **set de evals** son sesenta. Se corre antes de cada cambio (modelo, prompt, esquema, embedding) y después. Si la nota baja, el cambio no entra. Es la única defensa contra «mejoramos el prompt y rompimos las facturas».



Lo que debe ver: cinco filas con fracciones. La de `permisos` tiene que ser **5/5 siempre**; si baja, nada más importa. La de `trampa` dice si el sistema inventa.

En una tabla

Fuente	Cuántos
Set de facturas anotadas (7.4)	20
Consultas del cliente con ficha esperada (6.2)	20
Transcripciones con acta corregida por el gerente (8.1)	10
Casos de permisos (6.3)	5
Casos «trampa» (pregunta sin respuesta en los documentos)	5

El comando, copiable

```
python3 evals.py --set evals/ --salida corrida-$(date +%F).json
python3 - <<'PY'
import json; r=json.load(open("corrida-2026-09-05.json"))
for t in ["extraccion","consulta","acta","permisos","trampa"]:
    c=[x for x in r if x["tipo"]==t]; print(f"{t:11s} {sum(x['ok'] for x in c)}/{len
    (c)}")
PY
```

Si no corre desde cero con un comando, no entra al curso.

Cuándo NO usar esto

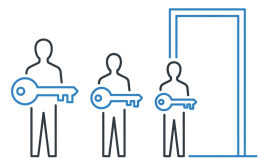
Un set de evals desde cero

- No use un modelo grande «como juez» sin casos con respuesta esperada: el juez también se equivoca, y se equivoca parecido.
- No escriba casos que el sistema ya pasa: los evals valiosos son los que fallan.

12.2 · Métricas por tipo de tarea

Métricas por tipo de tarea

«Precisión del 92 %» no significa nada sin decir de qué. Cada tipo de tarea tiene una métrica que corresponde a **lo que cuesta el error**.



`tablero.py` que lee las corridas (`corrida-*.json`) y produce una tabla por tarea con valor actual, valor anterior, tendencia y umbral. Salida en Markdown para pegar en el reporte mensual (11.2).

Cuándo NO usar esto

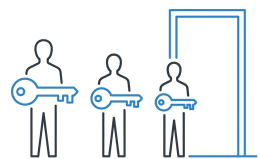
Métricas por tipo de tarea

- No use BLEU/ROUGE ni similitud de embeddings para juzgar respuestas de negocio: miden parecido, no corrección.
- No ponga umbrales sin haber medido el desacuerdo humano (7.4): no se le exige al sistema más de lo que dos personas acuerdan.

12.3 · Trazado multipaso y el bucle traza → eval → corrección

Trazado multipaso y el bucle traza → eval → corrección

Con evals que fallan y trazas de cada paso (11.2), el diagnóstico deja de ser adivinanza:



El bucle:

El comando, copiable

```
eval ev-031 falló → traza …tr_01J → paso 1: buscar_documentos devolvió [contrato
    -2026-014#3, #9]→
 la ficha #7 no estaba entre las 5 → recall (6.2) → ¿por qué? la consulta "plazo de
    garantía"
no se parece a la cláusula, que dice "término de la garantía"
```

Si no corre desde cero con un comando, no entra al curso.

Cuándo NO usar esto

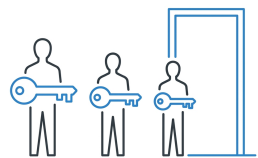
Trazado multipaso y el bucle traza → eval → corrección

- No corrija sin la traza «porque ya sé qué fue»: casi nunca es lo que uno cree (6.4).
- No acumule tres cambios antes de correr el set.

12.4 · Evals canario nocturnos y degradación

Evals canario nocturnos y degradación

El canario de 9.3 tenía tres pruebas. El **canario de evaluación** corre un subconjunto del set de evals (15 casos, los más sensibles: permisos, trampa, 5 facturas, 5 consultas) todas las noches, y compara con la noche anterior.



Monte el canario, y **provoque** una degradación (cambie la cuantización del modelo a Q3). Lo que debe ver: la alerta a las 4:15 con el eval que bajó. Luego revierta y verifique que vuelve a verde.

En una tabla

Cambio silencioso	Cómo se ve
Se actualizó el runtime o el driver	latencia cambia; a veces la calidad
Alguien cambió el modelo «para probar»	hash distinto (9.3) + evals bajan
El cliente añadió un proveedor con otro formato	extracción baja en ese proveedor
Se reindexó con otro embedding	recall cae
Cambió el proveedor de identidad	permisos fallan → alerta inmediata

El comando, copiable

```
# /etc/cron.d/el-tornillo (añadir)
15 4 * * *  app  timeout 1h /opt/tornillo/canario-evals.sh >> /var/log/tornillo/
    canario-evals.log 2>&1
```

Si no corre desde cero con un comando, no entra al curso.

Cuándo NO usar esto

Evals canario nocturnos y degradación

- No ponga los 60 casos en el canario nocturno: 15 sensibles bastan; los 60 se corren antes de cada cambio.
- No silencie una alerta «porque ya sé qué es»: se corrige o se ajusta el eval, nunca se apaga.

Entregables del módulo

- `evals/` con 60 casos, `evals.py`, y la primera corrida.
- `tablero.py` y el tablero con dos corridas.
- `labs/12-03/bucle.md` con los tres ciclos completos.
- `canario-evals.sh`, `evals/canario/` (15 casos), y las dos alertas (degradación y vuelta a verde).

Un set de evals desde cero, métricas por tarea, trazas multipaso y canarios nocturnos que avisan cuando algo se degrada.