Lectura y Extracción de Datos

De Facturas Eléctricas

Cesar Damian Celis
Maria Angel Lobon Gonzalo
Elisban Montañez Montalvo
Samuel Porcayo Fraustro

Indice

Desafio
Necesidad del modelo

Solucion
Cambios Propuestos

Arquitectura
Diseño de la arquitectura

Modelos y Herramientas
LLMs y Frameworks utilizados

Pruebas
Outputs de los Modelos

Evaluación
Mecanismos de Evaluación de los Modelos

Desafio

Alta Complejidad

Difícil lectura manual y el análisis eficiente de los datos clave.

Falta de Estandarización

Diferentes compañías presentan la misma información de formas distintas

Gestion Ineficiente

Difícil guardar un histórico sólo de los datos esenciales por orden y que estos datos sean fáciles de buscar

Solucion

Extraer y gestionar información clave de facturas eléctricas

Extracción Datos Importantes

Jupiter is the biggest planet of them

Detección de Anomalias

Detecta posibles errores en los datos

Optimización de Asesoría

Análisis de los datos más accesible

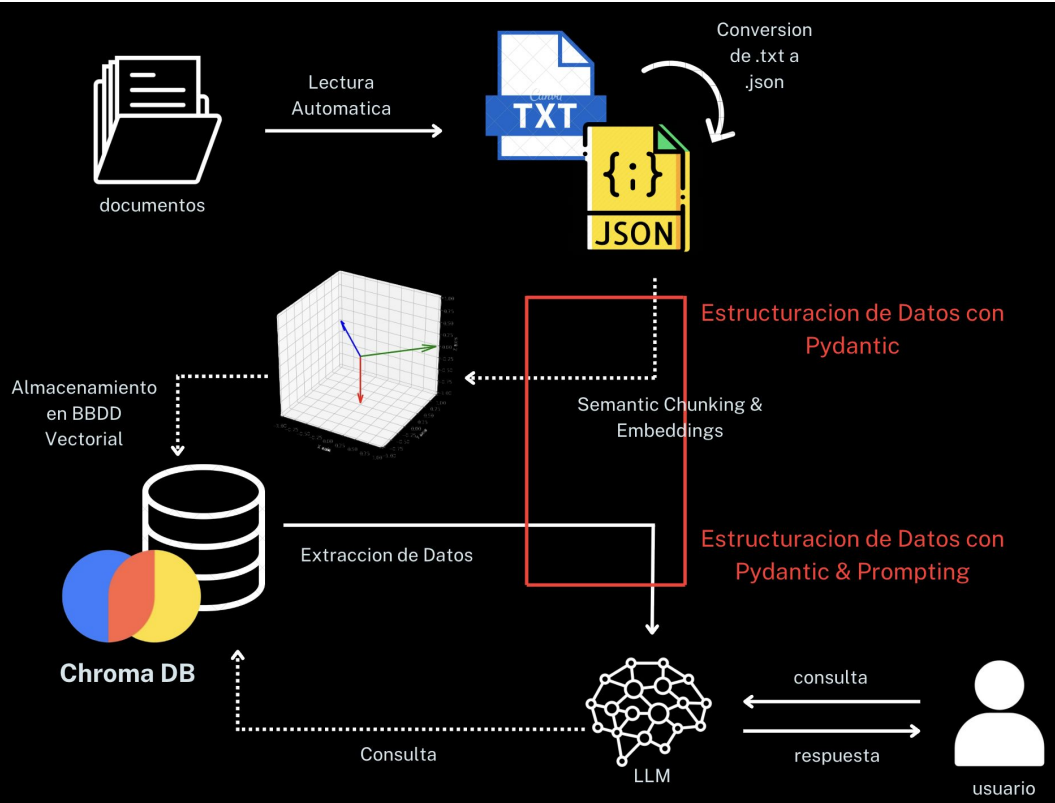
Lector de Datos

Explicaciones sin tecnicismos de los valores y precios de la factura

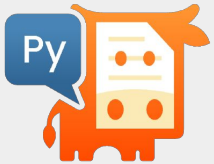
Gestión Mensual

Datos almacenados - busqueda facil

Arquitectura



Herramientas



Ingesta de Datos PDF

PyMuPDF

RegEx

Semantic Chunking

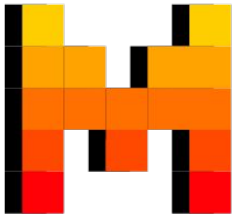
División del texto en secciones semánticas

Chroma

BBDD Vectorial

Almacenamiento de embeddings creados

all-MiniLM-L6-v2



LLM

Mixtral-8x7B-Instruct (Together API)

Pydantic

Validación y estructuración de los campos extraídos

Pydantic

```
# 3. Prompt para solicitar los datos estructurados al LLM
prompt_json = """
Extrae los siguientes datos de la factura eléctrica y devué

Devuelve el resultado en este formato JSON:

{
  "N factura:": "string",
  "fecha_emision": "YYYY-MM-DDTHH:MM:SS",
  "periodo_inicio": "YYYY-MM-DDTHH:MM:SS",
  "periodo_fin": "YYYY-MM-DDTHH:MM:SS",
  "consumo_total_kwh": float,
  "potencia_punta_kw": float,
  "potencia_valle_kw": float,
  "importe_total": float
}

Usa solo los datos que estén claramente presentes. Si no ap

Devuelve exclusivamente el JSON, sin explicaciones.
"""
```

1

Estructurar datos con clases (BaseModel)

Define una estructura de relevancia para el almacenamiento de datos

2

Validar tipos automáticamente

Al crear una instancia del modelo, Pydantic verifica que los datos coincidan con los tipos definidos.

3

Convertir datos al tipo correcto (ej. string → int)

Si el dato no es compatible, Pydantic lo transforma automáticamente al tipo esperado

Output de los dos Modelos

Obtuvimos el mismo resultado de datos con todos las facturas que se introdujeron en los modelos

```
Pregunta: ¿Cuál es el valor del campo 'fecha_emision'?
Valor extraído por el LLM (del JSON final): 2025-02-13T00:00:00

Pregunta: ¿Cuál es el valor del campo 'periodo_inicio'?
Valor extraído por el LLM (del JSON final): 2024-12-31T00:00:00

Pregunta: ¿Cuál es el valor del campo 'periodo_fin'?
Valor extraído por el LLM (del JSON final): 2025-01-31T00:00:00

Pregunta: ¿Cuál es el valor del campo 'consumo_total_kwh'?
Valor extraído por el LLM (del JSON final): 66.321

Pregunta: ¿Cuál es el valor del campo 'potencia_punta_kw'?
Valor extraído por el LLM (del JSON final): 6.928
```

Análisis de Resultados

Campo	Valor Real	Valor Predicho	¿Coincide?
numero_factura	P25CON005526043	P25CON005526043	✅ Sí
fecha_emision	2025-02-13T00:00:00	2025-02-13T00:00:00	✅ Sí
periodo_inicio	2024-12-31T00:00:00	2024-12-31T00:00:00	✅ Sí
periodo_fin	2025-01-31T00:00:00	2025-01-31T00:00:00	✅ Sí
consumo_total_kwh	66.321	66.321	✅ Sí
potencia_punta_kw	6.928	6.928	✅ Sí
potencia_valle_kw	6.928	6.928	✅ Sí
importe_total	50.65	50.65	✅ Sí

Se evaluó campo por campo comparando valores reales y extraídos.

- Para los valores reales se hizo un labelling manual para cada campo

Métricas alcanzadas:

- F1 Score > 0.9
- Recall > 0.9

se compararon los resultados obtenidos por dos configuraciones de modelo (MiniLM vs MPNet).