



**UNR** Universidad  
Nacional de Rosario



FACULTAD DE  
CIENCIAS EXACTAS,  
INGENIERÍA Y AGRIMENSURA

**UNIVERSIDAD NACIONAL DE ROSARIO**  
Facultad de Ciencias Exactas, Ingeniería y Agrimensura



**TECNICATURA UNIVERSITARIA EN INTELIGENCIA ARTIFICIAL**

Asignatura:

**NLP**

**TRABAJO PRÁCTICO FINAL**

**TEMA: Asistente Virtual**

**PROFESORES:**

jpmanson@gmail.com

alan.geary.b@gmail.com

constantinoferrucci@gmail.com

**INTEGRANTES:**

Santiago Giaveno

**FECHA:** 07.12.2025

---

**ÍNDICE:**

---

<b>TEMA</b>	<b>2</b>
<b>1   INTRODUCCIÓN</b>	<b>2</b>
<b>2   DESARROLLO Y DECISIONES TOMADAS</b>	<b>3</b>
<b>3   RESULTADOS</b>	<b>8</b>
<b>4   CONCLUSION</b>	<b>10</b>

## Introducción

En el siguiente trabajo práctico se desarrollará un asistente virtual especializado en información de una empresa de electrodomésticos. El mismo va a estar dividido en dos etapas, en la primera es la creación de un RAG a partir de la fuente de información proporcionada. Luego para la segunda etapa, se migró el RAG en un agente inteligente que pueda decidir qué herramientas utilizar para responder preguntas complejas, utilizando el Langchain con el paradigma ReAct.

## DESARROLLO Y DECISIONES TOMADAS

### Ejercicio 1 - RAG

LLM a utilizar durante todo el ejercicio 1.

#### Ollama

Seleccione Ollama como entorno de ejecución local por las siguientes razones:

- Facilidad de uso y despliegue: permite descargar, administrar y ejecutar distintos modelos con comandos simples, facilitando el proceso prueba entre opciones
- Ejecución local: evita depender de servicios externos o APIs, lo cual mejora la seguridad y reduce costos.
- Soporte amplio de modelos: Ollama ofrece un catálogo muy completo de modelos optimizados para uso local, con variantes en diferentes tamaños y cuantificaciones.

Durante esta etapa de experimentación se probaron modelos diversos utilizando Ollama, tales como:

- qwen2.5:7b
- qwen3:latest
- mistral-small
- qwen2.5:1.5b
- entre otros modelos de la misma familia y hasta de tamaños más grandes.

Elección de **Gemma 3** como modelo

- **Relación precisa entre calidad y rapidez:** Gemma 3 mostró un nivel de razonamiento suficiente para el dominio del trabajo práctico. Combinando un buen equilibrio entre precisión y velocidad.
- **Consistencia en lenguaje y formato:** En las pruebas, modelos como Qwen y Mistral tuvieron buen rendimiento, pero Gemma 3 produjo respuestas más estructuradas y fáciles de controlar mediante prompt.
- **Tamaño adecuado para un asistente especializado:** La variante 1B es extremadamente rápida y útil para tareas sencillas (clasificación, reformulación, limpieza de texto). La variante 4B ofrece una mejor precisión en consultas complejas sin un costo computacional excesivo.

#### Justificación de contenido en cada fuente de datos.

**Vectorial:** Decido usar los manuales de los productos, las faq y reseñas en la base de datos vectorial. La elección se basa en que todos estos elementos son textos no estructurados, extensos y muy contextuales. El contenido de un manual o una reseña no se puede representar fácilmente con columnas fijas, ya que incluye descripciones libres, instrucciones, advertencias, opiniones y explicaciones que requieren búsqueda semántica para que el modelo lo interprete correctamente. Las FAQ contienen información de procedimiento o técnicas también, que se beneficia de la recuperación semántica (embeddings), ya que las consultas van a estar en lenguaje natural y de manera muy diversa.

**Tabular:** En la base de datos tabular use productos.csv, que contiene toda información específica relacionada a cada producto. Estos datos presentan estructura fija, fácil de representar mediante: columnas, tipos definidos (numéricos, categóricos, booleanos), filtros exactos, consultas directas. Ordenar, comparar precios, filtrar por categoría, o devolver atributos específicos son operaciones propias de una base tabular.

**Grafos:** Aca use los siguientes csv: productos,tickets,inventario sucursales.

- **Productos** pueden estar relacionados con: categorías, sucursales donde hay stock, tickets o reclamos que los mencionan
- **Tickets** representan interacciones entre clientes y productos, y permiten análisis como: “¿Qué producto genera más reclamos?”, “¿Qué problemas están relacionados a qué modelos?”
- **Inventario de sucursales** permite responder cosas como: “¿Dónde hay stock del modelo, “¿Qué sucursales tienen más problemas con determinado producto?”

La representación en grafo **permite:** explorar conexiones, navegar relaciones, responder consultas complejas como:

- *“Mostrame productos con tickets asociados en sucursales del interior.”*
- *“Encontrá el producto con mayor cantidad de reclamos por categoría.”*

Estas consultas serían mucho más costosas o difíciles de modelar en una base tabular tradicional.

Para la base datos vectorial use ChromaDB ya que **es liviano y fácil de integrar** en entornos locales. Proporciona **consultas semánticas rápidas** y bien optimizadas, ideales para trabajar con manuales, FAQs y reseñas. Permite almacenar metadatos junto a los embeddings, algo fundamental para identificar el origen de cada chunk (producto, sección, modelo, etc.).

Como modelo de embedding use SentenceTransformers (multilingual-e5-small). **Multilingüe** dado que los textos estaban en español (y algunas consultas podrían venir en otros idiomas), era importante usar un modelo que soporte múltiples lenguas sin comprometer la precisión. **Ligero y rápido**, al ser un modelo “small”, es ideal para ser ejecutado localmente sin requerir GPU. Este modelo fue suficiente para capturar el significado de textos técnicos y reseñas informales, manteniendo un desempeño ágil dentro del pipeline.

Como text splitter, decidí crear funciones personalizadas para los datos, para así obtener chunks mas específicos y no perder información.

### Splitter para reseñas

Implementé una función para extraer automáticamente: nombre del producto, ID del producto (si existe), puntaje, texto completo de la reseña

Esto me permitió generar **metadatos homogéneos** y normalizados para todas las reseñas antes de convertirlas en embeddings.

Los manuales de electrodomésticos tenía estructura por **secciones** (por ejemplo: “## Seguridad”, “## Instalación”, “## Configuración”, etc.).

Por eso diseñe un splitter que: detecta solo encabezados de nivel ##, agrupa dentro de cada sección

todos los subtítulos ###, evita fragmentar pasos o advertencias internas y genera chunks de cada sección completos.

Luego, cada sección se divide en chunks más pequeños mediante un **chunker deslizando** con:  
`chunk_size = 500, chunk_overlap = 75`

Las FAQs ya vienen estructuradas como pares “pregunta–respuesta”. Mi proceso: concatenar cada pareja en un solo bloque de texto. Generar metadatos específicos como: tipo (“faq”), producto relacionado, categoría, pregunta original.

Luego para base de datos tabular, decidí usar pandas ya que eso me da un mejor control y facilidad para manejar los datos. Elegí Pandas porque: Permite **cargar, limpiar y transformar datos** de manera muy eficiente. Ofrece un control muy preciso sobre columnas, tipos de datos, valores nulos y filtros. Se integra fácilmente con Python, que es el lenguaje del resto del pipeline del asistente. Cuenta con funciones muy potentes para análisis descriptivo (min, max, unique, value\_counts, etc.).

Extraje la metadata de los productos como categorías, marcas y colores disponibles, rango de precios, potencia y peso, voltajes disponibles y campos nulos. Con esta información construí una base tabular que funciona como fuente de datos estructurados para el modelo.

Para la base de datos de grafo use neo4j ya que es un **modelo intuitivo para representar relaciones reales**.

Muchos datos del TP tienen una estructura inherentemente relacional, por ejemplo: productos vendidos por sucursales, tickets asociados a productos, inventario vinculado a cada tienda, categorías conectadas a cada producto. Neo4j permite modelar todo esto de manera directa mediante nodos y relaciones, sin la complejidad de joins repetidos o tablas intermedias.

**Excelente integración con Python:** A través del *Neo4j Python Driver* pude integrar la base fácilmente dentro del agente y ejecutar Cypher dinámico generado por el LLM.

En lo que es Clasificador de Intención Avanzado. Primero entreno una SVM con un dataset sintético creado con preguntas (con su respectiva etiqueta) en base a la fuente de información y otras no, para así tener un contexto general

Luego utilice un clasificador basado en LLM con el modelo gemma3:1b. . Usé un enfoque de **Few-Shot Prompting**, en el cual incluí ejemplos reales de preguntas y etiquetas para guiar al modelo

En base a las métricas obtenidas luego de testear ambos modelos de clasificación con las mismas preguntas y obteniendo sus respuestas, puedo concluir que el modelo entrenado no tiene malas métricas por ser entrenado de 0, con un accuracy 0.73. Pero el LLM ya tiene mejores métricas en general, algo esperable, con un accuracy de 0.83.

Por esta razón, el clasificador LLM fue el elegido como componente principal del pipeline del asistente virtual.

Para lo que es el pipeline de recuperación. Para consultas en bases de datos semánticas, implemente una búsqueda híbrida que combina la búsqueda semántica con una búsqueda por palabras clave, que en mi caso use BM25, luego aplique un mecanismo de reranks sobre los resultados para mejorar la relevancia de los chunks recuperados. Decido usar CrossEncoder para rerank, algo que nos permite ...

Para consultas en base de datos de grafos y tabulares, aplico funciones que a partir del prompt del usuario, generan un filtro pandas para la tabular y una query cypher para el grafo.

Luego, desarrollé una función que coordina todo el pipeline y hace las llamadas a un **LLM corriendo localmente en Ollama**. Decidí utilizar el modelo **Gemma** porque ofrece:

- buena relación costo–rendimiento para uso local,
- tiempos de inferencia bajos,
- buena performance en tareas de razonamiento ligero y clasificación de intención,
- compatibilidad con hardware moderado sin necesidad de GPU especializada.

Para terminar con el ejercicio 1, integro todos los componentes (clasificador, recuperadores y un llm generador) en un bucle conversacional. Utilizó langdetect, para poder detectar el idioma de la consulta y que el llm pueda responder en el mismo idioma, luego. También agregué un **mecanismo de memoria**, lo que permite conservar el contexto de la conversación en turnos sucesivos y ofrecer respuestas más consistentes. En caso de que el llm no encuentre información relevante, genera una respuesta en donde sugiere al usuario reformular su pregunta

## Ejercicio 2 - Evolución del RAG a un Agente Autónomo

Utilizó langchain para implementar un agente basado en el paradigma ReAct. Como herramientas (tools) utilizar las funciones del ejercicio uno (doc\_search, table\_search, graph\_search). Luego como adicional creó una función analytics\_tool en donde se genera un gráfico a partir de la consulta del usuario. El agente recibe la consulta del usuario, razona en pasos (Thought), ejecuta acciones con herramientas, observa resultados y finalmente genera la respuesta. Las herramientas encapsulan la lógica de las fuentes del Ejercicio 1.

### Herramientas (Tools)

#### 1. doc\_search(query, top\_k=10)

- Entrada: texto del usuario.
- Acción: realiza búsqueda híbrida (embeddings + BM25) en ChromaDB, combina resultados y aplica **CrossEncoder** para rerank.
- Salida: lista de chunks relevantes con metadata (origen, sección, score).
- Uso esperado: preguntas procedimentales, instrucciones técnicas, FAQs, reseñas, comparativas basadas en texto libre.

#### 2. table\_search(query)

- Entrada: prompt del usuario (o filtros ya parseados).

- Acción: convierte el prompt a un filtro Pandas (p. ej. condiciones en columnas, agregaciones). Ejecuta la consulta sobre productos.csv en memoria y devuelve resultados tabulares (y/o resumen estadístico).
- Salida: filas filtradas, agregaciones (promedio, conteo) y tabla en formato legible (CSV/tabla).
- Uso esperado: consultas de catálogo, disponibilidad por atributos, comparaciones numéricas y rangos.

### 3. **graph\_search(natural\_language\_query)**

- Entrada: un texto en NL.
- Acción: si recibe NL, genera dinámicamente la consulta Cypher (usando el LLM o templates). Ejecuta la consulta en Neo4j y devuelve nodos/relaciones relevantes.
- Salida: resultados estructurados (nodos y relaciones) y descripción textual de la subconsulta.
- Uso esperado: preguntas sobre relaciones, historial de tickets, trazabilidad entre sucursales/productos/tickets.

### 4. **analytics\_tool(sql\_query\_or\_nl, chart\_type='bar', save\_path=None)**

- Entrada: consulta, tipo de gráfico deseado.
- Acción: si es NL, genera la consulta SQL sobre la base SQL integrada. Ejecuta la consulta, genera un gráfico con **matplotlib** y lo guarda en el directorio de trabajo).
- Salida: path del archivo de imagen + breve resumen estadístico.
- Uso esperado: peticiones que requieren visualización (distribuciones, series temporales, comparativos).



# RESULTADOS

```

🔥 INICIANDO SISTEMA DE CONVERSACIÓN...

♦ Turno 1
👤 Usuario: Cual es el mantenimiento de un frigobar
Tipo de consulta: vectorial
🗨️ Bot: El mantenimiento de un Frigobar incluye las siguientes acciones:

### Mantenimiento Diario:
- Limpiar todas las partes en contacto con alimentos después de cada uso.
- Secar completamente antes de guardar.
- Verificar que el cable no tenga daños visibles.
- Limpiar la base con paño húmedo (NUNCA sumergir en agua).

### Mantenimiento Semanal:
- Limpiar las juntas de goma con agua tibia y jabón.
- Verificar que las cuchillas estén afiladas.
- Revisar que no haya residuos en orificios de ventilación.
- Limpiar el área del motor con pincel suave.
-----
♦ Turno 2
👤 Usuario: ¿Cuál es el precio de la freidora de aire y cuanto stock hay?
Tipo de consulta: tabular
🗨️ Bot: La información que tengo disponible incluye licuadoras con precios que van desde 259.42 USD hasta 2662.78 USD y stock que va desde 75 hasta 198 unidades. No tengo informac
-----
♦ Turno 3
👤 Usuario: ¿La garantía cubre daños por inundación?
Tipo de consulta: vectorial
🗨️ Bot: La garantía NO cubre daños por uso indebido, roturas accidentales o reparaciones no autorizadas. No especifica cobertura para daños por inundación. Sugiero que reformule s
-----
♦ Turno 4
👤 Usuario: ¿Cuántos tickets abiertos hay en este momento?
Tipo de consulta: grafo
🗨️ Bot: Hay 2000 tickets abiertos en este momento.
-----

```

```

♦ Turno 5
👤 Usuario: Hay en la sucursal de Buenos Aires stock de Freidora de Aire
Tipo de consulta: grafo
🗨️ Bot: En la sucursal de Buenos Aires tenemos stock de Freidora de Aire. Tenemos disponibilidad en las sucursales con stock de 24, 25 y 38 unidades.
-----
♦ Turno 6
👤 Usuario: ¿Que dicen las reseñas sobre el caloventor?
Tipo de consulta: vectorial
🗨️ Bot: Según las reseñas, el Caloventor (en sus diferentes versiones: Caloventor, Caloventor II y Profesional Caloventor II) tiene opiniones mixtas. Algunos usuarios lo considera
-----
♦ Turno 7
👤 Usuario: Gracias, y ¿cuál es el peso en kg del producto P0299?
Tipo de consulta: tabular
🗨️ Bot: El peso del producto P0299 es de 5.7 kg.
-----

```

```

🔍 VERIFICANDO ESTADO DE LA MEMORIA INTERNA
=====
[0] USUARIO: Cual es el mantenimiento de un frigobar...
[1] DOCS: [{'producto': 'Frigobar', 'modelo': None, 'categoria': 'Garantía', 'section': None, 'texto': '¿Cuánt...
[2] BOT: El mantenimiento de un Frigobar incluye las siguientes acciones:

### Mantenimiento Diario:
- Limpia...
[3] USUARIO: ¿Cuál es el precio de la freidora de aire y cuanto stock hay?...
[4] DOCS:
      nombre  precio_usd  stock
3      Compacto Licuadora      259.42      75
...
[5] BOT: La información que tengo disponible incluye licuadoras con precios que van desde 259.42 USD hasta 26...
[6] USUARIO: ¿La garantía cubre daños por inundación?...
[7] DOCS: [{'producto': 'Premium Picadora', 'modelo': 'P0014', 'categoria': 'Cocina - Preparación', 'section':...
[8] BOT: La garantía NO cubre daños por uso indebido, roturas accidentales o reparaciones no autorizadas. No ...
[9] USUARIO: ¿Cuántos tickets abiertos hay en este momento?...
[10] DOCS: [{'count(t)': 2000}]...
[11] BOT: Hay 2000 tickets abiertos en este momento....
[12] USUARIO: Hay en la sucursal de Buenos Aires stock de Freidora de Aire...
[13] DOCS: [{'i.stock_sucursal': 24}, {'i.stock_sucursal': 25}, {'i.stock_sucursal': 38}]...
[14] BOT: En la sucursal de Buenos Aires tenemos stock de Freidora de Aire. Tenemos disponibilidad en las sucu...
[15] USUARIO: ¿Que dicen las reseñas sobre el caloventor?...
[16] DOCS: [{'producto': 'Profesional Caloventor II', 'modelo': None, 'categoria': 'Problemas Comunes', 'sectio...
[17] BOT: Según las reseñas, el Caloventor (en sus diferentes versiones: Caloventor, Caloventor II y Profesion...
[18] USUARIO: Gracias, y ¿cuál es el peso en kg del producto P0299?...
[19] DOCS:
      peso_kg
298      5.7...
[20] BOT: El peso del producto P0299 es de 5.7 kg....

✅ Total de elementos en memoria: 21

```

## CONCLUSIÓN

El desarrollo de este asistente virtual me permitió entender y ver la eficacia de combinar arquitecturas de **Generación Aumentada por Recuperación (RAG)** con paradigmas de **Agentes Autónomos (ReAct)** para la resolución de consultas complejas como para una empresa de productos y soporte técnico de la mismo

La integración de grafos de conocimiento con búsqueda vectorial y herramientas analíticas, organizadas por un LLM local, establece una arquitectura sólida para sistemas de asistencia técnica empresarial moderna.

## CONCLUSIÓN

