



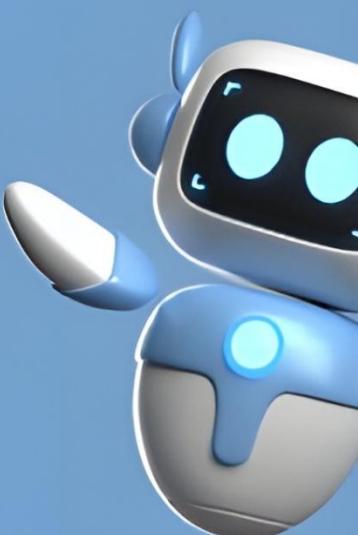
# **TRABAJO PRÁCTICO FINAL**

## **PROCESAMIENTO DEL LENGUAJE NATURAL**

**NOMBRE: GENARO CANCIANI**

**LEGAJO: C-7449/7**

**DOCENTES:**  
**CONSTANTINO FERRUCCI,**  
**ALAN GEARY,**  
**JUAN PABLO MANSON**



# Índice

1. Introducción: .....	3
2. Bases de datos:.....	3
2.1. Grafo: .....	3
2.2. Tabular:.....	4
2.3. Vectorial:.....	4
3. Modelos y Splitters: .....	4
3.1. LLM: .....	4
3.2. Splitters:.....	5
3.3. Embedding:.....	5
3.4. Reranker: .....	6
3.5. Clasificador: .....	6
4. Resultados de ejecución: .....	7
4.1. Ejercicio 1(RAG):.....	7
4.1.1. Pregunta 1: .....	8
4.1.2. Pregunta 2: .....	8
4.1.3. Pregunta 3: .....	8
4.1.4. Pregunta 4: .....	9
4.1.5. Pregunta 5: .....	10
4.1.6. Conclusiones Ejercicio 1:.....	11
4.2. Ejercicio 2(ReAct):.....	11
4.2.1. Pregunta 1: .....	11
4.2.2. Pregunta 2: .....	12
4.2.3. Pregunta 3: .....	12
4.2.4. Pregunta 4: .....	14
4.2.5. Pregunta 5: .....	14
4.2.6. Pregunta 6: .....	15
4.2.7. Conclusión ejercicio 2:.....	15
5. Mejoras posibles:.....	15
6. Bibliografía: .....	17

# 1. Introducción:

El presente trabajo final de la asignatura Procesamiento del Lenguaje Natural tiene como objetivo principal la implementación de un pipeline integral para el desarrollo de un asistente virtual orientado a una empresa de electrodomésticos. El proyecto abarca desde la extracción y preprocesamiento de datos hasta la construcción de un agente inteligente basado en el paradigma ReAct, capaz de interactuar con el usuario y consultar bases de datos dinámicamente.

Para ello, se utilizan diversas fuentes de información sintética con formatos heterogéneos: datos estructurados de ventas, productos y vendedores en CSV; manuales en Markdown; reseñas de usuarios en texto plano (TXT) y una base de conocimientos de preguntas frecuentes en JSON. Todo el sistema se apoya en Grandes Modelos de Lenguaje (LLMs) consumidos vía API para la comprensión y generación de respuestas.

Durante el primer ejercicio se generará la base del sistema de recuperación de información, definiendo la estructura de cada una de las bases de datos y creándolas. También se definirán las funciones necesarias para interactuar con las mismas, cerrando con la creación de un asistente conversacional con RAG.

En el segundo y último ejercicio, se utilizarán las herramientas creadas anteriormente y se definirán las necesarias para que el asistente pase a funcionar en base al paradigma ReAct. Esto le permitirá tomar decisiones y usar las herramientas a disposición de manera autónoma según lo que consulte el usuario.

Todo el trabajo de código se realizará en un entorno de Colab, usando la GPU que se ofrece de manera gratuita.

# 2. Bases de datos:

Se generaron, en base a los datos con los que se contaba, tres bases de datos, que funcionaron como la fuente de verdad para el asistente.

## 2.1. Grafo:

La Base de Datos de Grafos se implementó utilizando **Neo4j** con el objetivo fundamental de servir como una fuente de información detallada y de alto nivel de **tecnicismo** para el agente, ideal para responder a consultas sobre los componentes, compatibilidades o repuestos de cada uno de los productos. A diferencia de las bases de datos vectoriales y tabulares, el grafo se construyó exclusivamente a partir del procesamiento de los **manuales técnicos** (formato Markdown). La estructura modelada incluye nodos clave como **Producto**, **Componente** y **Procedimiento**, unidos por relaciones dirigidas como **[:TIENE\_COMPONENTE]**, **[:COMPARTIRREPUESTO]** y **[:COMPATIBLE\_CON]**. Esta topología relacional permite al agente resolver preguntas complejas de manera eficiente, como **la dependencia de piezas**, **la identificación de repuestos específicos** o **la verificación de compatibilidad** entre diferentes modelos de la marca.

## 2.2. Tabular:

La fuente principal para la consulta de información **cuantitativa y precisa** es la Base de Datos Tabular. Esta fuente se compone de un conjunto de **DataFrames de Pandas**, los cuales fueron cargados a partir de los archivos CSV originales (como ventas, productos, inventario y tickets). Debido a su formato estructurado, la Base Tabular es esencial para responder consultas que requieren **agregación, cálculo y reportes de negocio**. Esto incluye obtener cifras exactas de *stock*, calcular el *precio promedio* de un producto, realizar conteo de transacciones o filtrar datos por fecha y sucursal. La precisión del agente para responder preguntas como "*¿Cuántas unidades se vendieron de X?*" depende directamente de la correcta manipulación de estos **DataFrames**.

## 2.3. Vectorial:

La **Base de Datos Vectorial** constituye el repositorio de información **no estructurada y semi-estructurada** del sistema. Esta se generó mediante el procesamiento de los **manuales de uso** (Markdown) y las **reseñas de usuarios** (TXT), transformando sus contenidos textuales en vectores numéricos de alta dimensión. La función principal de esta base es proveer a los usuarios información de contexto amplio, incluyendo **descripciones detalladas de procedimientos de uso**, especificaciones técnicas, y un resumen de las **opiniones y el sentimiento** general del cliente respecto a los productos. Gracias a la búsqueda por similitud complementada con re-ranking avanzado, el agente puede recuperar fragmentos relevantes incluso cuando la consulta se formula de manera coloquial o conceptual.

Durante el proceso de creación de esta base de datos se probó utilizar las **preguntas frecuentes (Json)** en conjunto con las reseñas y los manuales. Sin embargo, la adición no generó mejoras, si no que empeoró y dificultó la consulta a la fuente. Esto porque la cantidad de preguntas es muy grande (3000), y cuenta con información que se puede obtener fácilmente o en los manuales o en las reseñas. Esto generaba mucho **ruido** y dificultaba la extracción de información específica. Por esa razón, se decidió que sería la única fuente de información que no se utilizaría en el trabajo de todas las disponibles.

## 3. Modelos y Splitters:

Durante la creación de las bases de datos generadas anteriormente se usaron una gran cantidad de herramientas. Entre ellas principalmente se encuentran modelos de embedding, el **LLM** y el reranker. También se utilizaron dos splitters para generar chunks de texto a vectorizar según el tipo de documento que se procesó.

### 3.1. LLM:

El mismo fue la herramienta más fundamental y transversal utilizada para el trabajo. No solo se uso como modelo conversacional, si no que también para generar código de manera automática, clasificar intención, hacer reformulaciones de consultas fallidas, etc.

Se decidió trabajar con API's de manera remota debido a las limitaciones que conllevaba correr los modelos de manera local, al no contar con hardware lo suficientemente potente como para alcanzar el rendimiento de los modelos que se ofrecen para usar de manera remota.

El modelo elegido fue *gemini-2.5-flash*, ofrecido por Google. Es uno de los modelos más potentes que se puede acceder mediante la API de Gemini, quedando por detrás solo de *gemini-2.5-pro* en las opciones que se podían acceder gratuitamente. La razón principal por la que se eligió este modelo es porque era la alternativa más balanceada en términos de funcionamiento y costo. El mismo permitía ser consultado 250 veces al día y 10 veces por minuto. Si bien esto presentó una limitación (que se tratará a profundidad en el apartado de mejoras), fue lo suficientemente bueno como para permitir realizar cada una de las tareas necesarias, dando un rendimiento y unos resultados acordes a lo esperado teniendo en cuenta sus limitaciones.

### 3.2. Splitters:

Se usaron dos splitters, cada uno para procesar un tipo de documento diferente:

- **Splitter recursivo:** Se utilizó un **RecursiveCharacterTextSplitter** con `chunk_size=2000`, `chunk_overlap=200` y separadores jerarquizados porque las reseñas poseen un formato homogéneo y relativamente breve, por lo que resulta conveniente mantener cada una íntegra dentro de un mismo fragmento para preservar su coherencia semántica. El tamaño de 2000 caracteres garantiza que las reseñas no se fragmenten, mientras que el solapamiento de 200 caracteres permite conservar continuidad contextual cuando el splitter procesa textos más extensos o múltiples reseñas en conjunto. Los separadores `["\n\n", "\n", ". ", " ", ""]` permiten dividir primero por saltos de línea y oraciones (respectando la estructura natural del texto) y recurrir a separaciones más finas solo si es necesario, evitando cortes dentro de frases. En conjunto, esta configuración produce chunks completos, limpios y semánticamente coherentes, optimizando la calidad del proceso de embedding y la posterior recuperación de información.
- **Markdown Splitter:** Se empleó un **MarkdownHeaderTextSplitter** configurado con encabezados #, ## y ### porque los manuales técnicos presentan una estructura altamente estandarizada y repetitiva, donde las secciones, subsecciones y procedimientos están claramente delimitados mediante títulos en formato Markdown. Este splitter permite aprovechar esa organización natural dividiendo el documento exactamente en los bloques semánticos correspondientes a “título principal”, “sección” y “subsección”, evitando cortes arbitrarios dentro de procedimientos, listas o explicaciones técnicas. Al segmentar según la jerarquía real del documento, se obtiene un conjunto de chunks limpios, coherentes y temáticamente bien diferenciados, lo que mejora la calidad de los embeddings y permite que el agente recupere información precisa sobre especificaciones, componentes, pasos de uso, fallas y soluciones. Dado que todos los manuales comparten un formato muy similar, esta estrategia resulta consistente, estable y óptima para su procesamiento.

### 3.3. Embedding:

Se seleccionó el modelo **intfloat/multilingual-e5-base** como generador de embeddings para la base de datos vectorial debido a que ofrece un equilibrio óptimo entre rendimiento, costo computacional y compatibilidad multilingüe. Durante las pruebas comparativas, la versión **small** del modelo mostró limitaciones claras: producía representaciones menos precisas y conducía a recuperaciones incorrectas en búsquedas semánticas, especialmente en consultas

más complejas o con términos implícitos. Por otro lado, la versión **large**, si bien mejoraba la calidad de los embeddings, resultaba demasiado costosa en tiempo y recursos, lo que afectaba la eficiencia del sistema sin aportar una mejora considerable. En cambio, la variante **base** proporcionó resultados consistentes, mayor precisión en la recuperación y un rendimiento adecuado en el entorno de Colab. Adicionalmente, al ser un modelo **multilingüe**, permite manejar sin problemas texto en español y conserva robustez frente a variaciones lingüísticas, sinónimos y diferencias regionales.

### 3.4. Reranker:

Para las consultas a la base de datos vectorial se empleó una estrategia híbrida que combina BM25 con búsqueda semántica y un sistema de **reranking** basado en *BAAI/bge-reranker-v2-m3*, un modelo diseñado para evaluar con gran precisión la relevancia entre pares de textos. Este reranker se basa conceptualmente en la familia de modelos tipo **BERT**, aprovechando su arquitectura de **cross-encoder**, donde ambos textos (consulta y candidato) se procesan juntos para capturar interacciones semánticas finas. Gracias a esta arquitectura inspirada en BERT, el modelo entiende matices, inferencias y relaciones profundas, permitiendo distinguir entre resultados superficialmente similares y aquellos verdaderamente relevantes. Además, es un modelo multilingüe altamente optimizado, capaz de trabajar en español sin pérdida de calidad. En conjunto, la búsqueda híbrida más este reranker avanzado proporciona un sistema de recuperación robusto, preciso y especialmente eficaz en escenarios donde la relevancia semántica es crítica.

### 3.5. Clasificador:

Se requería generar **dos clasificadores** en el primer ejercicio, uno con LLM y otro con modelos de clasificación clásica. La idea era entrenarlos con **datos sintéticos** para poder predecir a qué base de datos tenía que consultar el modelo según la query del usuario. Se procuró que los datos generados estuvieran **balanceados** para no sesgar la decisión del modelo a una clase en particular.

Estos son algunos ejemplos de consultas artificiales generadas para el entrenamiento y la prueba:

- "¿Cuántas ventas hubo ayer?", "tabular"
- "¿Qué opinan los usuarios del lavarropas?", "vectorial"
- "¿Qué componentes tiene el motor?", "grafo"

En cuanto a los modelos, el primero fue una **regresión logística**, entrenada con consultas como las anteriores. El modelo conseguido no dió buenos resultados. Esto se debe principalmente a la poca cantidad de datos utilizados para el entrenamiento. Al probar la generación masiva de datos sintéticos, se generaba una situación en la que el modelo se entrenaba y se testeaba con datos demasiado similares, generando un claro **overfitting**.

Salida del **reporte de clasificación** de sk-learn:

--- REPORTE DE CLASIFICACIÓN (Test Set) ---				
	precision	recall	f1-score	support
grafo	1.00	0.33	0.50	3
tabular	0.50	0.67	0.57	3
vectorial	0.50	0.67	0.57	3
accuracy			0.56	9
macro avg	0.67	0.56	0.55	9
weighted avg	0.67	0.56	0.55	9

Por el lado del clasificador con LLM, el mismo generó resultados perfectos al ser testeado con el mismo conjunto de prueba que la regresión logística. Se utilizó el mismo modelo que se utiliza para la mayoría de las tareas, *gemini-2.5-flash*.

Salida del **reporte de clasificación** de sk-learn para clasificador con LLM:

--- B. MODELO LLM (GEMINI) ---				
	precision	recall	f1-score	support
grafo	1.000	1.000	1.000	3
tabular	1.000	1.000	1.000	3
vectorial	1.000	1.000	1.000	3
accuracy			1.000	9
macro avg	1.000	1.000	1.000	9
weighted avg	1.000	1.000	1.000	9

Como las métricas del LLM superaron ampliamente a las métricas de la regresión, se optó por utilizar el primero para el sistema RAG.

## 4. Resultados de ejecución:

Se realizaron una cantidad exhaustiva de **pruebas** para ambos ejercicios. La idea en ambos casos era simular una interacción posible entre un usuario y el asistente en un entorno de producción real, logrando probar todo el potencial del chatbot generado a profundidad. Ambos asistentes cuentan con capacidad de **memoria** y de **comprender el contexto** de la conversación, implementado de maneras diferentes.

### 4.1. Ejercicio 1(RAG):

El asistente con RAG engloba todas las herramientas que se habían generado hasta ese punto (bases de datos y funciones para consultarlas). Además, se implementó un sistema (usando el mismo LLM) para **reformular las consultas** del usuario para que sea más óptima cuando se consulte de manera directa a la base de datos. También se implementó una pequeña función que **detecta si el mensaje del usuario es un saludo**, evitando así una búsqueda innecesaria a alguna base de datos.

Para la prueba del sistema de RAG se generó una conversación que simula una interacción en producción. La misma cuenta con 5 mensajes de parte del usuario, procurando que cada uno obtenga una respuesta de cada una de las bases de datos.

Conversación simulada:

```
consultas_prueba = [
    "Hola, ¿qué tal?", 
    "¿Cuántas licuadoras vendimos en total?", # Tabular
    "¿Con qué productos es compatible?", # Grafo
    "Búscame reseñas sobre si hace mucho ruido", # Vectorial
    "Tiene algún error particular que sea peligroso?", # Vectorial
]
```

#### 4.1.1. Pregunta 1:

La pregunta “Hola, ¿qué tal?” se usó para probar el **sistema de detección de saludo** del agente. Resultado:

```
Usuario: 'Hola, ¿qué tal?'
🤖 Asistente: ¡Hola! Soy tu asistente de TechHome. ¿En qué puedo ayudarte?
-----
```

Se ve como el sistema logra detectar el saludo y no hace una búsqueda innecesaria.

#### 4.1.2. Pregunta 2:

La siguiente consulta, “¿Cuántas licuadoras vendimos en total?” buscaba probar las capacidades de la recuperación a partir de la base de datos tabular. Resultado:

```
Usuario: '¿Cuántas licuadoras vendimos en total?'
Esperando 7s... Listo.
    📈 Query Limpia: '¿Cuántas licuadoras vendimos en total?' -> 'Total de licuadoras vendidas'
    Reformulado: 'Total de licuadoras vendidas'
    Procesando 1 consultas con LLM...
        [1/1] Query: 'Total de licuadoras vendidas...' ✅ tabular
    Intención: TABULAR
    Buscando en Tabla... Esperando 7s... Listo.
    Hecho.
    Contexto: DATO OFICIAL BD: El resultado numérico exacto para 'Total de licuadoras vendidas' es: 700...
    Esperando 7s... Listo.
    🤖 Asistente: El total de licuadoras vendidas es 700.
-----
```

La consulta es exitosa, se logra obtener el resultado esperado que es un numero entero proveniente de la base tabular. Se puede observar también el funcionamiento de la reformulación, que mejora el resultado de la búsqueda.

#### 4.1.3. Pregunta 3:

La query “¿Con qué productos es compatible?” tiene dos fines, generar una consulta a la **base de grafos** y probar si el agente cuenta con la capacidad de comprender el contexto y de usar la **memoria**, al no mencionar de manera directa la licuadora. Resultado (resumido por temas de espacio):

Consultas ejecutadas:

```
1) MATCH (p:Producto) WHERE toLower(p.nombre) CONTAINS 'licuadora'
```

```
MATCH (p)-[:COMPATIBLE_CON]->(compatibleProduct:Producto)  
2) MATCH (p:Producto) WHERE toLower(p.nombre) CONTAINS 'licuadora'  
MATCH (p)-[:TIENE_COMPONENTE]->(c:Componente)-[:USADO_EN]->(compatibleProduct)
```

Productos recuperados:

- Sandwichera 3000 (P0064)
- Super Freezer (P0102)
- Profesional Batidora de Mano (P0017)
- Profesional Mixer (P0024)

Mensaje del agente:

 Asistente: Lamento, pero la información proporcionada no especifica cuáles de los productos listados son compatibles específicamente con una licuadora.

La información recuperada incluye los siguientes productos que son catalogados como "compatibleProduct", pero no se detalla con qué son compatibles:

- \* Sandwichera 3000 (marca ChefMaster, id P0064)
- \* Super Freezer (marca TechHome, id P0102)
- \* Profesional Batidora de Mano (marca TechHome, id P0017)
- \* Profesional Mixer (marca Generica, id P0024)

Si la pregunta se refiere a alguno de estos productos o a otro tipo de compatibilidad, por favor, házmelo saber para poder ayudarte mejor.

El resultado es interesante. Por un lado, se confirma que el sistema **tiene capacidad de memoria**, al poder generar la búsqueda deseada contra la base correcta sin que se deba mencionar de manera explícita el producto. Sin embargo, la primera consulta Cypher falló, por lo que se tuvo que generar una segunda query con la misma función. En el segundo caso el agente consiguió resultados. Sin embargo, no dió una respuesta muy segura para el usuario. Si bien el resultado de esta consulta no es ideal, se logró dar información al usuario y usar las capacidades de memoria del asistente.

#### 4.1.4. Pregunta 4:

La query “Búscame reseñas sobre si hace mucho ruido” está orientada a ver si el sistema logra obtener resultados sobre reseñas de productos particulares. De vuelta se hace uso del contexto para entender de qué producto se habla. Resultado (resumido por temas de espacio):

→ Clasificación: VECTORIAL

→ Búsqueda semántica: "Licuadora ruido reseñas"

Reseñas recuperadas:

- 2024-05-08 | Emma\_Benítez (Neuquén) | Licuadora P0005 | Puntaje 2/5

Comentó que es "muy ruidoso" y "complicado".

- 2025-01-11 | Nicolás\_Ramírez (Mendoza) | Licuadora P0006 | Puntaje 1/5

Indicó que es "muy ruidoso e inestable".

- 2025-06-22 | Pablo\_Ramírez (San Luis) | Licuadora P0002 | Puntaje 4/5

Destacó que es "silencioso", además de liviano e intuitivo.

Salida del asistente:

 Asistente: Claro, aquí tienes las reseñas sobre las licuadoras en relación al ruido:

\* \*\*\*Emma\_Benítez\*\*\* (2024-05-08, Neuquén) puntuó con \*\*\*2/5\*\*\* la \*\*\*Licuadora (P0005)\*\*\* y comentó que es "muy ruidoso" y "complicado", no recomendándola.

\* \*\*\*Nicolás\_Ramírez\*\*\* (2025-01-11, Mendoza) puntuó con \*\*\*1/5\*\*\* la \*\*\*Licuadora (P0006)\*\*\* y mencionó que es "muy ruidoso e inestable", indicando que no volvería a comprarla.

\* \*\*\*Pablo\_Ramírez\*\*\* (2025-06-22, San Luis) puntuó con \*\*\*4/5\*\*\* la \*\*\*Licuadora (P0002)\*\*\* y destacó positivamente que es "silencioso", además de liviano e intuitivo.

Se ve como el agente hace nuevamente uso del contexto y vuelve a hacer una consulta correcta a la base vectorial sin que el producto se mencione directamente. La respuesta es exitosa, ya que se obtuvieron datos de varias de las reseñas y el asistente se las enseñó al usuario correctamente.

#### 4.1.5. Pregunta 5:

La última pregunta “¿Tiene algún error particular que sea peligroso?” sirve para ver como responde la base vectorial ante una consulta un poco más ambigua, teniendo en cuenta que la información de la respuesta podría estar en una reseña o en algún manual. Resultado (resumido):

Consultas ejecutadas:

→ Clasificación: VECTORIAL

→ Búsqueda semántica: "Errores peligrosos licuadoras"

Resultados recuperados:

- 2025-01-21 | Santiago\_Gómez (Tierra del Fuego) | Licuadora P0007 | Puntaje 4/5

La reseña es completamente positiva y no menciona fallas, problemas ni riesgos de seguridad.

Salida del asistente:

 Asistente: Hola,

Revisando la información proporcionada, no encuentro detalles específicos sobre "errores peligrosos" relacionados con licuadoras. La reseña del usuario Santiago\_Gómez sobre la Licuadora (P0007) es muy positiva y no menciona ningún tipo de problema o error.

La lista de posibles causas y soluciones se refiere a un "Elite Deshidratador" (P0143), no a una licuadora, por lo que no puedo aplicarla directamente a tu consulta.

Por lo tanto, no tengo información que responda directamente a tu pregunta sobre errores peligrosos en licuadoras. Si tienes más detalles o si te refieres a otro producto, por favor, házmelo saber para poder ayudarte mejor.

En esta ejecución, el sistema no encontró evidencia de “errores peligrosos” asociados a licuadoras dentro del corpus analizado. La única reseña recuperada describe un funcionamiento correcto y destaca aspectos positivos del producto. Esto demuestra que el asistente no alucina si no encuentra la información exacta que se le pide.

#### 4.1.6. Conclusiones Ejercicio 1:

En general, el asistente **generó respuestas coherentes** e hizo uso de su capacidad de usar el contexto para generar consultas adecuadas a las bases de datos. En ningún caso el agente falló a la hora de clasificar la base de datos o de hacer la búsqueda. Por lo tanto, se concluye que el resultado es **positivo** y que la **robustez** del sistema de RAG es aceptable.

### 4.2. Ejercicio 2(ReAct):

El ejercicio 2 involucra el paradigma ReAct para poder crear un agente con una cantidad considerable de autonomía. El mismo hace uso de las mismas herramientas que se usaron antes para consultar las bases de datos, y además añade una nueva: “generate\_analytics”.

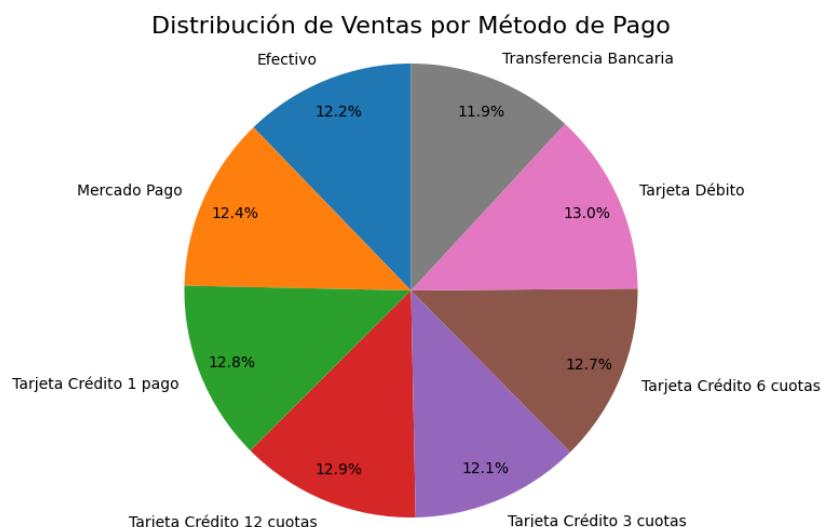
La misma tiene el objetivo de generar gráficos informativos obtenidos de la base tabular.

El asistente seguirá reglas estrictas de pensar, elegir la herramienta adecuada, aplicarla y finalmente darle un resultado al usuario.

En este caso se generaron 6 pruebas diferentes para observar el comportamiento del agente.

#### 4.2.1. Pregunta 1:

La primera consulta “Dame un gráfico de torta sobre la distribución de los medios de pagos en base de las ventas realizadas” prueba las capacidades del agente de **generar gráficos** relativamente complejos solo con una query. Resultado:



Mensaje del agente:

RESPUESTA: El gráfico de torta sobre la distribución de los medios de pago en base a las ventas realizadas ya se ha generado y mostrado en pantalla. Puedes ver la proporción de cada medio de pago en el total de las ventas.

El resultado es el esperado, tanto para la generación del gráfico como para la respuesta. Se ve como el agente logró por su cuenta discernir cuál era la herramienta que debía usar.

#### 4.2.2. Pregunta 2:

La segunda consulta, “¿Como hago un smoothie en la Licuadora?” busca evaluar la capacidad del agente para utilizar la herramienta *doc\_search*, que sirve para la búsqueda vectorial. Resultado (resumido):

```
> Entering new AgentExecutor chain...
Action: search_documents
Action Input: "cómo hacer un smoothie en La Licuadora" Buscando: 'cómo hacer un smoothie en la licuadora' | Filtro: None
[Fuente: manual_P0004_Compacto_Licuadora.md]
**Dificultad:** Fácil | **Tiempo:** 3-5 minutos
**Pasos:**
1. Lavar y cortar las frutas en trozos medianos (2-3 cm)
2. Colocar los ingredientes líquidos primero (Leche, yogurt, jugo)
3. Agregar las frutas y hielo en la jarra
4. Cerrar la tapa asegurándose del cierre hermético
5. Comenzar en velocidad baja (nivel 1-2) por 10 segundos
6. Aumentar gradualmente a velocidad alta (nivel 4-5)
7. Procesar durante 45-60 segundos hasta textura homogénea
8. Usar función PULSE para romper trozos grandes si es necesario
9. Verificar consistencia y procesar 10-15 seg adicionales si hace falta
10. Apagar y desconectar antes de retirar la jarra
11. Servir inmediatamente para mejor sabor y textura
```

Se ve uno de los tres manuales que encuentra el agente.

Mensaje:

 RESPUESTA: Para hacer un smoothie en la licuadora, sigue estos pasos:

1. Lava y corta las frutas en trozos medianos (2-3 cm).
2. Coloca los ingredientes líquidos primero (leche, yogurt, jugo) en la jarra.
3. Agrega las frutas y hielo.
4. Cierra la tapa asegurándose de que esté hermética.
5. Comienza en velocidad baja (nivel 1-2) por 10 segundos.
6. Aumenta gradualmente a velocidad alta (nivel 4-5).
7. Procesa durante 45-60 segundos hasta obtener una textura homogénea.
8. Si es necesario, usa la función PULSE para romper trozos grandes.
9. Verifica la consistencia y procesa 10-15 segundos adicionales si hace falta.
10. Apaga y desconecta la licuadora antes de retirar la jarra.
11. Sirve inmediatamente para disfrutar del mejor sabor y textura.

La respuesta del agente es muy acertada. Logró encontrar más de un manual que explica como hacer lo que se le pidió en diferentes productos, y da una respuesta final sobre la licuadora.

#### 4.2.3. Pregunta 3:

La consulta 3 es “Dame reseñas sobre ella”. De vuelta se busca probar las capacidades de memoria del agente. Resultado:

Imagen de la primera iteración del agente:

```

> Entering new AgentExecutor chain...
Action: search_documents
Action Input: reseñas de usuarios sobre la licuadora. Buscando: 'reseñas de usuarios sobre
[Fuente: manual_P0143_Elite_Deshidratador.md]
**Posibles causas:**
- Sobreexposición a la luz solar
- Temperatura ambiente demasiado alta
- Motor sobrecalentado
- Acumulación de residuos en componentes móviles
- Desgaste de componentes internos
**Soluciones:**
1. Reducir la cantidad de alimentos
2. Procesar en tandas más pequeñas
3. Dejar enfriar el motor 15-20 minutos
4. Limpiar todas las piezas móviles
5. Si es recurrente, solicitar revisión técnica
---

[Fuente: manual_P0143_Elite_Deshidratador.md]
**Posibles causas:**
- Junta de goma deteriorada o mal colocada
- Recipiente con rajaduras
- Sobrelleido del recipiente
- Tapa no cerrada correctamente
- Rosca de la cuchilla floja
**Soluciones:**
1. Inspeccionar y reemplazar juntas dañadas
2. Verificar que no haya grietas en el recipiente
3. No llenar más allá de la línea MAX
4. Asegurar cierre hermético de la tapa
5. Apretar la base de cuchillas (con el aparato apagado)

```

Se ve como hace la consulta correcta pero encuentra manuales, que no es lo que está buscando.

Pensamiento del agente:

Thought: The user is asking for "reseñas" (reviews) about the "Licuadora" from the previous turn. My previous attempt to find reviews using `search\_documents(query='reseñas de usuarios sobre la licuadora')` returned troubleshooting information from a dehydrator manual, which is not relevant to the request for reviews about the blender. I need to try again to find actual reviews for the licuadora. It's possible the system doesn't have explicit "reviews" documents, or my query needs to be more precise. However, the tool description explicitly states it's for "RESEÑAS de usuarios". I will try a slightly different query to see if it yields better results, focusing on the product type.

Resultado de segunda búsqueda:

#### Consultas ejecutadas:

→ Clasificación: VECTORIAL

→ Búsqueda semántica: "Opiniones de usuarios sobre la licuadora"

#### Resultados recuperados:

- 2024-12-03 | Laura\_Domínguez (Santa Cruz) | Licuadora P0007 | Puntaje 4/5  
Muy buena calidad, elegante, duradera y fácil de usar. La recomienda.
- 2025-06-22 | Pablo\_Ramírez (San Luis) | Licuadora P0002 | Puntaje 4/5  
Funciona de 10, liviana, intuitiva y silenciosa. Su familia está muy satisfecha.
- 2025-06-25 | Tomás\_González (Córdoba) | Licuadora P0005 | Puntaje 5/5  
Muy buena calidad, precisa y duradera. La recomienda ampliamente.

Salida del asistente:

 RESPUESTA: Aquí tienes algunas reseñas de usuarios sobre la licuadora:

\* \*\*\*Laura\_Domínguez (2024-12-03, Santa Cruz):\*\*\* Muy buena calidad, elegante, duradera y fácil de usar. La recomienda. (Puntaje: 4/5)

\* \*\*\*Pablo\_Ramírez (2025-06-22, San Luis):\*\* Funciona de 10, muy satisfecho. Es liviana, intuitiva y silenciosa. Su familia está muy contenta. (Puntaje: 4/5)

\* \*\*\*Tomás\_González (2025-06-25, Córdoba):\*\* Muy buena calidad, precisa y duradera. La recomienda. (Puntaje: 5/5)

Este caso es el más interesante hasta ahora. Se ve en su máxima capacidad la **potencia** del agente ReAct. Primero, **reconoció el contexto** sin que se mencionara el producto de manera explícita. Luego, obtuvo una información que no le era útil de la base vectorial sobre manuales de otros productos. Eso lo llevó a **pensar** y a darse cuenta de dos cosas, que la información que obtuvo **no era relevante**, y que debe volver a intentar, ya que la herramienta le da acceso a reseñas. Finalmente, consigue reformular la consulta y devolver una respuesta adecuada.

#### 4.2.4. Pregunta 4:

La pregunta 4, “¿Cuál es el precio promedio de los productos de cocina?” busca probar la capacidad del agente de consultar la base tabular. Resultado:

```
> Entering new AgentExecutor chain...
Action: search_table
Action Input: precio promedio de productos de cocina1412.248775510204Final Answer:

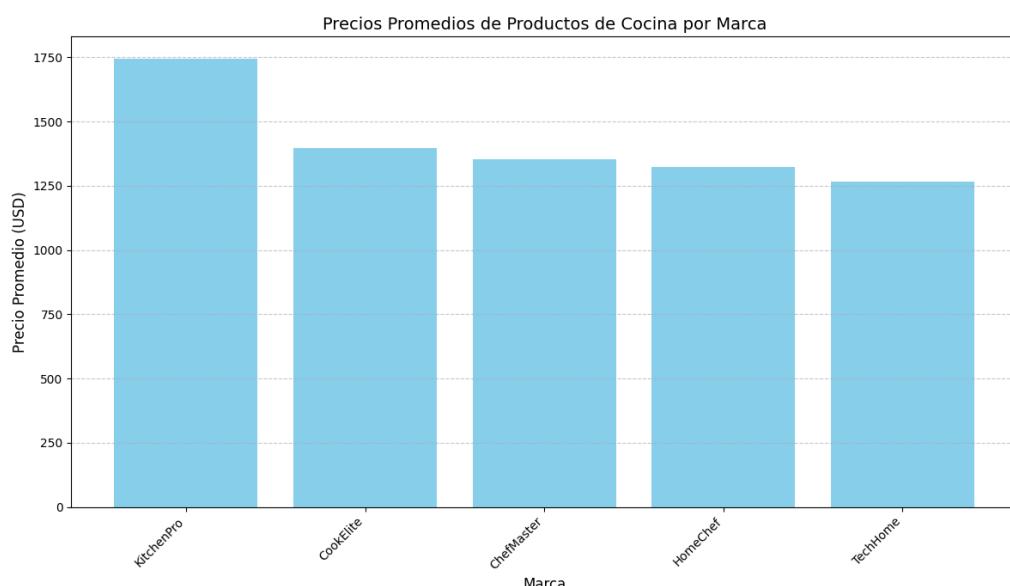
> Finished chain.

🤖 RESPUESTA: El precio promedio de los productos de cocina es de 1412.25.
```

Consigue responder correctamente.

#### 4.2.5. Pregunta 5:

De vuelta se pide un gráfico con la consulta "Mostrame en un gráfico de barras una comparación de los precios promedios de la categoría de cocina por cada marca.". Resultado:



Mensaje del agente:

 **RESPUESTA:** El gráfico de barras que compara los precios promedios de la categoría de cocina por cada marca ya se ha generado y mostrado en pantalla. Puedes ver la distribución de los precios promedio para cada marca dentro de la categoría de cocina.

Logra generar el gráfico correctamente.

#### 4.2.6. Pregunta 6:

La última consulta ("¿Qué productos tienen el motor como componente?") busca probar la capacidad del agente con consultas a la base de grafos. Resultado (resumido, la respuesta original lista muchos más productos):

 **RESPUESTA:** Los productos que tienen el motor como componente son:

- \* Lavarropas Automático (WashPro)
- \* Pro Ventilador de Pie (EcoClima)
- \* Digital Horno Eléctrico (CookElite)

La respuesta que da el agente es correcta, listando una gran cantidad de productos.

#### 4.2.7. Conclusión ejercicio 2:

El agente demuestra una **gran robustez** y da buenos resultados para cada una de las pruebas realizadas. Se pudo ver en cada una de las pruebas una capacidad diferente. El asistente logra **entender el contexto**, hacer uso de la **memoria**, **reconocer que herramienta usar** para cada caso y **replantearse usar otras herramientas** si el resultado no fue bueno en el primer intento.

Se concluye por lo tanto que se alcanzó el objetivo profundamente con los resultados obtenidos.

### 5. Mejoras posibles:

Si bien el resultado que se vio en las pruebas para ambos ejercicios fue positivo, existen limitaciones del sistema generado que se podrían corregir. Se mencionan algunas de las mejoras posibles:

- **Optimización de tiempo:** La mayor limitación del agente es el tiempo que tarda en generar las respuestas. En ambos ejercicios hubo que hacer pausas grandes (hasta de un minuto) para procurar no sobrepasar la cantidad límite de peticiones que permite la API de *Gemini*. Para evitar que tarde tanto, se podrían haber aplicado funciones en ambos ejercicios que manejen el tiempo y la cantidad de consultas en relación a los límites, modificándose de manera dinámica y optimizando el tiempo de respuesta.
- **Mejora de modelo:** La mayoría de los problemas con los que hubo que enfrentarse durante el desarrollo del trabajo tienen que ver con la potencia del modelo y la limitación de uso. Por ejemplo, con un mejor modelo, probablemente la base de datos de grafo podría haberse formado de manera más precisa, y los resultados de la misma podrían haber mejorado de manera considerable.
- **Caché semántico:** Por la misma razón que se mencionó antes del tiempo, implementar un sistema de caché, que devuelva una respuesta cargada en memoria si la consulta del

usuario es similar o casi igual, podría reducir mucho el tiempo de respuesta, manteniendo la calidad de la misma.

- **Corrección propia para base de datos tabular:** Para el caso de la base de datos de grafo se implementó un sistema que permite que se haga una nueva consulta si la primera falló. Agregar esa lógica para todas las funciones hubiera aumentado la robustez del modelo. No se aplicó por limitaciones mencionadas anteriormente (límite de API).

## 6. Bibliografía:

- [Neo4j Documentation: Graph Database Modeling and Usage.](#)
- [Límites de frecuencia-Gemini API](#)
- [Langchain-Agent docs](#)
- [Text Embeddings by Weakly-Supervised Contrastive Pre-training](#)
- [Hugging Face-Embedding Leaderboard](#)
- [LMArena-leaderboard](#)