



IIC3103 - Taller de Integración

Departamento Ciencia de la Computación
Escuela de Ingeniería
Pontificia Universidad Católica

Enunciado Tarea 3

Objetivo

El objetivo de esta tarea es desarrollar una aplicación web que permita a los usuarios resolver sus dudas sobre su Tesla, utilizando técnicas de **Retrieval-Augmented Generation (RAG)** y **Modelos de Lenguaje de Gran Tamaño (LLMs)**. Se deberá implementar un proceso que incluya el scraping de los manuales de usuario de Tesla, la construcción de un **sistema RAG** para recuperar información relevante, la interacción con una **API de un LLM** proporcionada por el equipo docente, y la creación de una **interfaz web** para la interacción con los usuarios.

Trabajo a realizar

En esta tarea, deberán desarrollar una **aplicación web que funcione como un chatbot, permitiendo a los usuarios hacer consultas sobre su Tesla** y recibir explicaciones detalladas. Para lograr esto, deberán:

- Permitir al usuario seleccionar su modelo de Tesla.
- Procesar y almacenar el contenido de los manuales, manejando posibles errores y asegurando la calidad de los datos.
- Generar una base de datos vectorial de fragmentos de artículos en Nomic Atlas.
- Implementar un sistema RAG para recuperar fragmentos relevantes de los artículos en respuesta a las consultas de los usuarios.
- Interactuar con una API de un LLM proporcionada por el equipo docente, enviando el contexto recuperado y recibiendo respuestas generadas.

Luego, cada uno deberá crear una interfaz web intuitiva que permita a los usuarios interactuar con el chatbot, seleccionar el modelo, realizar consultas y visualizar respuestas.



IIC3103 - Taller de Integración

Departamento Ciencia de la Computación
Escuela de Ingeniería
Pontificia Universidad Católica

Índice

Objetivo	1
Trabajo a realizar	1
Índice.....	2
Desarrollo de la tarea	2
Procesamiento de Datos.....	3
Creación de Dataset en Nomic Atlas	4
Creación de Servicio y Página Web de consultas	6
Búsqueda vectorial	7
Retrieval Augmented Generation.....	8
Interfaz de usuario	8
Interacción con la API del LLM.....	9
Configuración de la API.....	9
Detalles técnicos del modelo.....	10
Restricciones de la API	10
Integración de la API en el Backend	10
Manejo de Respuestas y Errores	10
Versionamiento del código	10
Entregables	11
Fecha de entrega	11
Requisitos mínimos.....	11
Penalizaciones.....	11
Versiones del documento	13
Anexos.....	14
3Blue1Brown: How large language models work, a visual intro to transformers	14
RAG + Langchain Python Project: Easy AI/Chat For Your Docs	16

Desarrollo de la tarea



IIC3103 - Taller de Integración

Departamento Ciencia de la Computación
Escuela de Ingeniería
Pontificia Universidad Católica

Procesamiento de Datos

Deberás descargar los archivos PDF de la página oficial de Tesla: <https://www.tesla.com/ownersmanual>

Para esta tarea, utilizaremos los manuales en español:

- [Model S](#)
- [Model X](#)
- [Model 3](#)
- [Model Y](#)
- [Cybertruck](#)

Deberán descargar los archivos PDF y procesarlos para generar un archivo en formato **.jsonl**.

Durante este proceso es importante **limpiar y preprocessar el texto**, eliminando:

- caracteres especiales,
- metadatos irrelevantes,
- referencias a imágenes u otros elementos no textuales,
- y cualquier contenido no deseado.

El objetivo es obtener un texto uniforme y consistente que pueda ser utilizado en los pasos posteriores.

Se recomienda **dividir el texto en fragmentos manejables**, por ejemplo en párrafos, utilizando herramientas como los **Text Splitters de LangChain**.

El tamaño sugerido para cada fragmento es de aproximadamente **800 caracteres**.

Cada línea del archivo .jsonl debe contener un objeto con el siguiente formato (recomendado):

```
{  
    "text": "Seguridad de los airbags. Si este indicador no parpadea momentáneamente cuando el Cybertruck se prepara para la conducción, o si permanece encendido, póngase en contacto con Tesla de inmediato. Consulta Indicador de estado del airbag en la página 51. Una puerta o el maletero delantero eléctrico están abiertos. Cybertruck detecta una conexión eléctrica incorrecta de las luces de remolque. Algunas luces, o todas, puede que no funcionen. Deténgase en cuanto la seguridad lo permita e inspeccione las luces del remolque en busca de averías en los cables o las conexiones. Si el problema se resuelve y se sigue mostrando el icono rojo, desactive el Modo Remolque y actívelo de nuevo. Consulta Uso de un remolque en la página 118. Advertencia de la presión de los neumáticos.",  
    "metadata": {  
        "document_title": "Tesla Cybertruck Owner's Manual",  
    }  
}
```



IIC3103 - Taller de Integración

Departamento Ciencia de la Computación
Escuela de Ingeniería
Pontificia Universidad Católica

```
"document_type": "tesla_manual",
"source_file": "Cybertruck.pdf",
"page_number": 17,
"char_count": 774,
"processed_at": "2025-10-04T22:56:45.745121"
}
}
```

Luego de haber generado los archivos jsonl para cada manual, y revisar que esté bien generado, es decir, que cada chunk esté bien definido, que no se corten palabras, etc.

Finalmente, deberás generar una archivo jsonl compilado de todos los manuales. Como referencia, este archivo debería tener alrededor de 10.000 líneas. Si tu archivo compilado tiene ordenes de magnitud de diferencia, probablemente hay algo mal y deberías revisar y procesar nuevamente los archivos.

⚠️ Todos los archivos jsonl, y los scripts para generarlos, deberán estar incluidos en el repositorio. Los PDFs de los manuales no es necesario que los suban (no lo hagan ya que son muy pesados y no aportan nada a la corrección)

Creación de Dataset en Nomic Atlas

Debes registrarte para una prueba gratuita en <https://atlas.nomic.ai/>.

La capa gratuita de Nomic te permitirá crear **un dataset público** y utilizar hasta **10 millones de tokens** para la generación de embeddings y búsquedas vectoriales. Este límite es más que suficiente para completar la tarea.

Una vez creada la cuenta, deberás crear un **dataset público** utilizando la opción **File Upload**, donde subirás el archivo previamente compilado.

Durante la configuración, selecciona el campo **text** como fuente de embeddings y crea el dataset.

Este proceso generará embeddings para cada línea de tu archivo .jsonl, produciendo un vector de N dimensiones que representa el contenido en forma semántica (es decir, según su significado).

La creación del dataset debería tardar entre **2 y 3 minutos**. Una vez finalizado, podrás visualizar el mapa correspondiente. La **URL de este mapa** será uno de los entregables de la tarea.

Ejemplo de URL:

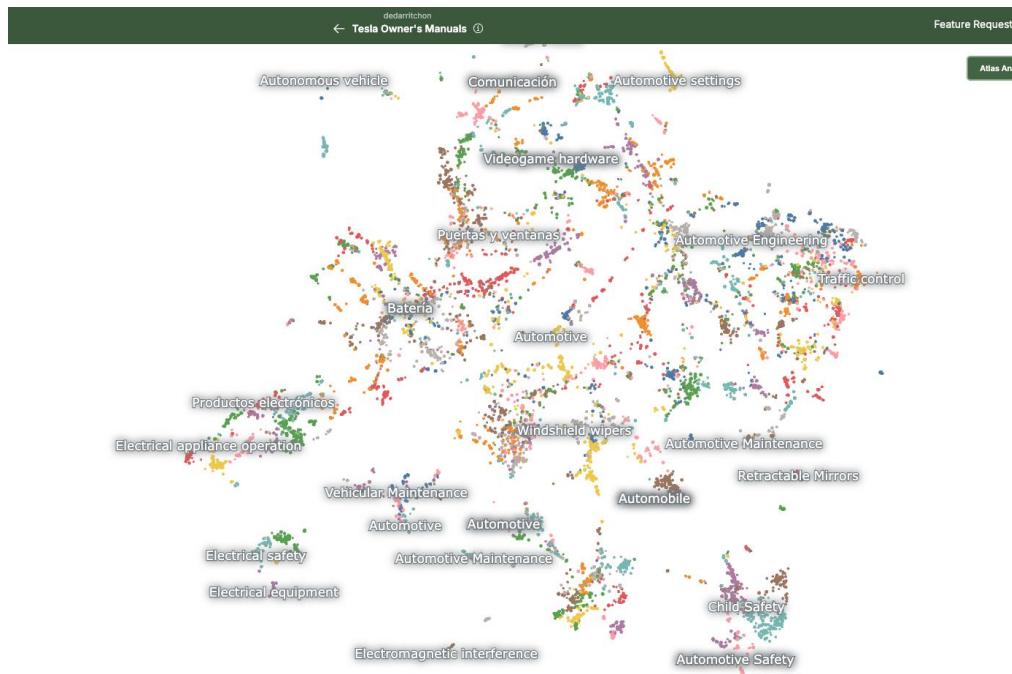
<https://atlas.nomic.ai/data/dedarritchon/tesla-owners-manuals/map>

Donde deberíamos ver algo similar a esto:



IIC3103 - Taller de Integración

Departamento Ciencia de la Computación
Escuela de Ingeniería
Pontificia Universidad Católica



Por otro lado, deberás tomar nota del projection id de tu dataset, valor que usaremos para hacer consultas a este dataset via API. Este identificador lo puedes encontrar en los settings del dataset:

Identifiers

Dataset ID
adb72830-f619-4310-a125-0ba698f01cae

Projection ID
f549d4dc-b7c7-411a-8468-42cf6fcc973

Por otra parte, necesitarás una API Key para usar la API de Nomic. Esta la podrás encontrar en los settings de tu cuenta:

<https://atlas.nomic.ai/data/<username>/org/settings>



IIC3103 - Taller de Integración

Departamento Ciencia de la Computación
Escuela de Ingeniería
Pontificia Universidad Católica

Settings

Plan & Billing

Usage

General

API Keys

Upgrade

API Keys

Manage your API keys for accessing Nomic services

ⓘ Your secret API keys are listed below. Please note that we do not display your secret API keys again after you generate them.

The API keys you create here will be scoped to your current organization dedarritchon. You can learn about organization & dataset permissions in the documentation for permissions and RBAC.

Do not share your API key with others, or expose it in the browser or other client-side code. In order to protect the security of your account, Nomic may disable keys that it detects have been exposed publicly.

Learn how to use your Nomic API key in the [API Reference](#).

NAME	VALUE	CREATION	EXPIRATION	
Prueba	nk-...U06WI	Oct 3, 2025	Oct 3, 2026	⋮
test	nk-...WN10E	Oct 4, 2025	Oct 4, 2026	⋮

[Create New API Key](#)

También en la sección **usage** puedes ver la cantidad de tokens que has usado hasta el momento:

Plan & Billing

Usage Overview

Usage

Monitor your platform utilization and get a detailed breakdown of your usage.

General

API Keys

Upgrade

Dataset Storage

the amount of storage used by your datasets including embedding storage

0 GBs used out of 100 GBs

Text Tokens

the amount of text you've processed (Dataset embedding creation or API usage)

1.8M tokens used out of 10M tokens

Images

the amount of images you've processed (Dataset embedding creation or API usage)

0 images used out of 1M images

Recuerda que solo tienes 10M de tokens en la capa gratuita, por lo que intenta no malgastar tokens y dejar suficientes tokens (idealmente 2M) para la corrección de la tarea (las consultas que haremos al corregir igual consumirán estos tokens).

Creación de Servicio y Página Web de consultas



IIC3103 - Taller de Integración

Departamento Ciencia de la Computación
Escuela de Ingeniería
Pontificia Universidad Católica

Búsqueda vectorial

Ahora deberás crear un servicio que te permita hacer consultas al dataset de Nomic Atlas.

Para esto, deberás usar la API de Nomic, específicamente, el método [Vector Search](#).

El body de la request debería ser tener el siguiente formato (esto depende del formato que hayan definido al crear el dataset)

```
{  
    "projection_id": <projection id de tu dataset>,  
    "k": 3,  
    "fields": ["text", "metadata"],  
    "query": "Cuanta presión deben llevar los neumáticos",  
    "selection": {  
        "method": "composition",  
        "conjunctor": "ALL",  
        "filters": [  
            {  
                "method": "search",  
                "query": "Cybertruck",  
                "field": "metadata"  
            }  
        ]  
    }  
}
```

En este caso, realizamos la consulta: “**¿Cuánta presión deben llevar los neumáticos?**”, aplicando un filtro por metadatos para obtener únicamente los contenidos relacionados con **Cybertruck**.

Esto resulta útil, ya que las consultas en la página estarán asociadas a un modelo en particular.

De esta forma, **Nomic devolverá los 3 vectores más similares semánticamente a la pregunta**.

El proceso funciona de la siguiente manera:

1. Nomic genera un vector a partir del texto de la consulta.
2. Luego compara este vector con los del dataset.
3. Finalmente, retorna aquellos que presentan mayor similitud semántica.



IIC3103 - Taller de Integración

Departamento Ciencia de la Computación
Escuela de Ingeniería
Pontificia Universidad Católica

Retrieval Augmented Generation

No queremos que nuestro sistema de soporte de Tesla responda mostrando directamente los fragmentos de los manuales, ya que esto no sería una experiencia amigable para el usuario. En su lugar, buscamos que el sistema responda **como si fuera una persona**.

Para lograrlo, utilizaremos **IA generativa** mediante una consulta “aumentada” con un LLM. Esto significa que construiremos un **prompt** que incluya tanto los fragmentos relevantes de los manuales (como contexto) como la **pregunta del usuario**.

De esta manera, el LLM podrá decidir cuál es la respuesta más adecuada y redactarla en un tono humano y amigable. Incluso podrá combinar información de varios fragmentos para dar una respuesta más completa.

Estructura sugerida del prompt para el LLM:

Eres un asistente de soporte de Tesla.
Tu tarea es responder preguntas de los usuarios de manera clara, precisa y amigable, como lo haría una persona experta
No muestres directamente los fragmentos de los manuales, pero utiliza su información como contexto para elaborar la respuesta.

Contexto (fragmentos relevantes de los manuales):

[FRAGMENTO 1]
[FRAGMENTO 2]
[FRAGMENTO 3]

Pregunta del usuario:

[PREGUNTA]

Interfaz de usuario

Deberás desarrollar una página web que implemente el siguiente flujo:

1. Selección del modelo Tesla:

2. El usuario debe poder elegir el modelo sobre el cual desea realizar la consulta.

3. Ingreso de la pregunta:

El usuario escribe su consulta en un campo de texto.

4. Ejecución de la consulta:



IIC3103 - Taller de Integración

Departamento Ciencia de la Computación
Escuela de Ingeniería
Pontificia Universidad Católica

Al presionar el botón “**Consultar**”, el sistema debe:

- a. Buscar en el dataset de manuales de Tesla los fragmentos más relevantes a la pregunta.
- b. Construir una **consulta aumentada con contexto** y enviarla a la **API de LLM del curso**.
- c. Mostrar la respuesta generada al usuario.

5. Referencias en la respuesta:

La respuesta debe incluir siempre referencias a los fragmentos utilizados, por ejemplo:

Tesla Cybertruck Owner Manual, página 3.

6. Manejo de errores:

Si no se encuentran fragmentos relevantes, el sistema debe mostrar un mensaje de error:

“No se encontraron fragmentos relacionados con la consulta.”

Si alguno de los pasos falla (obtener fragmentos o generación de respuesta con LLM) tambien deberá mostrarse claramente un mensaje de error en el chat.

7. Interfaz tipo chat:

El usuario debe poder realizar múltiples preguntas de manera continua. La interfaz debe presentarse como un chat.

Por simplicidad, **no es necesario mantener el contexto** entre preguntas: cada consulta se procesa de manera independiente. Al recargar la página, todo se reinicia desde 0 (no es necesario mantener un registro persistente de la conversación actual ni conversaciones anteriores)

Interacción con la API del LLM

Configuración de la API

El equipo docente gestionará el modelo LLM en un servidor separado, accesible mediante una API RESTful.

La API está alojada en el servidor <https://asteroide.ing.uc.cl>. Este servidor utiliza [Ollama](#) para servir un modelo LLM basado en [Llama3.2](#).

Para interactuar con la API, se pueden usar los siguientes métodos:

- [Generate a completion](#): genera una respuesta a un *prompt* con el modelo provisto.
(recomendado)
- [Generate a chat completion](#): genera el siguiente mensaje de un chat con el modelo provisto.



IIC3103 - Taller de Integración

Departamento Ciencia de la Computación
Escuela de Ingeniería
Pontificia Universidad Católica

Ollama también expone una [API basada en la API de OpenAI](#). Esta API es equivalente a los servicios anteriores, pero puede servir para compatibilidad de librerías u otros componentes.

Detalles técnicos del modelo

- Nombre modelo: integracion
 - NOTA: Este modelo se debe usar en todas las llamadas a la API. El servidor no tiene otros modelos implementados.
- Algoritmo base: [Llama3.2](#)
- Parámetros: 3.21B
- Configuraciones específicas ([documentación](#))
 - Temperatura (temperature): 6
 - Ventana de contexto (num_ctx): 512 tokens
 - Ventana de repetición (repeat_last_n) : 10
 - Límite de opciones de siguiente token (top_k): 18

Restricciones de la API

La API cuenta con un rate-limit de 10 request/seg. Se debe respetar este límite para asegurar un correcto funcionamiento para todo el curso.

Todas las llamadas a la API están sujetas a un límite de tiempo de 120 segundos. En este sentido, si el cálculo del resultado excede este tiempo, se deberá reducir el contexto o los prompt correspondientes para garantizar la capacidad de completar los cálculos necesarios en ese plazo de tiempo determinado.

Integración de la API en el Backend

Implementar llamadas a la API del LLM desde el backend de la aplicación para generar respuestas basadas en el contexto proporcionado por RAG.

Manejo de Respuestas y Errores

Procesar las respuestas del LLM y manejar posibles errores o tiempos de espera de la API.

Implementar mecanismos de reintento o mensajes de error informativos para el usuario.

Versionamiento del código

El sitio y todo su código fuente deberá estar versionado en un repositorio en GitHub, creado en el classroom del curso:

- Link para creación de repositorio: <https://classroom.github.com/a/Klo95gF1>

El repositorio deberá contener:

- El código fuente del frontend
- El código fuente del backend



IIC3103 - Taller de Integración

Departamento Ciencia de la Computación
Escuela de Ingeniería
Pontificia Universidad Católica

- **⚠️** Todos los archivos jsonl, y los scripts para generarlos, deberán estar incluidos en el repositorio. Los PDFs de los manuales no es necesario que los suban (no lo hagan ya que son muy pesados y no aportan nada a la corrección)

Entregables

Cada alumno deberá entregar, mediante un formulario publicado en el sitio del curso, las siguientes *url's*:

- URL sitio web.
- URL del repositorio a GitHub.
- Projection_id del dataset en Nomic.
- URL del dataset map público de Nomic.

La evaluación de la tarea se realizará bajo una rúbrica a publicar. Se evaluará la completitud del sitio generado y la correctitud de la integración solicitada.

Fecha de entrega

La tarea debe entregarse antes del 21 de octubre a las 18:00, y deberá estar desplegada y disponible durante el periodo de corrección (2 semanas). Además, deberán contar con suficientes tokens en Nomic para las consultas de la corrección.

Requisitos mínimos

Las tareas que no cumplan con las siguientes condiciones no serán corregidas y serán evaluados con la nota mínima:

- La página web deberá ser pública, accesible desde cualquier dispositivo conectado a internet.
- El código deberá estar versionado en su totalidad en un repositorio Git.
- El sitio implementado debe corresponder al código entregado. Para la revisión, más de una tarea serán corridas localmente por los ayudantes para comprobar el cumplimiento de este punto.
- En caso de que el código entregado no represente fielmente al sitio entregado, se calificará la tarea con la nota mínima.

Penalizaciones

Se descontarán 0,2 puntos de la nota de la tarea por cada hora de atraso en la entrega, contados a partir de la fecha estipulada en el punto anterior.

Cualquier intento de copia, plagio o acto deshonesto en el desarrollo de la tarea, será penalizado con nota 1,1 de acuerdo con la política de integridad académica del DCC.



IIC3103 - Taller de Integración

Departamento Ciencia de la Computación

Escuela de Ingeniería

Pontificia Universidad Católica



IIC3103 - Taller de Integración

Departamento Ciencia de la Computación
Escuela de Ingeniería
Pontificia Universidad Católica

Versiones del documento

- **Versión 1:**
 - 05/10/2025



IIC3103 - Taller de Integración

Departamento Ciencia de la Computación
Escuela de Ingeniería
Pontificia Universidad Católica

Anexos

Rúbrica Tentativa

Criterio	Descripción	Ponderación	No cumple	Bueno	Excelente
Preparación del dataset	Descarga, limpieza, fragmentación y carga de los manuales en Nomic.	20 %	No se genera el .jsonl o el dataset en Nomic.	Dataset creado pero con errores de limpieza, fragmentación o metadatos.	Dataset correctamente creado, limpio, fragmentado (~800 caracteres), con metadatos bien definidos.
Integración con Nomic	Uso correcto del dataset para recuperar fragmentos relevantes.	20 %	No logra conectarse ni consultar al dataset.	Recupera fragmentos, pero sin filtros o con resultados poco consistentes.	Recupera fragmentos precisos, aplica filtros por modelo y muestra coherencia con la pregunta.
Construcción del prompt aumentado	Creación del prompt con contexto y pregunta del usuario.	20 %	No genera prompt o lo hace sin incluir contexto.	Prompt incluye fragmentos pero con errores de estructura o claridad.	Prompt bien estructurado, con contexto y pregunta claramente integrados.
Interfaz web	Flujo de interacción con el usuario (selección de modelo, consulta, respuesta).	20 %	La interfaz no funciona o no permite realizar el flujo completo.	Flujo básico implementado, pero con fallas en usabilidad o diseño.	Interfaz clara, funcional y amigable, con flujo completo y aspecto de chat.
Referencias en la respuesta	Inclusión de referencias a los fragmentos usados (manual, página, etc.).	10 %	No incluye referencias.	Incluye referencias pero incompletas o poco claras.	Referencias completas y claras en cada respuesta.
Manejo de errores	Respuesta adecuada cuando no se encuentran fragmentos.	10 %	No maneja el caso de error.	Muestra error pero sin claridad.	Maneja el error correctamente con un mensaje claro y útil.

3Blue1Brown: How large language models work, a visual intro to transformers

Video que muestra cómo funciona un modelo de LLM. Explica cómo se entrena, como se generan los embeddings y por qué se usan relaciones vectoriales para esto, y otros conceptos como la temperatura del modelo.

Link: <https://www.youtube.com/watch?v=wjZofJX0v4M>



IIC3103 - Taller de Integración

Departamento Ciencia de la Computación
Escuela de Ingeniería
Pontificia Universidad Católica





IIC3103 - Taller de Integración

Departamento Ciencia de la Computación
Escuela de Ingeniería
Pontificia Universidad Católica

RAG + Langchain Python Project: Easy AI/Chat For Your Docs

Video que muestra una demo de un proyecto usando RAG para mejorar los resultados de consultas a un LLM, utilizando Python y Langchain.

En el video, usa como ejemplo la documentación de AWS, vectorizándola usando Embeddings y luego usando una BD vectorial para encontrar el mejor contexto relacionado con el prompt.

Link: <https://www.youtube.com/watch?v=tcqEUSNCn8I>

