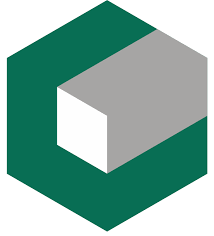
**Universidad Tecnológica de La Habana**

**“José Antonio Echeverría”**

**CUJAE**



**Construcción automática de grafos de conocimiento a partir de texto**

**Facultad de Ingeniería Informática**

Informe de Práctica Laboral 2

**Autora**: Sofia Alvarez Cisneros (salvarez@ceis.cujae.edu.cu)

**Tutor**: Dr. Alfredo Simón Cuevas (asimon@ceis.cujae.edu.cu)

**La Habana, Cuba**

**17 de enero 2025**

**Resumen**

Los grafos de conocimiento son estructuras de datos que organizan información mediante nodos y aristas, representando entidades y las relaciones entre ellas de forma semántica y navegable. Este proyecto aborda el problema de la ausencia de soluciones accesibles para la construcción automática de grafos de conocimiento a partir de texto en español, especialmente en contextos de recursos computacionales limitados.

La solución propuesta se centra en el desarrollo de un sistema que prioriza la funcionalidad en entornos de bajo rendimiento. El sistema utiliza spaCy para la extracción de entidades, un modelo basado en Transformers con prompts diseñados para identificar relaciones, y módulos para procesar, visualizar y exportar los grafos generados en formatos accesibles como CSV.

Este enfoque busca proporcionar una herramienta práctica para proyectos con infraestructura modesta, contribuyendo al avance en la representación y análisis semántico del conocimiento.

**Palabras claves:** Bajo rendimiento, extracción de entidades, grafos de conocimiento, LLM, relaciones semánticas, spaCy, texto en español

**Abstract**

Knowledge graphs are data structures that organize information through nodes and edges, representing entities and the relationships between them in a semantic and navigable manner. This project addresses the problem of the lack of accessible and efficient solutions for the automatic construction of knowledge graphs from text in Spanish, particularly in contexts with limited computational resources.

The proposed solution focuses on developing a system that prioritizes efficiency and functionality in low-performance environments. The system leverages spaCy for entity extraction, a Transformer-based model with custom-designed prompts to identify relationships, and modules to process, visualize, and export the generated graphs in accessible formats such as CSV.

This approach aims to provide a practical tool for projects with modest infrastructure, contributing to advancements in the semantic representation and analysis of knowledge.

**Keywords:** Low performance, entity extraction, knowledge graphs, LLM, semantic relationships, spaCy, Spanish text

**Índice**

[Introducción 1](#_Toc188012301)

[1. Fundamentos Teóricos 5](#_Toc188012302)

[1.1. Introducción a los grafos de conocimiento 5](#_Toc188012303)

[1.1.1. Conceptos Relacionados 7](#_Toc188012304)

[1.2. Métodos y tecnologías utilizadas 10](#_Toc188012305)

[1.3. Generación automática de grafos de conocimiento 12](#_Toc188012306)

[1.3.1. Extracción de entidades 12](#_Toc188012307)

[1.3.2. Extracción de relaciones 18](#_Toc188012308)

[1.4. Representación y visualización del grafo 22](#_Toc188012309)

[1.4.1. Heurística de optimización visual del grafo 24](#_Toc188012310)

[1.5. Comparación general con otros enfoques 26](#_Toc188012311)

[1.6. Conclusiones parciales 28](#_Toc188012312)

[2. Desarrollo de la solución propuesta 30](#_Toc188012313)

[2.1. Descripción general de la solución 30](#_Toc188012314)

[2.1.1. Requisitos del sistema 31](#_Toc188012315)

[2.1.2. Casos de uso del sistema 32](#_Toc188012316)

[2.2. Diseño de la solución 37](#_Toc188012317)

[2.2.1. Flujo del proceso 37](#_Toc188012318)

[2.2.2. Arquitectura del sistema 39](#_Toc188012319)

[2.2.3. Modelo de dominio 41](#_Toc188012320)

[2.3. Implementación de la extracción de información 43](#_Toc188012321)

[2.3.1. Extracción de entidades 43](#_Toc188012322)

[2.3.2. Identificación de relaciones 46](#_Toc188012323)

[2.4. Interfaz de usuario 48](#_Toc188012324)

[2.4.1. Interfaces Principales 49](#_Toc188012325)

[2.4.2. Interfaces Secundarias 49](#_Toc188012326)

[2.4.3. Retroalimentación al Usuario 50](#_Toc188012327)

[2.4.4. Manejo de Errores 51](#_Toc188012328)

[2.4.5. Implementación técnica 51](#_Toc188012329)

[2.5. Exportación del grafo 53](#_Toc188012330)

[2.6. Conclusiones parciales 54](#_Toc188012331)

[3. Validación de los resultados 56](#_Toc188012332)

[3.1. Marco de evaluación 56](#_Toc188012333)

[3.2. Datos para la evaluación 57](#_Toc188012334)

[3.2.1. Textos simples 57](#_Toc188012335)

[3.2.2. Textos intermedios 58](#_Toc188012336)

[3.2.3. Textos avanzados 59](#_Toc188012337)

[3.2.4. Textos extensos 61](#_Toc188012338)

[3.3. Métricas utilizadas 64](#_Toc188012339)

[3.4. Resultados obtenidos 65](#_Toc188012340)

[3.4.1. Evaluación de la extracción de entidades 65](#_Toc188012341)

[3.4.2. Evaluación de la identificación de relaciones 68](#_Toc188012342)

[3.4.3. Recursos computacionales utilizados 71](#_Toc188012343)

[3.5. Conclusiones parciales 72](#_Toc188012344)

[Conclusiones 74](#_Toc188012345)

[Recomendaciones 75](#_Toc188012346)

[Referencias bibliográficas 76](#_Toc188012347)

**Índice de imágenes**

[Ilustración 1: Ejemplo de grafos dirigidos y no dirigidos 5](#_Toc188012283)

[Ilustración 2:Ejemplo de grafo conexo y grafo completo 5](#_Toc188012284)

[Ilustración 3: Ejemplo de grafo de conocimiento 6](#_Toc188012285)

[Ilustración 4: Visualización simple de un grafo de conocimiento 23](#_Toc188012286)

[Ilustración 5: Grafo antes de la ejecución del método heurístico 26](#_Toc188012287)

[Ilustración 6: : Grafo después de la ejecución del método heurístico 26](#_Toc188012288)

[Ilustración 7: Diagrama de casos de uso del sistema 33](#_Toc188012289)

[Ilustración 8: Flujo del sistema mediante el patrón de filtros y tuberías 38](#_Toc188012290)

[Ilustración 9: Patrón N-capas 39](#_Toc188012291)

[Ilustración 10: Modelo de dominio 42](#_Toc188012292)

[Ilustración 11: Resultado de la extracción de entidades en textos sencillos 66](#_Toc188012293)

[Ilustración 12: Resultado de la extracción de entidades en textos intermedios 66](#_Toc188012294)

[Ilustración 13: Resultado de la extracción de entidades en textos avanzados 67](#_Toc188012295)

[Ilustración 14: Resultado de la extracción de entidades en textos extensos 67](#_Toc188012296)

[Ilustración 15: Resultado de la identificación de relaciones en textos sencillos 69](#_Toc188012297)

[Ilustración 16: Resultado de la identificación de relaciones en textos intermedios 69](#_Toc188012298)

[Ilustración 17: Resultado de la identificación de relaciones en textos avanzados 70](#_Toc188012299)

[Ilustración 18: Resultado de la identificación de relaciones en textos extensos 70](#_Toc188012300)

**Índice de tablas**

[Tabla 1: Comparativa de grafos de conocimiento con otras estructuras de datos 7](#_Toc188012270)

[Tabla 2: Comparativa de métodos para extracción de entidades 17](#_Toc188012271)

[Tabla 3 Comparativa de métodos para la extracción de relaciones semánticas 21](#_Toc188012272)

[Tabla 4: Comparativa general con otros enfoques 28](#_Toc188012273)

[Tabla 5: Ejemplo de extracción de entidades para cada categoría de texto 66](#_Toc188012274)

[Tabla 5: Ejemplo de identificación de relaciones para cada categoría de texto 69](#_Toc188012275)

[Tabla 6: Recursos computacionales de cada ejecución promedio por categoría de texto 71](#_Toc188012276)

# Introducción

Los grafos de conocimiento son estructuras de datos que representan información mediante nodos y aristas, donde los nodos representan entidades, como personas, lugares, conceptos u objetos, y las aristas describen las relaciones entre ellas (1). Estas estructuras permiten organizar el conocimiento de forma semántica y navegable, ofreciendo un modelo que facilita la interpretación y el análisis de datos en contextos diversos.

La construcción de grafos de conocimiento ha emergido como una solución poderosa para organizar, modelar y visualizar información de manera estructurada, permitiendo conectar entidades y relaciones semánticas en dominios diversos. Estos grafos son utilizados en una variedad de aplicaciones, desde la recuperación de información hasta el razonamiento automático y la inteligencia artificial explicable (1).

Sin embargo, la generación automática de grafos de conocimiento a partir de texto enfrenta múltiples desafíos, especialmente en contextos de bajo rendimiento o recursos computacionales limitados. Aunque los modelos basados en Transformers, como Llama, BERT, etc, han demostrado ser herramientas eficaces para la extracción de entidades y relaciones, su implementación requiere una infraestructura computacional considerable, lo cual dificulta su uso en escenarios más modestos (2, 3). En este contexto, surge la necesidad de sistemas que prioricen el equilibrio entre la funcionalidad y el uso limitado de recursos, especialmente al procesar textos complejos y en español (4, 5).

**Situación Problemática:** Los sistemas actuales para construir grafos de conocimiento no se adaptan bien a entornos de bajos recursos, lo que limita su uso en aplicaciones educativas y proyectos con infraestructura modesta. Este problema se agrava en el procesamiento de texto en español, dado su mayor complejidad lingüística.

El **problema** central radica en la ausencia de soluciones accesibles para construir grafos de conocimiento a partir de texto en sistemas de bajos recursos.

Este proyecto tiene como **objetivo general** desarrollar un sistema para la construcción automática de grafos de conocimiento que priorice la utilización de herramientas y modelos ligeros, con el propósito de ofrecer una solución funcional en entornos de bajo rendimiento.

La solución se compone de cuatro etapas fundamentales: la extracción de entidades utilizando spaCy, la identificación de relaciones a través de un modelo basado en Transformers, mediante prompts diseñados, el procesamiento de datos para estructurar el grafo, y la visualización y exportación del grafo generado en formatos accesibles como CSV.

**Objetivos Específicos:**

1. **Desarrollar un módulo ligero para la extracción de entidades en textos en español**
   1. Desarrollar una arquitectura que permita utilizar herramientas ligeras como spaCy para la tarea.
   2. Definir los tipos de entidades que serán extraídas.
   3. Diseñar patrones y reglas personalizadas en caso de que las entidades específicas no sean reconocidas por defecto.
   4. Identificar frases idiomáticas que no aportan contenido relevante, para ser excluidas.
2. **Desarrollar un módulo ligero para la identificación de relaciones semánticas en textos en español**
   1. Desarrollar una arquitectura que permita utilizar un modelo LLM a partir de prompts.
   2. Definir un prompt que optimice el desempeño del modelo en tareas de identificación de relaciones.
   3. Extraer las relaciones identificadas en la respuesta del modelo para generar una estructura manejable
3. **Generar visualizaciones del grafo**
   1. Implementar una herramienta para generar representaciones gráficas claras y coherentes del grafo.
   2. Integrar técnicas para mejorar la disposición de nodos y aristas en la visualización.
4. **Exportar los grafos generados a formatos accesibles** 
   1. Definir un formato estándar para los archivos CSV.
   2. Implementar un módulo que convierta los datos estructurados del grafo en archivos CSV.
5. **Validar el sistema en un escenario con recursos computacionales limitados.**
   1. Seleccionar el conjunto de datos sobre el cual se evaluará la solución propuesta.
   2. Registrar tiempo de procesamiento por texto, consumo de memoria y uso de CPU y de RAM.
   3. Evaluar métricas clave como Precision, Recall, F1-Score para la extracción de las entidades y relaciones.

**Aporte**

El sistema desarrollado proporciona una solución práctica para la construcción de grafos de conocimiento en entornos con recursos limitados. Este enfoque facilita su adaptación en contextos educativos, pequeños proyectos empresariales o investigaciones que no cuentan con acceso a infraestructura avanzada

**Estrategia de Validación**

La evaluación del sistema se llevará a cabo mediante pruebas manuales para medir la calidad de los grafos generados, considerando aspectos como la precisión y correcta identificación de entidades y relaciones, y uso de los recursos computacionales. Se utilizarán textos con estructuras variadas para comprobar su versatilidad

**Aplicación y Aplicabilidad**

El sistema está diseñado para su aplicación en entornos donde la extracción de conocimiento semántico de texto sea crucial, o sea necesario llevar texto en lenguaje natural a un formato estructurado. Su diseño lo hace particularmente útil en escenarios con recursos computacionales modestos.

Este proyecto, por tanto, no busca optimizar los resultados sino garantizar la funcionalidad en un marco de limitados recursos, permitiendo que los grafos de conocimiento sean más accesibles y prácticos en aplicaciones cotidianas y de bajo costo. Con ello, se espera contribuir al avance en la democratización de estas tecnologías, allanando el camino para su integración en contextos diversos.

Para el desarrollo de este proyecto, se contó con los recursos de una laptop hp de gama baja, con un procesador Intel(R) Core (TM) i3-5005U CPU @ 2.00GHz 2.00 GHz, y 8gb de ram, con tarjeta de video integrada, un medio cómputo con propiedades entre lo estándar y bajo, que entra dentro de lo que es considerado para este proyecto como un entorno de bajo rendimiento.

# Fundamentos Teóricos

## Introducción a los grafos de conocimiento

Un grafo es una estructura matemática que consiste en un conjunto de nodos (también llamados vértices) y aristas (o bordes), las cuales representan las conexiones entre los nodos. Formalmente, un grafo se define como ***G=(V,E)*** donde ***V*** es el conjunto de nodos y ***E⊆ V×V*** es el conjunto de aristas que conectan pares de nodos. Las aristas pueden ser dirigidas, cuando tienen una orientación específica (por ejemplo, A→B), o no dirigidas, cuando la relación es bidireccional (A−B)​. (6, 7)

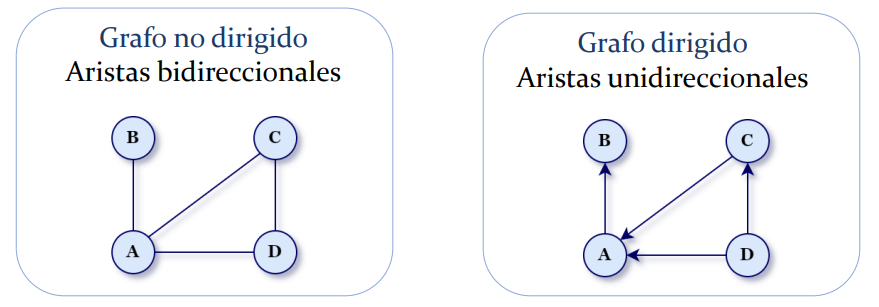
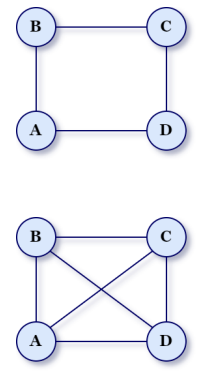


Ilustración : Ejemplo de grafos dirigidos y no dirigidos



**Grafo conexo**

Los grafos pueden clasificarse según varias propiedades:

1. **Grafos dirigidos**: Cada arista tiene una dirección definida.
2. **Grafos no dirigidos**: Las aristas no tienen una orientación específica.

**Grafo completo**

1. **Grafos ponderados**: Las aristas tienen un valor asociado, como peso o costo ​
2. **Grafo conexo:** cada par de vértices está conectado por un camino.

Ilustración :Ejemplo de grafo conexo y grafo completo

1. **Grafo completo:** existen aristas uniendo todos los pares posibles de vértices.

**Aplicaciones Generales**

Los grafos son ampliamente utilizados en diversos campos, como:

* **Redes sociales**: Para modelar relaciones entre individuos.
* **Sistemas de transporte**: Representación de rutas y conexiones.
* **Sistemas de recomendación**: Modelado de relaciones entre usuarios y productos ​(7, 8)

Un grafo de conocimiento se compone de nodos y aristas, donde los nodos representan entidades (por ejemplo, personas, objetos, conceptos o lugares) y las aristas describen las relaciones entre ellas. Estas relaciones pueden incluir conexiones directas como "es amigo de" o "trabaja en", y conexiones complejas como causales o jerárquicas.(9) En el caso del proyecto, los grafos de conocimiento fueron desarrollados como grafos dirigidos y completos, con aristas etiquetadas, para una mejor representación del contenido de los textos.

Básicamente, son redes semánticas que estructuran información mediante tripletas (entidad, relación, entidad), donde cada tripleta captura un hecho específico. Por ejemplo, la tripleta ("Einstein", "descubrió", "Teoría de la Relatividad") representa un hecho en un grafo (1)​(3)

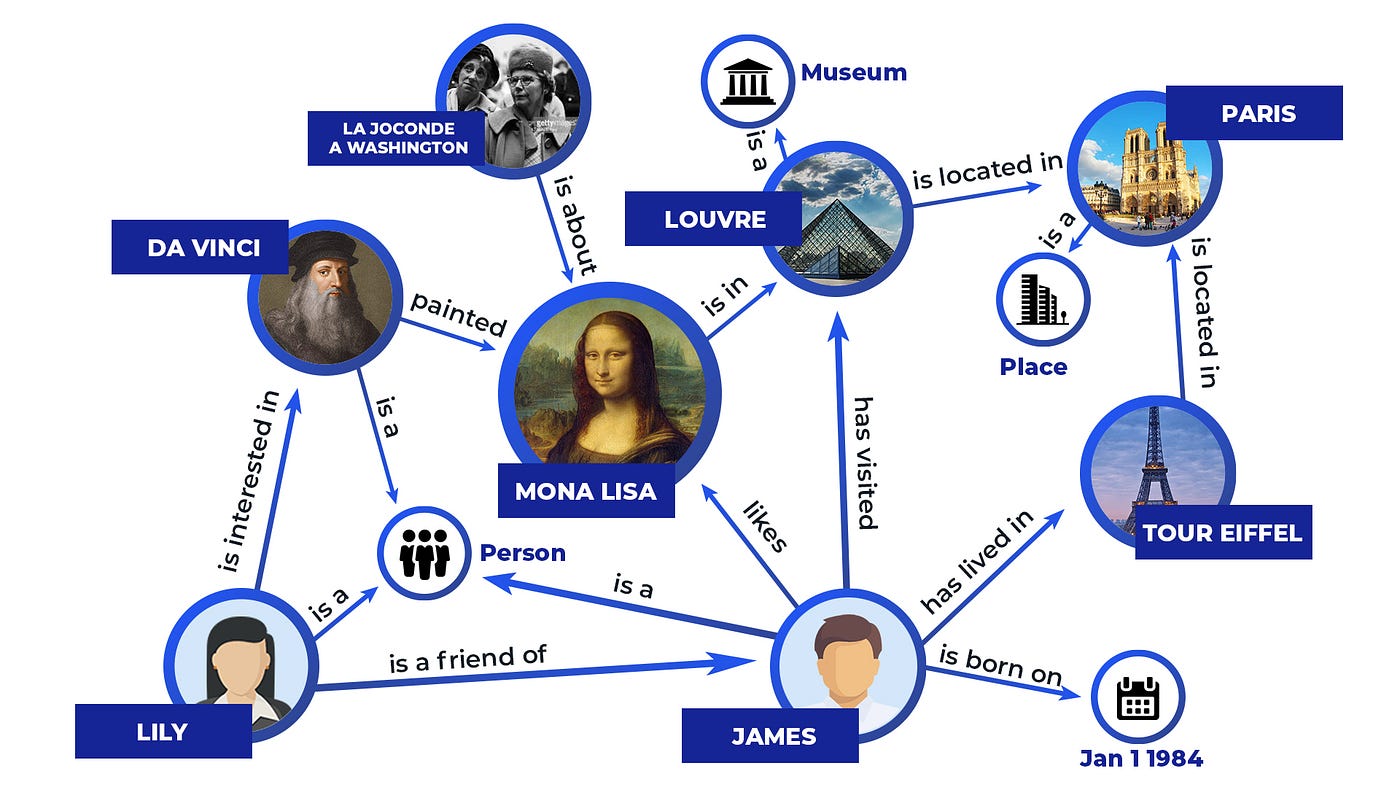


Ilustración : Ejemplo de grafo de conocimiento

Los grafos de conocimiento han evolucionado como una herramienta crucial en el campo del procesamiento de información, especialmente en el contexto de la inteligencia artificial y el aprendizaje automático. Estas estructuras de datos permiten organizar el conocimiento de forma semántica, representando entidades y sus relaciones en un modelo gráfico​(1). Se caracterizan por:

1. **Estructura Semántica**: Representan información de manera jerárquica y lógica, lo que facilita su comprensión y análisis por humanos y máquinas. Esto hace que puedan ser útiles para el entrenamiento de modelos y demás tareas relacionadas con la inteligencia artificial. Su capacidad para incorporar tanto datos estructurados como no estructurados es una ventaja clave.
2. **Conectividad**: Permiten enlazar entidades y relaciones complejas, creando un modelo global interconectado que respalda el razonamiento automático (4)​(10)
3. **Escalabilidad**: Se pueden expandir continuamente con nuevos datos, lo que los hace ideales para sistemas dinámicos como motores de búsqueda y asistentes virtuales​ (1)
4. **Adaptabilidad Multidominio**: Los grafos de conocimiento pueden ser específicos para dominios como la medicina, el derecho o la educación, o generales como en Wikidata o DBpedia ​

La siguiente tabla compara los grafos de conocimiento con otras estructuras de datos, como bases de datos relacionales y sistemas de etiquetas, destacando sus ventajas y limitaciones​

|  |  |  |  |
| --- | --- | --- | --- |
| Característica | Grafos de Conocimiento | Bases de Datos Relacionales | Etiquetado Semántico |
| Estructura | Tripletas (nodo-arista-nodo) | Tablas | Palabras clave |
| Escalabilidad | Alta | Limitada | Limitada |
| Conectividad | Semántica y jerárquica | Relacional | Superficial |

Tabla : Comparativa de grafos de conocimiento con otras estructuras de datos

### Conceptos Relacionados

Para comprender completamente el funcionamiento de los grafos de conocimiento y su aplicación en proyectos como el tuyo, es esencial entender estos conceptos básicos en profundidad:

**1. Nodo**

* **Definición**: Un nodo representa una entidad o concepto dentro de un grafo de conocimiento. Puede ser un elemento tangible (como "Albert Einstein" o "París") o intangible (como "Teoría de la Relatividad" o "Justicia"). Los nodos son los puntos de anclaje que contienen la información principal sobre los objetos que queremos modelar​ (11, 12) . Cada nodo puede conectarse con más de un nodo dentro de un grafo de conocimiento.
* **Importancia**:
  + Facilitan la identificación y la organización de las entidades.
  + Constituyen los "bloques fundamentales" del grafo.

**2. Arista**

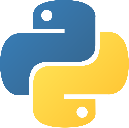
* **Definición**: Una arista representa la conexión o relación entre dos nodos. Estas conexiones pueden ser de diversos tipos, dependiendo del contexto y del dominio del grafo. Por ejemplo, relaciones jerárquicas ("es un tipo de"), causales ("provoca"), temporales ("ocurrió antes que") o semánticas ("es amigo de")​ (11, 13).
* **Características**:
  + Las aristas pueden ser dirigidas o no dirigidas (en el caso de este proyecto, se seleccionaron aristas dirigidas para una mayor cobertura de todos los casos):
    - *Dirigidas*: La relación tiene un sentido específico (por ejemplo, "París es la capital de Francia").
    - *No dirigidas*: La relación no tiene un sentido particular (por ejemplo, "A y B son amigos").
* **Ejemplo**:
  + Relación jerárquica: ("Mamífero", "es un tipo de", "Animal").
  + Relación causal: ("Albert Einstein", "descubrió", "Teoría de la Relatividad") ​
* **Importancia**:
  + Proveen el significado semántico que conecta los nodos.
  + Permiten realizar inferencias y análisis de las relaciones entre entidades.

**3. Tripleta**

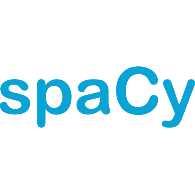
* **Definición**: Una tripleta es la unidad básica de los grafos de conocimiento, formada por dos nodos y la arista que los conecta. Se representa comúnmente como *(sujeto, relación, objeto)*. Por ejemplo: ("París", "es la capital de", "Francia") ​(14)
* **Características**:
  + Captura un hecho o unidad de conocimiento en el grafo.
  + Las tripletas son fácilmente transformables en formatos compatibles con bases de datos y consultas.
* **Ejemplo**:
  + En un grafo sobre historia: ("Napoleón", "nació en", "1769").
  + En un grafo sobre cine: ("El Padrino", "dirigido por", "Francis Ford Coppola").
* **Importancia**:
  + Facilitan la representación estructurada de hechos, asegurando que la información sea accesible y reutilizable.
  + Son esenciales para la construcción y consulta de grafos de conocimiento, sirviendo como "ladrillos" básicos para el modelado de información (7, 14)

## Métodos y tecnologías utilizadas

El desarrollo de este proyecto implicó una integración meticulosa de herramientas y tecnologías modernas. A continuación, se describen los métodos, bibliotecas y modelos clave utilizados en el sistema:

**Lenguaje de Programación:** El proyecto fue implementado utilizando Python, un lenguaje ampliamente reconocido por su versatilidad y su extensa biblioteca de paquetes, que lo hacen ideal para aplicaciones de procesamiento de lenguaje natural (PLN) e inteligencia artificial (IA)​

**Modelos LLaMa:** Modelos Transformer preentrenados desarrollado por Meta, ideales para tareas relacionadas con el procesamiento del lenguaje natural debido a su capacidad para comprender y generar texto coherente basado en contextos complejos​ (12, 15). En este proyecto fue utilizado para la extracción de relaciones entre entidades identificadas en el texto el modelo LLaMa2:7b, el sufijo 7b indica que el modelo tiene aproximadamente 7 mil millones de parámetros, este modelo tiene una gran capacidad para entender y generar texto, pero es más compacto y eficiente que versiones más grandes como las de 13 mil millones (13b) o 70 mil millones (70b).

* **Características principales**:
  + Basado en la arquitectura Transformer.
  + Preentrenado en grandes conjuntos de datos multilingües.
  + ****Ajuste fino mediante prompts diseñados para extraer relaciones semánticas entre nodos del grafo.

**spaCy:** La biblioteca spaCy y el modelo de idioma **es\_core\_news\_sm** fueron utilizados para la extracción de entidades nombradas (NER) en textos en español. SpaCy es una biblioteca avanzada de procesamiento de lenguaje natural (NLP) en Python diseñada para ser rápida, eficiente y fácil de usar. Ofrece herramientas para tareas como tokenización, lematización, análisis sintáctico, etiquetado de partes de la oración, reconocimiento de entidades nombradas y más. Es ampliamente utilizada en aplicaciones de extracción de información, análisis de texto y generación de datos estructurados gracias a su capacidad para manejar grandes volúmenes de texto con modelos preentrenados optimizados para múltiples idiomas, incluido el español.

**NetworkX:** Biblioteca de Python diseñada para la creación, manipulación y análisis de grafos complejos.

**Matplotlib** es una biblioteca de Python utilizada para la creación de gráficos y visualizaciones. Permite generar visualizaciones 2D personalizables. Se utilizo para la representación gráfica del grafo, con nodos y aristas estilizados, ajuste dinámico del tamaño de los nodos y etiquetas para maximizar la claridad visual​

**Metodo heurístico:** Es un enfoque basado en reglas o estrategias prácticas que se utiliza para resolver problemas complejos de manera eficiente, aunque no garantiza encontrar la solución óptima. Estos métodos son especialmente útiles en problemas donde la búsqueda de la solución exacta sería computacionalmente costosa o inviable. Fue utilizado para mejorar la disposición de los nodos y las aristas, optimizando la claridad de las visualizaciones generadas​. Se basa en la implementación de algoritmos para reducir el cruce de aristas y mejorar la distribución espacial de los nodos.

**Tkinter**: Módulo estándar de Python para la creación de interfaces gráficas de usuario (1). Proporciona herramientas para desarrollar aplicaciones visuales con elementos interactivos como botones, cuadros de texto, etiquetas, menús y más. Es conocido por su simplicidad y su integración nativa en Python, lo que lo hace ideal para proyectos que requieren una interfaz de usuario básica pero funcional.

**LangChain-Ollama:** Integración específica de LangChain diseñada para trabajar con modelos de lenguaje preentrenados (LLMs) gestionados localmente por Ollama. Su objetivo principal es facilitar el desarrollo de aplicaciones que requieren procesamiento de lenguaje natural, permitiendo conectar el modelo Llama2:7b alojado en un servidor local con flujos de datos, herramientas externas y pipelines personalizados. Fue empleado como un componente clave para manejar las interacciones con el servidor Ollama, estructurando las solicitudes al modelo y procesando sus respuestas de manera eficiente y coherente, garantizando una ejecución completamente local**.**

Se seleccionaron spaCy y Llama2:7b por su balance entre eficiencia y rendimiento en entornos de recursos limitados. spaCy ofrece una extracción precisa de entidades en español, mientras que Llama2:7b, modelo multilingüe, permite identificar relaciones semánticas complejas mediante prompts específicos, sin necesidad de entrenamiento adicional.

## Generación automática de grafos de conocimiento

La **generación automática de grafos de conocimiento** es un proceso que transforma datos textuales no estructurados en representaciones gráficas estructuradas, capturando entidades y las relaciones entre ellas. Este enfoque es esencial en áreas como la minería de textos, el procesamiento del lenguaje natural (PLN) y la inteligencia artificial (IA), ya que permite extraer y organizar información valiosa de grandes volúmenes de texto.

Según (7), los grafos proporcionan una forma eficaz de representar la estructura y las relaciones intrínsecas en los datos textuales, superando las limitaciones de modelos tradicionales como el "bag of words" (BOW) . Los grafos capturan no solo las entidades presentes en el texto, sino también las conexiones semánticas y sintácticas entre ellas, lo que enriquece el análisis y comprensión de la información.

En el contexto de este proyecto, se desarrolló un sistema que automatiza la generación de grafos de conocimiento a partir de texto en español. Este sistema integra técnicas de extracción de entidades, identificación de relaciones y visualización de grafos, utilizando herramientas y bibliotecas avanzadas en Python. El objetivo es facilitar la representación y comprensión de la información contenida en textos, permitiendo a los usuarios visualizar y analizar las conexiones entre diferentes conceptos y entidades.

A continuación, se hace un estudio de las etapas fundamentales de este sistema: la extracción de entidades y la extracción de relaciones.

### Extracción de entidades

La **extracción de entidades nombradas** es el proceso de identificar y clasificar elementos clave en el texto, como personas, organizaciones, lugares y conceptos. Estas entidades sirven como nodos en el grafo de conocimiento, representando los elementos fundamentales sobre los cuales se construyen las relaciones.

La extracción de entidades y conceptos implica identificar automáticamente información estructurada, como entidades, relaciones y atributos, desde fuentes de texto no estructurado. Es un campo crítico en el procesamiento de lenguaje natural (PLN) que permite estructurar conocimiento desde datos dispersos, como artículos académicos, blogs o bases de datos de la web (16, 17).

**Metodologías Principales**

1. **Enfoques Basados en Reglas y Patrones**

Los enfoques basados en reglas en inteligencia artificial se fundamentan en la utilización de reglas predefinidas, generalmente extraídas del conocimiento experto, para resolver problemas o tomar decisiones. Estas reglas siguen una lógica condicional, por ejemplo, "Si sucede A, entonces hacer X". Estos enfoques pueden ser limitados en escenarios complejos, ya que requieren actualizar manualmente las reglas y no son capaces de aprender de manera autónoma. A pesar de estos retos, siguen siendo útiles por su transparencia y capacidad de generar decisiones explicables.

Los métodos tradicionales para la tarea se han centrado en patrones lingüísticos (como los patrones de Hearst) para identificar relaciones de hiponimia y conjuntos de términos coordinados. Estos métodos son útiles, pero limitados en escalabilidad y precisión en textos complejos ​(17, 18)

* **Ventajas**: Simples y eficientes para tareas limitadas.
* **Limitaciones**: Baja generalización en dominios abiertos.
* **Requerimientos Computacionales**:
  + Procesador: CPU básica.
  + Memoria: ~4 GB RAM suficiente para textos pequeños.
  + Software: Librerías como NLTK o SpaCy.

1. **Modelos Basados en Ontologías**

Las ontologías en inteligencia artificial representan de manera formal un conjunto de conceptos y las relaciones entre ellos en un dominio específico. Los modelos basados en ontologías permiten estructurar y organizar el conocimiento de manera que las máquinas puedan procesarlo y razonarlo, facilitando la interpretación semántica de los datos. Estos modelos son especialmente valiosos en aplicaciones de razonamiento automatizado y en la interoperabilidad de sistemas. Por ejemplo, en el campo de la medicina, las ontologías pueden ayudar a integrar datos de distintas fuentes para diagnóstico y tratamiento.

Sistemas como los que usan la ontología YAGO han destacado en la extracción de entidades semánticas mediante desambiguación de entidades ambiguas, aprovechando el conocimiento estructurado en forma de tripletas RDF (entidad-relación-entidad) ​(16)

* **Ventajas**: Alta precisión y semántica enriquecida.
* **Limitaciones**: Alto costo computacional en preprocesamiento.
* **Requerimientos Computacionales**:
  + Procesador: Múltiples núcleos (>4).
  + Memoria: >8 GB RAM.
  + Almacenamiento: ~15 GB para bases de datos grandes como YAGO.
  + Procesamiento de corpus grandes (~22 días en CPU dual-core con 2 GB RAM​ según (16) ).

1. **Enfoques de Aprendizaje Automático**

El aprendizaje automático (Machine Learning) es un subcampo de la inteligencia artificial que permite a los sistemas aprender patrones a partir de datos, sin ser explícitamente programados. Se basa en algoritmos que ajustan modelos estadísticos para predecir o clasificar datos. El enfoque más utilizado es el aprendizaje supervisado, donde un modelo se entrena con un conjunto de datos etiquetados. A medida que el modelo procesa más datos, mejora su capacidad para hacer predicciones o clasificaciones. El aprendizaje profundo (Deep Learning), una variante avanzada, utiliza redes neuronales profundas para aprender representaciones más complejas y abstractas de los datos.

Métodos supervisados como Random Forest y redes neuronales profundas (Deep Neural Networks) han mostrado buenos resultados en tareas como la clasificación y predicción de atributos relacionados con entidades, particularmente en aplicaciones industriales y científicas​(19).

Este enfoque fue desarrollado en la extracción de entidades en textos financieros o académicos​ en(20).

* **Ventajas**: Alta precisión y adaptabilidad.
* **Limitaciones**: Dependencia de datos etiquetados.
* **Requerimientos Computacionales**:
  + Procesador: GPU moderna
  + Memoria: 64 GB RAM para grandes conjuntos de datos.
  + Software: TensorFlow o PyTorch.
  + Entrenamiento: Horas a días para datasets moderados (~1M muestras).
  + Inferencia: Milisegundos a segundos por muestra.

1. **Enfoques No Supervisados**

El aprendizaje no supervisado se refiere a técnicas donde el algoritmo encuentra patrones en datos sin la necesidad de etiquetas predefinidas. El objetivo principal es descubrir estructuras subyacentes en los datos, como agrupaciones o relaciones. Clustering y reducción de dimensiones son dos técnicas comunes dentro de este enfoque. Un ejemplo famoso es el algoritmo K-means, que agrupa los datos en clusters basados en su similitud. Este enfoque es útil cuando no se dispone de datos etiquetados o cuando se buscan patrones emergentes en grandes volúmenes de datos no estructurados.

Técnicas como WebSets han explorado la agrupación de términos desde tablas HTML y la asignación de conceptos mediante datos masivos no etiquetados. Este método combina datos estructurados y semiestructurados para identificar relaciones implícitas entre entidades​(18) .

* **Ventajas**: Procesa datos no etiquetados.
* **Limitaciones**: Menor precisión comparado con métodos supervisados.
* **Requerimientos Computacionales**:
  + Procesador: CPU multinúcleo.
  + Memoria: ~16 GB RAM.
  + Software: Herramientas distribuidas como Apache Spark.
  + Procesamiento en lotes: Minutos a horas dependiendo del corpus.

1. **Extracción End-to-End**

La extracción end-to-end es un enfoque en el cual un modelo de inteligencia artificial aprende a realizar una tarea completa, desde la entrada hasta la salida, de manera directa y sin necesidad de pasos intermedios manuales. En lugar de depender de varios módulos o etapas, este enfoque optimiza un único modelo que toma los datos crudos y produce el resultado deseado. Por ejemplo, en tareas de procesamiento de lenguaje natural (PLN), un modelo end-to-end puede tomar un texto sin procesar y generar una traducción o resumen sin intervención humana en las etapas intermedias. Este enfoque ha ganado popularidad con el auge de redes neuronales profundas y sistemas como BERT o GPT, que muestran una capacidad notable para realizar tareas complejas de manera autónoma.

La arquitectura de redes neuronales recurrentes (RNN) integradas con modelos de reconocimiento de habla automática (ASR) permite la extracción directa de entidades y conceptos desde el habla, eliminando pasos intermedios y reduciendo la propagación de errores(19). ​

* **Ventajas**: Eliminación de errores intermedios, todo en un paso.
* **Limitaciones**: Requiere grandes volúmenes de datos anotados.
* **Requerimientos Computacionales**:
  + Procesador: GPU de alto rendimiento
  + Memoria: 64-128 GB RAM.
  + Almacenamiento: SSD para datos grandes (~1 TB).
  + Entrenamiento: Semanas para modelos con ~160 horas de audio​(19)
  + Inferencia: Milisegundos a segundos por audio corto.

A continuación, se presenta una tabla comparativa con los enfoques analizados:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Enfoque | Ventajas | Limitaciones | Requerimientos Computacionales | Tiempos de ejecución |
| Basados en Reglas y Patrones | Simples y eficientes para tareas limitadas | Baja generalización en dominios abiertos | - CPU básica- Memoria: ~4 GB RAM- Software: NLTK, SpaCy | Ejecución rápida en corpus pequeños (segundos a minutos). |
| Modelos Basados en Ontologías | Alta precisión y semántica enriquecida | Alto costo computacional en preprocesamiento | - Procesador: Múltiples núcleos (>4)- Memoria: >8 GB RAM- Almacenamiento: ~15 GB para bases de datos grandes como YAGO | Procesamiento de corpus grandes (~22 días en CPU dual-core con 2 GB RAM). |
| Aprendizaje Automático | Alta precisión y adaptabilidad | Dependencia de datos etiquetados | - Procesador: GPU moderna (NVIDIA Tesla V100 o superior)- Memoria: 64 GB RAM- Software: TensorFlow o PyTorch | Entrenamiento: Horas a días para datasets moderados (~1M muestras).- Inferencia: Milisegundos a segundos por muestra. |
| No Supervisados | Procesa datos no etiquetados | Menor precisión comparado con métodos supervisados | - Procesador: CPU multinúcleo- Memoria: ~16 GB RAM- Software: Apache Spark | Procesamiento en lotes: Minutos a horas dependiendo del corpus. |
| Extracción End-to-End | Eliminación de errores intermedios, todo en un paso | Requiere grandes volúmenes de datos anotados | - Procesador: GPU de alto rendimiento (e.g., NVIDIA A100)- Memoria: 64-128 GB RAM- Almacenamiento: SSD para datos grandes (~1 TB) | Entrenamiento: Semanas para modelos con ~160 horas de audio.- Inferencia: Milisegundos a segundos por audio corto. |

Tabla : Comparativa de métodos para extracción de entidades

Para llevar a cabo la extracción de entidades en este proyecto, se utilizó la biblioteca **SpaCy** junto con su modelo en español **es\_core\_news\_sm**. Este enfoque combina las capacidades de un modelo preentrenado basado en aprendizaje supervisado con una lógica personalizada para maximizar la precisión en la identificación de entidades. El modelo realiza un análisis morfosintáctico del texto, utilizando técnicas avanzadas como la tokenización, etiquetado de partes de la oración (POS tagging) y análisis sintáctico. Además, se definieron reglas adicionales para identificar entidades clave y excluir elementos irrelevantes.

Este enfoque se alinea con una combinación de:

* **Aprendizaje Supervisado**: Utilización del modelo preentrenado de SpaCy relativamente ligero.
* **Basado en Reglas**: Personalización mediante reglas gramaticales para filtrar y mejorar la precisión en la identificación de entidades.

### Extracción de relaciones

La extracción de relaciones es una tarea fundamental en PLN, que busca identificar conexiones semánticas entre entidades mencionadas en un texto. Este proceso resulta clave en la construcción de grafos de conocimiento y otras aplicaciones como los sistemas de recomendación, búsqueda semántica y análisis de datos estructurados ​(3)​(4)

Consiste en identificar el vínculo entre dos entidades en un texto, clasificando la relación para estructurarla en formato de tripletas (entidad1, relación, entidad2). Por ejemplo, en la frase *"Marie Curie ganó el Premio Nobel"*, las entidades *"Marie Curie"* y *"Premio Nobel"* están conectadas por la relación *"ganó"*. Este vínculo semántico se convierte en una arista dentro de un grafo de conocimiento​(3, 4).

Los métodos modernos han evolucionado desde enfoques basados en reglas hasta modelos de aprendizaje profundo que aprovechan los Transformers, como BERT y GPT. Estas tecnologías han permitido una mejor comprensión contextual, mejorando significativamente la precisión y la cobertura de las relaciones extraídas​ (3, 4)

Además, investigaciones recientes han destacado cómo los modelos basados en aprendizaje profundo, combinados con enfoques simbólicos, pueden mejorar significativamente la precisión en dominios específicos, como la literatura científica o los textos legales(4, 9).

A continuación, se describen diversos enfoques utilizados para abordar esta tarea:

**Métodos Híbridos**

Un enfoque híbrido es aquel que combina diferentes técnicas o metodologías para aprovechar las fortalezas individuales de cada una y compensar sus debilidades.

El método híbrido propuesto por (21) combina técnicas basadas en patrones lingüísticos con enfoques estadísticos y de aprendizaje automático para identificar relaciones semánticas entre entidades. Esta integración permite aprovechar las fortalezas de ambos enfoques: la precisión de las reglas lingüísticas y la generalización de los modelos estadísticos.

En este artículo, se describe un pipeline híbrido que:

1. Utiliza parsers sintácticos y sistemas de reconocimiento de entidades nombradas (NER) para identificar entidades clave en el texto.
2. Aplica patrones basados en gramática y conocimiento lingüístico para identificar relaciones preliminares.
3. Ajusta los resultados utilizando modelos estadísticos para aumentar la cobertura y resolver ambigüedades.

Requerimientos computacionales: Medio a Alto, debido al uso intensivo de recursos lingüísticos y bases de conocimiento.

**Métodos Basados en Kernels de Árbol**

Este enfoque, empleado por (22), utiliza estructuras de árboles enriquecidas con información sintáctica y semántica para capturar relaciones. Se introduce un *Context-Sensitive Convolution Tree Kernel* que considera nodos ancestrales en la estructura del árbol para mejorar la precisión en la detección de relaciones, este permite integrar tanto información sintáctica como semántica al calcular la similitud entre subestructuras en los árboles de dependencia.

Los kernels de árbol calculan similitudes entre las estructuras jerárquicas de oraciones, enfocándose en las subestructuras comunes (subárboles) que pueden representar relaciones.

Pertenece al aprendizaje supervisado, con un enfoque estadístico-estructural utilizando modelos como SVM para clasificar relaciones en base a similitudes estructurales.

Requerimientos computacionales: Alto, ya que implica cálculos complejos en árboles sintácticos y estructuras jerárquicas de datos.

**Extracción Basada en Gramáticas de Dependencia**

Los métodos basados en gramáticas de dependencia exploran las relaciones gramaticales directas entre palabras en una oración. Utilizan parsers de dependencia que identifican relaciones entre palabras en forma de grafos dirigidos, donde las entidades están conectadas por verbos u otros nodos intermedios.

En (23), se describe un sistema que utiliza patrones definidos sobre la gramática de dependencia para identificar tripletas sujeto-predicado-objeto en textos. Este enfoque es particularmente útil en escenarios de extracción no supervisada y abierto a múltiples dominios​

Requerimientos computacionales: Medio, al aprovechar parsers de dependencia como el Link Grammar Parser y procesos de clustering.

**Métodos Basados en Patrones Lingüísticos**

Este enfoque utiliza patrones predefinidos (reglas basadas en expresiones regulares o sintácticas) para identificar relaciones entre entidades en un texto. Estos patrones son diseñados manualmente o generados a partir de corpus anotados  
En (24) se combina herramientas como MetaMap y patrones lingüísticos diseñados manualmente para identificar relaciones entre entidades médicas, utilizando información del UMLS Metathesaurus. Este enfoque prioriza precisión sobre cobertura, lo cual es crucial en aplicaciones biomédicas.

Requerimientos computacionales: Bajo a Medio, dependiendo de la complejidad del corpus y los patrones utilizados.

**Métodos Basados en Redes Neuronales Profundas (*Deep Learning*)**

Estos métodos emplean modelos de aprendizaje profundo para aprender representaciones complejas del lenguaje y extraer relaciones semánticas entre entidades. Utilizan grandes cantidades de datos para entrenar modelos capaces de generalizar a nuevas instancias.

El estudio de (25) desarrolla una metodología para la detección de entidades y la extracción de relaciones semánticas en textos no estructurados del dominio de salud, a través de modelos basados en redes neuronales recurrentes.

Aplicable en idioma español, siempre que se utilicen corpus de entrenamiento en español y modelos adaptados al idioma.

Requerimientos Computacionales: Alto. Requiere hardware especializado, como GPUs, y recursos significativos para el entrenamiento y la inferencia de los modelos.

**Comparativa de Métodos**

|  |  |  |  |
| --- | --- | --- | --- |
| Método | Ventajas | Limitaciones | Requerimientos |
| Híbrido | Alta cobertura y precisión con patrones definidos. | Requiere recursos lingüísticos extensos. | Medio-Alto |
| Kernels de Árbol | Captura información estructural y contextual compleja. | Alta demanda computacional y ajuste fino de parámetros. | Alto |
| Gramática de Dependencia | Generalizable a dominios abiertos. | Depende de la calidad del parser y patrones predefinidos. | Medio |
| Patrones Lingüísticos | Alta precisión en dominios específicos como la biomedicina. | Limitada generalización fuera del dominio médico. | Bajo-Medio |
| Redes Neuronales Profundas | Representación compleja y capacidad de generalización a nuevas instancias. | Requiere recursos computacionales significativos. | Alta |

Tabla Comparativa de métodos para la extracción de relaciones semánticas

Sin embargo, con modelos más ligeros, un enfoque de deep learning puede ejecutarse en sistemas de hardware moderado. Un modelo como **Llama2:7b** puede aprovechar las ventajas del enfoque de aprendizaje profundo en entornos de bajos recursos mediante su diseño compacto y optimizado para tareas específicas.

Al ser significativamente más pequeño que otros modelos de lenguaje de gran escala, Llama2:7b reduce la demanda de hardware avanzado, permitiendo su ejecución en dispositivos con recursos limitados, como ordenadores con 8GB de RAM. Además, su capacidad de ajuste a través de prompts específicos elimina la necesidad de un costoso entrenamiento adicional, permitiendo la extracción de relaciones semánticas con alta precisión utilizando solo el modelo preentrenado.

Con el uso de frameworks eficientes como ONNX Runtime, PyTorch Lite, o LangChain se puede maximizar el rendimiento del modelo en tareas de inferencia, ofreciendo una solución accesible para aplicaciones de procesamiento de lenguaje natural en gran variedad de dominios.

## Representación y visualización del grafo

La visualización de grafos es una técnica que permite representar datos estructurados en forma de nodos y aristas, facilitando la comprensión de relaciones complejas en diversas disciplinas, como la informática, las ciencias sociales y la biología. Esta representación gráfica ayuda a identificar patrones, conexiones y estructuras que serían difíciles de discernir mediante datos tabulares o textuales.

**Técnicas Comunes de Visualización de Grafos:**

* **Diagramas de nodos y enlaces:** Representan entidades como nodos y sus relaciones como aristas, proporcionando una visión clara de las conexiones directas.
* **Matrices de adyacencia:** Utilizan una matriz para mostrar las relaciones entre nodos, siendo útiles para grafos densos donde los diagramas de nodos y enlaces pueden volverse confusos.

**NetworkX** y **Matplotlib** son dos bibliotecas clave utilizadas en este proyecto para manejar la construcción y visualización de los grafos de conocimiento. Estas herramientas trabajan en conjunto para crear, estructurar y representar gráficamente los datos extraídos del texto, ofreciendo flexibilidad y claridad en la visualización.

NetworkX es la biblioteca responsable de modelar y gestionar la estructura del grafo. Su principal función en el proyecto es la construcción del grafo, en el cual los nodos representan entidades, y las aristas, las relaciones entre ellas.

Matplotlib es la biblioteca utilizada para la representación visual del grafo generado por NetworkX. En el proyecto, se emplea para:

* **Dibujo de Nodos y Aristas**: Representar las entidades y relaciones mediante gráficos comprensibles.
* **Etiquetado**: Mostrar etiquetas tanto en los nodos como en las aristas para describir las entidades y relaciones de manera clara.
* **Configuración Gráfica**: Ajustar el tamaño, colores y posiciones de los elementos del grafo para optimizar su legibilidad y estética general.

A continuación, se muestra una imagen de ejemplo de la visualización deun grafo de conocimiento sencillo con estas tecnologías:

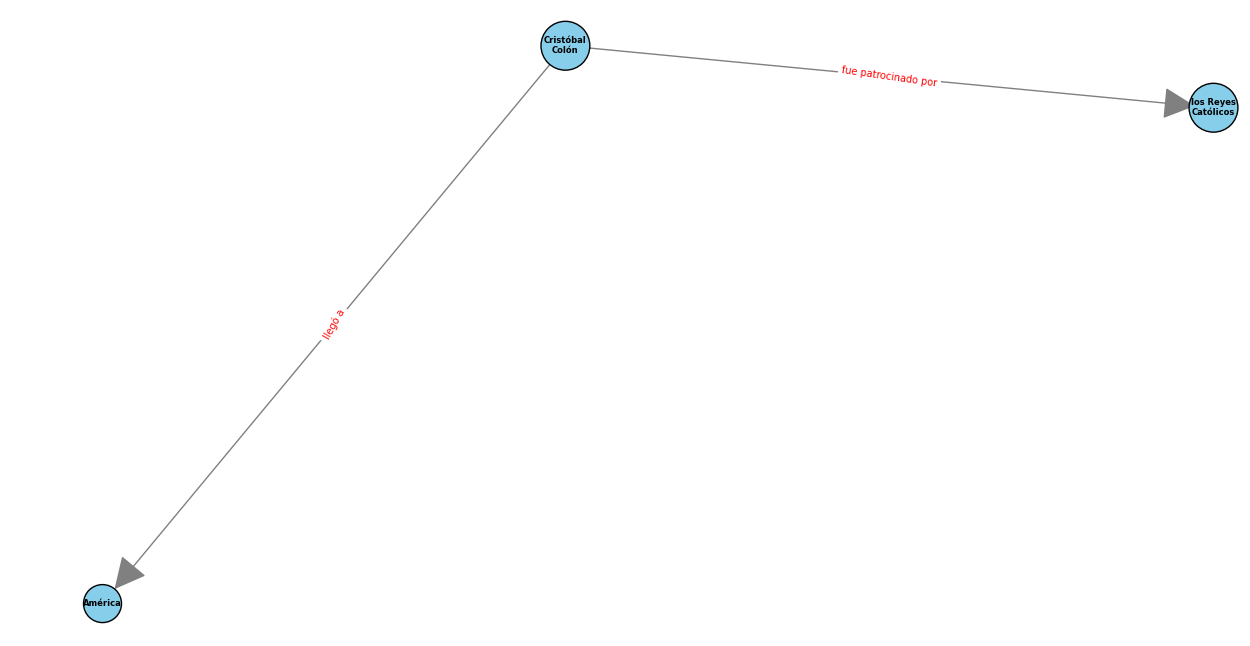


Ilustración : Visualización simple de un grafo de conocimiento

### Heurística de optimización visual del grafo

Según (26), una heurística es una regla o estrategia simple que se utiliza para tomar decisiones y resolver problemas de manera eficiente, ignorando parte de la información disponible. Estas estrategias están diseñadas para funcionar bajo condiciones de limitación de recursos cognitivos y tiempo, siendo especialmente útiles en contextos de incertidumbre.

Las heurísticas no son simplemente atajos; son herramientas poderosas que aprovechan la estructura del entorno para tomar decisiones rápidas, eficientes y, en muchos casos, sorprendentemente precisas. Este enfoque desafía la creencia de que un análisis exhaustivo siempre produce mejores resultados, mostrando que "menos puede ser más."

**Aspectos y características clave**

1. **Simplicidad:** Las heurísticas son reglas simples que no requieren cálculos complejos o el análisis de grandes cantidades de información, lo que las hace accesibles y rápidas.
2. **Eficiencia:** Están diseñadas para tomar decisiones en un tiempo reducido, maximizando la utilidad con recursos limitados. Esto las hace aplicables en situaciones del mundo real, como en medicina, negocios o vida cotidiana.
3. **Adaptación al entorno:** Las heurísticas no son universales; funcionan mejor en entornos donde las reglas simples coinciden con las estructuras estadísticas del problema. Esto se conoce como "ajuste ecológico."
4. **Ignorar para ganar:** Las heurísticas ignoran deliberadamente parte de la información para simplificar el proceso de toma de decisiones. Esto puede conducir a resultados más precisos y robustos en comparación con enfoques más complejos.

Con base en el artículo *'Circular Layout in the Graph Layout Toolkit' (27)*, el cual explora la implementación y los beneficios del diseño circular en la visualización de grafos, se desarrolló una implementación para optimizar la representación visual de los grafos de conocimiento generados en este proyecto. Este enfoque se fundamenta en las ventajas ofrecidas por el diseño circular, como la distribución equitativa de los nodos, la mejora de la simetría y la claridad en la disposición de las relaciones, contribuyendo significativamente a una interpretación más clara de las estructuras complejas del grafo.

Esta heurística se utiliza para distribuir y optimizar la posición de nodos en un espacio gráfico, utilizando círculos concéntricos como base para su disposición inicial y una estrategia iterativa para mejorar la configuración. En la etapa de distribución inicial, si el número de nodos es de 5 o menos, se colocan equidistantemente en un solo círculo. En caso de que haya más de 5 nodos, se emplean dos círculos concéntricos: un círculo interior para la mitad de los nodos y un círculo exterior para los restantes. Cada nodo es colocado calculando su ángulo correspondiente dentro de su círculo, asegurando una distribución uniforme.

La optimización se realiza mediante iteraciones donde dos nodos son seleccionados aleatoriamente para intercambiar sus coordenadas. Antes de confirmar el cambio, se evalúa una función objetivo (que mide la calidad de la distribución) tanto antes como después del intercambio. Si el intercambio mejora el valor de la función objetivo, se acepta; en caso contrario, se revierte el cambio, preservando la configuración previa. Este enfoque asegura una mejora gradual de la distribución de los nodos mientras mantiene una estructura inicial ordenada y lógica.

A continuación, se aprecia los cambios de un grafo con varias relaciones antes y después de la ejecución de la heurística implementada:

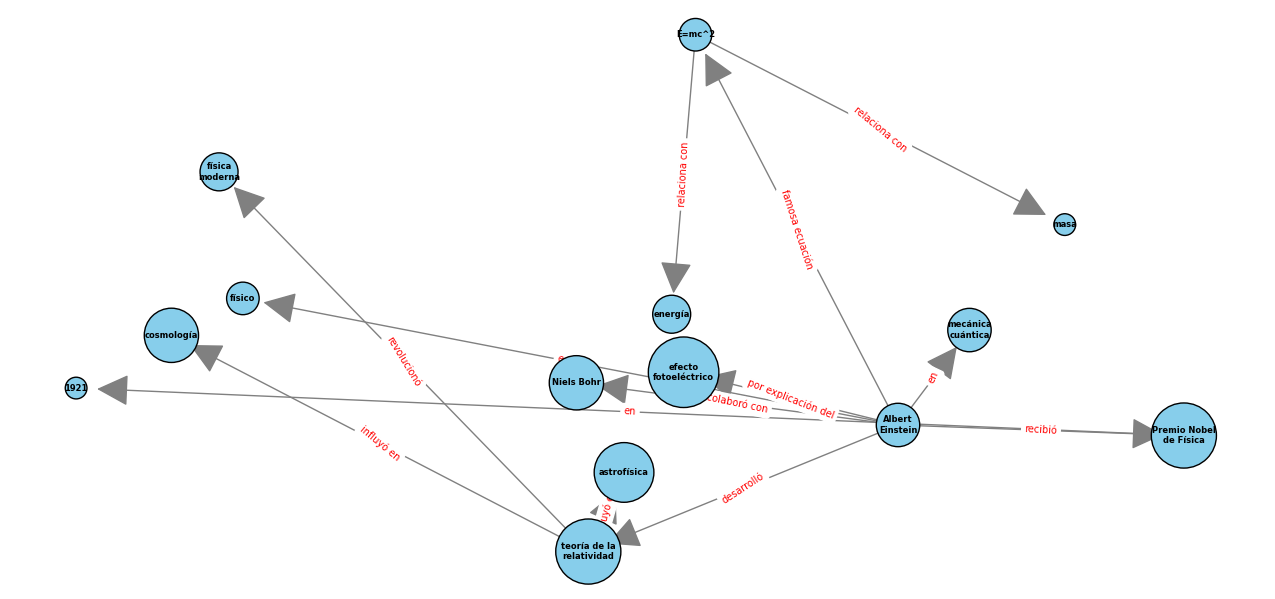


Ilustración : Grafo antes de la ejecución del método heurístico

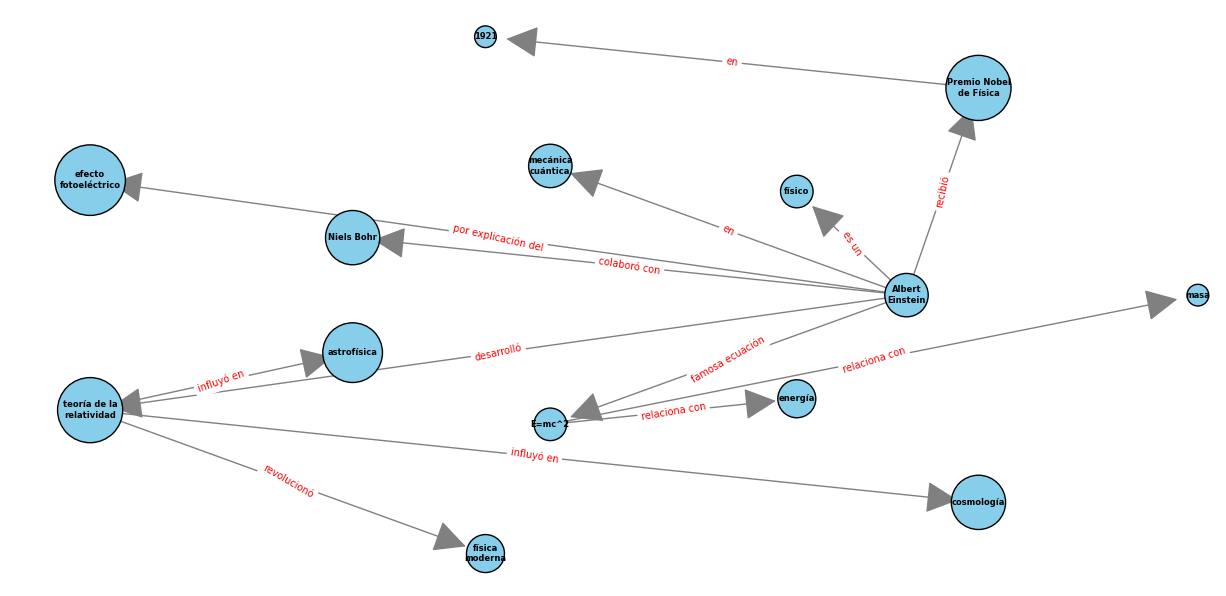


Ilustración : : Grafo después de la ejecución del método heurístico

## Comparación general con otros enfoques

A continuación, se presenta una tabla comparativa que incluye métodos tradicionales (sin aprendizaje profundo), métodos basados en deep learning y el método propuesto en el proyecto. Esta comparación abarca aspectos como metodología, rendimiento computacional, precisión, y aplicabilidad, con referencias específicas de los artículos revisados.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Aspecto | Método Propuesto | Métodos Basados en Patrones Léxicos para la Extracción de Información(28) | Grapher (15) | Graph Transformer (GTR) (29) | AutoKG(5) |
| Metodología | Modular pipeline con extracción de entidades (spaCy), relaciones (Llama2), y optimización gráfica mediante heurística. | no aborda la generación completa de grafos de conocimiento, sino que se enfoca en la extracción de relaciones semánticas explícitas. Utiliza patrones léxicos supervisados para identificar relaciones semánticas en texto, sin recurrir a deep learning | Divide la generación de grafos en dos etapas: nodos (con LMs) y aristas (clasificadores). | Transformer para transformar grafos en estructuras complejas utilizando mecanismos de atención intra e inter-grafos. | Emplea múltiples agentes LLM (GPT-4 y T5) colaborando para identificar entidades y relaciones, corrigiendo errores iterativamente. |
| Rendimiento Computacional Mínimo | Moderado; Llama2 y spaCy son funcionales en hardware estándar. | Bajo: Métodos simples que funcionan en hardware estándar. | Moderado; basado en preentrenamiento de modelos como T5, pero no tan intensivo como Transformers completos. | Alto; el procesamiento paralelo intensivo y las operaciones gráficas demandan una infraestructura avanzada. | Muy Alto; requiere hardware especializado para manejar razonamiento iterativo y múltiples pasos de verificación. |
| Precisión | Alta para textos narrativos y académicos en español. | Moderada: Dependiente de la calidad de los patrones léxicos, adecuada para dominios bien estructurados. | Alta para textos estructurados; precisión limitada en relaciones implícitas. | Alta para grafos estructurados y relaciones complejas, menos efectiva en texto no estructurado. | Muy Alta; precisión sobresaliente debido a la verificación iterativa y razonamiento avanzado. |
| Flexibilidad | Altamente adaptable a texto en español, narrativo y técnico. | Alta: Fácil adaptación a diferentes idiomas y dominios mediante la personalización de patrones. | Moderada; se adapta mejor a dominios específicos con datos preentrenados. | Moderada; puede procesar datos heterogéneos, pero requiere configuraciones específicas para texto no estructurado. | Amplia; ajustable a múltiples dominios con razonamiento inferencial incorporado. |
| Dominio de Aplicación | Generación de grafos de conocimiento en español para aplicaciones académicas, técnicas y narrativas. | Textos estructurados o semi-estructurados con relaciones explícitas. | Textos académicos y científicos con datos estructurados. | Ciencias y dominios médicos; aplicaciones con grafos predefinidos. | Bases de conocimiento dinámicas y corporativas; aplicaciones sociales complejas. |
| Ventajas | Modular y optimizado para bajo costo computacional. | |  | | --- | |  |  |  | | --- | | Simplicidad, bajo costo computacional, y adaptabilidad. | | Proceso eficaz gracias a la división en nodos y aristas, evitando redundancias. | Altamente eficaz en dominios con estructuras gráficas ricas. | Combinación de razonamiento y verificación asegura precisión en aplicaciones críticas. |
| Desventajas | Menos efectivo en dominios muy técnicos. El modelo Llama seleccionado, al ser más sencillo puede dar resultados no tan buenos. | Limitado en textos complejos o no estructurados; no captura relaciones implícitas. | Menos efectivo para dominios con datos no estructurados y relaciones implícitas. | Complejidad computacional y altos requerimientos de hardware. | Costoso y dependiente de instrucciones precisas para los agentes LLM. |

Tabla : Comparativa general con otros enfoques

**Análisis Comparativo**

1. **Adaptabilidad al español**:
   * El método propuesto destaca por su capacidad para procesar texto en español, una característica que los métodos tradicionales y algunos enfoques basados en Transformers no priorizan​.
2. **Eficiencia Computacional**:
   * Los métodos tradicionales son los más ligeros en términos de demanda computacional, pero carecen de la flexibilidad y adaptabilidad de los métodos basados en deep learning o el propuesto en este proyecto​.
3. **Flexibilidad y Dominio**:
   * Aunque Grapher y AutoKG son más avanzados, el método propuesto logra un balance entre precisión y adaptabilidad, especialmente en dominios narrativos y académicos en español​.

## Conclusiones parciales

El capítulo 1 ha permitido explorar los fundamentos teóricos y metodológicos que sustentan este trabajo, abordando conceptos clave, herramientas tecnológicas y enfoques aplicados. A partir de este análisis, se desprenden reflexiones que sintetizan los principales logros alcanzados y los desafíos identificados, ofreciendo una base sólida para el desarrollo de las etapas siguientes.

* Se estableció una base teórica sólida sobre los grafos de conocimiento, destacando su utilidad en la representación estructurada y semántica de información en diversos dominios.
* Se identificaron los principales retos para la generación automática de grafos de conocimiento en entornos de bajos recursos, especialmente en el procesamiento de texto en español.
* La integración de herramientas como spaCy y Llama2:7b se identificó como una solución posible para la tarea
* El desarrollo de heurísticas para la optimización visual de grafos contribuye significativamente a la claridad y comprensión de las estructuras generadas.
* El análisis comparativo evidenció que el enfoque propuesto logra un balance adecuado entre precisión, eficiencia computacional y aplicabilidad en entornos con recursos limitados.
* Se sentaron las bases metodológicas y técnicas para avanzar hacia la implementación de una solución práctica, accesible y ligera para la generación de grafos de conocimiento en español.

# Desarrollo de la solución propuesta

La solución propuesta en este proyecto fue desarrollada con el objetivo principal de implementar un sistema para la construcción automatizada de grafos de conocimiento a partir de texto en español, buscando la funcionalidad en entornos de bajos recursos. Este sistema responde a la necesidad de transformar información no estructurada en representaciones gráficas estructuradas que faciliten el análisis y la interpretación de datos.

La solución está orientada a proporcionar una herramienta accesible y flexible, capaz de adaptarse a diversos escenarios. Su diseño modular permite que las distintas etapas del proceso sean fácilmente escalables o ajustables.

En las siguientes subsecciones se presentan los detalles sobre la arquitectura del sistema, los métodos y tecnologías implementados, así como una descripción general de los componentes clave que conforman esta propuesta.

## Descripción general de la solución

La solución propuesta es un sistema modular y ligero que automatiza la construcción de grafos de conocimiento a partir de texto en español, diseñado para operar en entornos con recursos computacionales limitados. Se fundamenta en el uso de tecnologías avanzadas como spaCy para la extracción de entidades, el modelo Llama2:7b para la identificación de relaciones, y algoritmos heurísticos para la optimización visual.

En su diseño, se priorizó el uso de herramientas y modelos ligeros. La integración de tecnologías avanzadas como modelos de lenguaje preentrenados, bibliotecas para procesamiento de lenguaje natural y algoritmos heurísticos de visualización permite abordar los retos inherentes al procesamiento de texto en español y a la generación de representaciones gráficas semánticas.

El sistema utiliza spaCy para identificar entidades nombradas en el texto, aplicando modelos preentrenados adaptados al español. Esto permite extraer conceptos clave de manera precisa, incluso en textos complejos. Las relaciones entre entidades se determinan mediante el modelo Llama2:7b, configurado para generar conexiones semánticas basadas únicamente en información explícita o implícita en el texto, asegurando la coherencia del grafo.

La representación visual del grafo se realiza mediante NetworkX y Matplotlib, con un algoritmo heurístico que optimiza la disposición de nodos y aristas, reduciendo cruces y mejorando la claridad de las conexiones. Esta optimización permite que los grafos generados sean más legibles y útiles para su análisis, incluso en escenarios con estructuras complejas.

Finalmente, la solución incluye una interfaz gráfica desarrollada con Tkinter que simplifica la interacción del usuario. Esta herramienta permite procesar textos ingresados manualmente o cargados desde archivos.

EL sistema exporta los grafos generados en formatos como CSV, haciéndolos accesibles para diversas aplicaciones en educación, investigación y proyectos empresariales.

### Requisitos del sistema

El sistema fue desarrollado buscando el cumplimiento de los siguientes requisitos:

* 1. **Requisitos funcionales**
* **R1:** El sistema debe ser capaz de extraer entidades y conceptos claves en un texto en español.
* **R2:** El sistema debe identificar relaciones semánticas a partir de un texto.
* **R3:** El sistema debe permitir la entrada de texto a través de dos modalidades: (a) entrada manual de texto y (b) importación de archivos.
* **R4:** El sistema debe procesar las entidades y relaciones identificadas para construir un grafo de conocimiento.
* **R5:** El sistema debe visualizar el grafo de conocimiento de manera clara.
* **R6:** El sistema debe exportar las tripletas del grafo a un archivo CSV con un formato predefinido.
* **R7:** El sistema debe brindar retroalimentación visual sobre la ejecución del sistema.
  1. **Requisitos no funcionales**

**Requisitos de calidad**

* **Eficiencia**
  + **R8:** El sistema debe generar grafos en un entorno de hardware de gama baja (procesador Intel i3, 8GB RAM).
* **Usabilidad**
  + **R9:** La interfaz gráfica debe ser intuitiva, permitiendo a los usuarios cargar texto o archivos sin necesidad de conocimientos técnicos.
  + **R10:** La visualización del grafo debe ser entendible.
  + **R11:** El grafo debe exportarse en un formato estándar que permita su posterior uso.

**Restricciones**

* **R12:** El sistema debe ejecutarse en hardware estándar o de bajos recursos, sin depender de GPU para la inferencia del modelo.
* **R13:** La visualización del grafo debe representarse como un grafo dirigido, con aristas etiquetadas según las relaciones identificadas.

### Casos de uso del sistema

Para garantizar una adecuada comprensión de las funcionalidades y los requisitos del sistema, se ha elaborado un diagrama de casos de uso que representa las interacciones clave entre el usuario y las principales funcionalidades del sistema. A continuación, se presenta el diagrama acompañado de una descripción detallada de cada caso de uso:

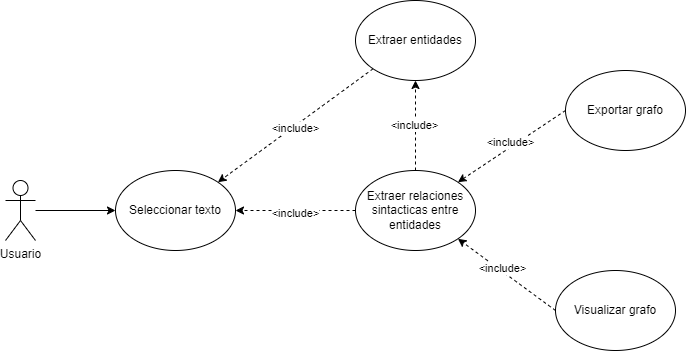


Ilustración : Diagrama de casos de uso del sistema

**Caso de uso: Seleccionar texto**

|  |  |
| --- | --- |
| Elemento | Descripción |
| Actores | Usuario (Inicia) |
| Descripción | El caso de uso comienza cuando el usuario ejecuta el programa. El usuario selecciona si quiere importar un archivo de texto o si va a introducir un texto manual directamente en un cuadro de texto. El proceso finaliza cuando el texto cumple con la cantidad de caracteres posibles y es enviado a la siguiente etapa del sistema. |
| Req Funcionales | R3, R7 |
| Casos de uso asociados | - Extraer entidades (include)  - Extraer relaciones sintácticas entre entidades (include) |
| Precondiciones | - La interfaz gráfica debe estar operativa.  - Si desea importar un archivo este está disponible en el sistema de archivos del usuario.  -Si desea importar un archivo este está en un formato compatible (txt o docx) |
| Postcondiciones | - Texto ingresado listo para la extracción de entidades y relaciones. |
| Req De Calidad | - Usabilidad: La interfaz debe ser intuitiva y permitir la entrada de texto sin errores (R9). |

**Caso de uso: Extraer entidades**

|  |  |
| --- | --- |
| Elemento | Descripción |
| Actores | --- |
| Descripción | Este caso de uso inicia cuando el texto es ingresado identificar entidades nombradas utilizando la biblioteca spaCy. Las entidades identificadas sirven como nodos en el grafo de conocimiento. El caso de uso finaliza cuando las entidades son extraídas del texto. |
| Req Funcionales | R1 |
| Casos de uso asociados | - Extraer relaciones sintácticas entre entidades (include) |
| Precondiciones | - Un texto debe estar disponible para ser procesado. |
| Postcondiciones | - Entidades extraídas para su posterior uso en la construcción del grafo. |
| Req De Calidad | - Eficiencia: El sistema debe generar grafos en un entorno de hardware de gama baja (procesador Intel i3, 8GB RAM). (R8). |

**Caso de uso: Extraer relaciones sintácticas entre entidades**

|  |  |
| --- | --- |
| Elemento | Descripción |
| Actores | --- |
| Descripción | Este caso de uso inicia luego que es seleccionado el texto a procesar y son extraídas las entidades de este. Consiste en identificar relaciones semánticas entre las entidades extraídas previamente, utilizando prompts diseñados para el modelo Llama2:7b. Las relaciones identificadas forman las aristas del grafo de conocimiento. El caso de uso finaliza cuando las relaciones entre las entidades son extraídas del texto y llevadas a formato de grafo. |
| Req Funcionales | R2, R4 |
| Casos de uso asociados | -Visualizar grafo (include)  - Exportar grafo (include)  -Extraer entidades (include)  -Seleccionar texto (include) |
| Precondiciones | - Las entidades fueron previamente extraídas.  -El texto fue ingresado correctamente. |
| Postcondiciones | - Relaciones semánticas entre las entidades identificadas. |
| Req De Calidad | - Eficiencia: El sistema debe generar grafos en un entorno de hardware de gama baja (procesador Intel i3, 8GB RAM). (R8). |

**Caso de uso: Visualizar grafo**

|  |  |
| --- | --- |
| Elemento | Descripción |
| Actores | --- |
| Descripción | Este caso de uso inicia luego de que se ha contruido el grafo de conocimiento. Permite al usuario visualizar el grafo generado a partir del texto procesado. La visualización utiliza las bibliotecas NetworkX y Matplotlib, con optimización heurística para mejorar la disposición de nodos y aristas. Finaliza luego que la visualización es optimizada y presentada al usuario. |
| Req Funcionales | R5, R7 |
| Casos de uso asociados | - Extraer relaciones sintácticas entre entidades (include) |
| Precondiciones | - El grafo ha sido generado previamente. |
| Postcondiciones | - Grafo presentado visualmente al usuario de forma clara. |
| Req De Calidad | - Eficiencia: El sistema debe generar grafos en un entorno de hardware de gama baja (procesador Intel i3, 8GB RAM) (R8)  - Usabilidad: La visualización del grafo debe ser entendible. (R10) |

**Caso de uso: Exportar grafo**

|  |  |
| --- | --- |
| Elemento | Descripción |
| Actores | --- |
| Descripción | Este caso de uso inicia luego que el grafo es construido. Permite al usuario exportar las tripletas del grafo de conocimiento generado a un archivo CSV. El archivo contiene la estructura del grafo en un formato estándar y accesible. Finaliza con el grafo exportado en un formato de csv. |
| Req Funcionales | R6 |
| Casos de uso asociados | - Extraer relaciones sintácticas entre entidades (include) |
| Precondiciones | - El grafo ha sido generado previamente. |
| Postcondiciones | - Tripletas del grafo guardadas en un archivo CSV con un formato predefinido. |
| Req De Calidad | - Usabilidad: El grafo debe exportarse en un formato estándar que permita su posterior uso. (R11) |

## Diseño de la solución

El diseño de la solución se llevó a cabo siguiendo un enfoque modular, con el objetivo de garantizar la escalabilidad, la reutilización del código y la facilidad de mantenimiento. La modularidad permite dividir el sistema en componentes independientes, donde cada módulo aborda una funcionalidad específica, como la extracción de entidades, la identificación de relaciones o la visualización del grafo de conocimiento. Este enfoque no solo facilita la identificación y solución de problemas durante el desarrollo, sino que también permite futuras ampliaciones y ajustes sin afectar la estructura general del sistema. Además, al trabajar de forma modular, se logra una mayor claridad en la implementación, lo que mejora la colaboración entre los desarrolladores y la comprensión global del sistema.

### Flujo del proceso

**Flujo del Proceso:**

El sistema sigue un flujo estructurado basado en el enfoque modular, utilizando el patrón de filtros y tuberías para procesar texto y generar un grafo de conocimiento. A continuación, se describen las etapas del proceso representadas en el siguiente diagrama:

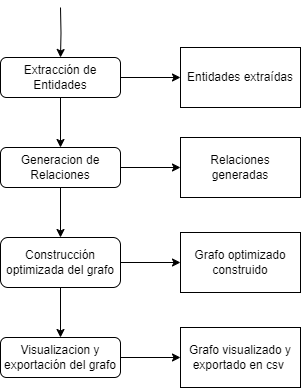


Ilustración : Flujo del sistema mediante el patrón de filtros y tuberías

1. **Extracción de Entidades:** El proceso comienza con la identificación de entidades clave en el texto utilizando técnicas de procesamiento del lenguaje natural (NLP). Este paso garantiza que se identifiquen correctamente los nodos que formarán parte del grafo de conocimiento.
2. **Generación de Relaciones:** Una vez identificadas las entidades, se analizan las conexiones semánticas entre ellas. Estas relaciones, derivadas del contexto del texto, constituyen las aristas del grafo. Para ello, se utilizan prompts diseñados específicamente para un modelo transformer, como Llama2:7b.
3. **Construcción Optimizada del Grafo:** Con las entidades y relaciones definidas, se construye un grafo inicial que es optimizada se disposición de los nodos mediante algoritmos heurísticos. La optimización mejora la disposición de los nodos y reduce problemas visuales como el cruce de aristas y el solapamiento de elementos.
4. **Visualización y Exportación del Grafo:** Finalmente, el grafo optimizado se visualiza de forma clara y accesible mediante herramientas como NetworkX y Matplotlib. Además, las relaciones estructuradas en formato de tripletas se exportan a un archivo CSV para facilitar su análisis y reutilización en otras aplicaciones.

Cada etapa del proceso está diseñada buscando la optimalidad en la generación del grafo, con el objetivo de que toda la información relevante del texto se represente de manera estructurada y comprensible.

### Arquitectura del sistema

Como se mencionó anteriormente, el sistema sigue una arquitectura modular, basada en capas. Por esta razón es útil para la mejor comprensión de sistema representarlo siguiendo un patrón N-capas con un enfoque de reutilización.

Desde un enfoque de reutilización, el patrón N-Capas fomenta el modularidad, ya que cada capa actúa como un componente independiente que puede ser desarrollado, modificado o reemplazado sin afectar a las demás. Por ejemplo, si se requiere un cambio en la capa específica, es posible que no necesario modificar la capa general. Este nivel de desacoplamiento no solo promueve la reutilización de código entre proyectos con requisitos similares, sino que también facilita la escalabilidad y la integración de nuevas funcionalidades sin necesidad de rediseñar todo el sistema. A continuación, se presenta el diagrama de este patrón presente en el proyecto:

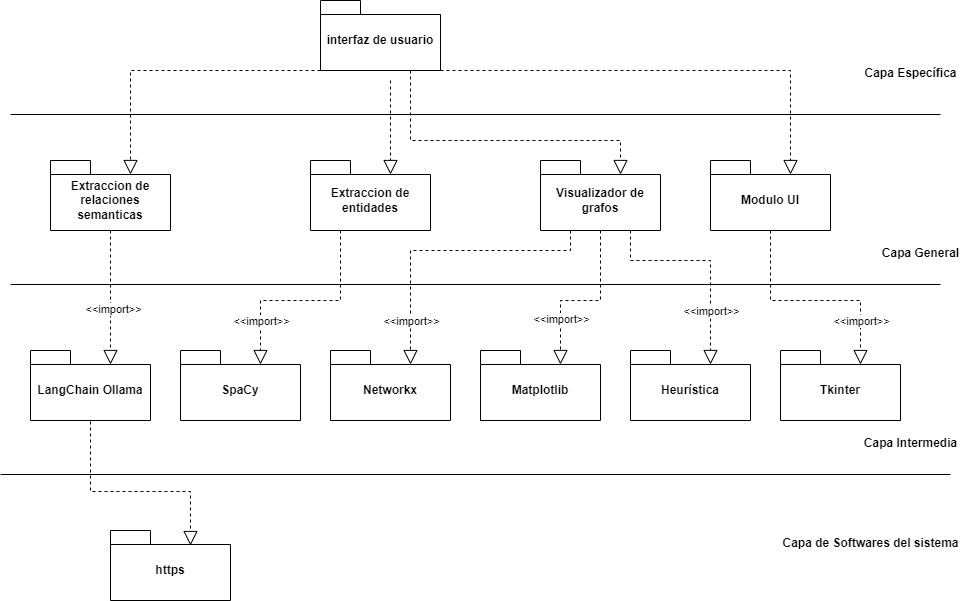


Ilustración : Patrón N-capas

**Capa Específica**

1. **Interfaz de Usuario (UI)**: Facilita la interacción del usuario con el sistema. Permite ingresar texto, importar archivos y visualizar el resultado final mediante botones y formularios gráficos (usando Tkinter).

**Capa General**

1. **Extracción de Relaciones Semánticas**: Usa modelos preentrenados (Llama2:7b a través de LangChain-Ollama) para identificar relaciones entre entidades extraídas del texto, formando las conexiones del grafo.
2. **Extracción de Entidades**: Implementa el modelo spaCy para identificar entidades nombradas y conceptos clave en el texto de entrada, utilizando reglas para filtrar entidades irrelevantes.
3. **Visualizador de Grafos**: Representa gráficamente los grafos generados utilizando NetworkX y Matplotlib, mejorando su claridad mediante un algoritmo heurístico para la disposición de los nodos y aristas.
4. **Módulo UI**: Gestiona la interfaz gráfica, facilitando opciones como la entrada manual o la carga de archivos y mostrando mensajes de progreso y resultados.

**Capa Intermedia**

1. **LangChain-Ollama:** Marco que conecta con el modelo Llama2:7b alojado en el servidor local de Ollama, permitiendo la extracción de relaciones semánticas entre entidades identificadas en el texto.
2. **spaCy**: Biblioteca para el reconocimiento de entidades nombradas, eficiente para procesar textos en español.
3. **NetworkX**: Permite construir, manipular y analizar grafos complejos para representar entidades y relaciones.
4. **Matplotlib**: Visualiza los grafos generados, personalizando la apariencia y mejorando la claridad.
5. **Heurística**: Implementa métodos para optimizar la disposición de nodos y reducir cruces entre aristas, utilizando criterios como distancia, solapamientos y ángulos pequeños.
6. **Tkinter**: Módulo gráfico que crea una interfaz visual simple pero funcional para interactuar con el sistema.

**Capa de Software del Sistema**

1. **Https:** Protocolo empleado por el paquete LangChain-Ollama para establecer comunicación con el servidor local de Ollama. A través de solicitudes se gestionan las consultas hacia los modelos alojados de forma local, garantizando la transmisión eficiente de datos entre la capa de software y la capa de aplicación.

### Modelo de dominio

Un modelo de dominio es una representación conceptual que describe los elementos clave y sus relaciones dentro de un sistema o área de interés específica. Su propósito principal es proporcionar una comprensión clara y compartida de cómo está estructurado el sistema y cómo interactúan sus componentes, lo que resulta crucial durante el diseño, desarrollo e implementación de proyectos de software o sistemas complejos. Un modelo de dominio abstrae los aspectos esenciales del problema que se busca resolver, eliminando detalles innecesarios para centrarse en los conceptos fundamentales.

El modelo de dominio es útil porque sirve como una guía para los desarrolladores, diseñadores y otras partes interesadas, ayudándoles a comprender cómo se deben organizar los componentes del sistema y sus interacciones.

A continuación, se presenta el modelo de dominio del sistema desarrollado:

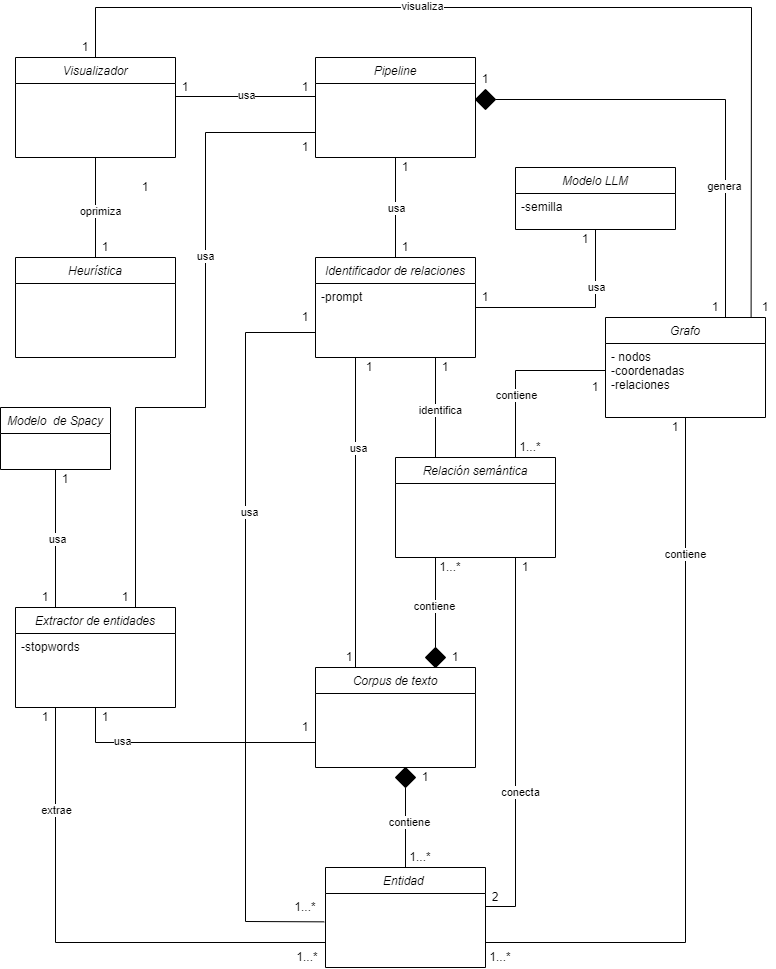


Ilustración : Modelo de dominio

En este modelo se representa lo siguiente:

El **Extractor de Entidades**, encargado de identificar **Entidades** clave a partir del **Corpus de Texto** (texto introducido por el usuario para procesar), utilizando reglas específicas y el **modelo de spaCy** de procesamiento de lenguaje natural. Estas entidades, que actúan como nodos, son enviadas junto con el contexto (el corpus de texto) al **Identificador de Relaciones**, que emplea un **modelo de lenguaje grande** (LLM, como Llama2:7b) para analizar las interacciones semánticas entre las entidades y generar **relaciones semánticas** precisas en forma de frases verbales, las cuales, junto con las entidades, conformarán el **grafo**. El **Pipeline** centraliza este proceso, gestionando la extracción, identificación y procesamiento de las entidades y relaciones para asegurar un flujo estructurado. La visualización del grafo resultante es optimizada por el módulo de **Heurística**, que mejora la disposición de los nodos y las aristas minimizando cruces y solapamientos para garantizar una representación visual clara. Finalmente, el **Visualizador** utiliza estas optimizaciones para generar una representación gráfica interactiva del grafo, facilitando la comprensión de las conexiones entre las entidades. Todo el sistema está diseñado para convertir un texto en un grafo de conocimiento completo y visualmente organizado, conectando entidades y relaciones derivadas directamente del contenido del corpus de texto inicial.

## Implementación de la extracción de información

El objetivo de esta sección es describir detalladamente cómo se lleva a cabo la extracción de información en el proyecto, cubriendo tanto la extracción de entidades como la identificación de relaciones. Estos procesos son fundamentales para la construcción automática de grafos de conocimiento a partir de texto.

### Extracción de entidades

La extracción de entidades se realiza utilizando **spaCy**, una biblioteca reconocida por su eficiencia en tareas de procesamiento del lenguaje natural. Se empleó el modelo preentrenado **es\_core\_news\_sm**, especializado en español, para identificar entidades relevantes en el texto procesado.

**Implementación:**

El módulo EntityExtraction.py define la clase EntityExtractor, la cual combina las capacidades del modelo preentrenado de spaCy con reglas personalizadas para maximizar la precisión de la extracción.

Se combina las capacidades de spaCy con reglas personalizadas para maximizar la precisión. La lógica se centra en identificar estructuras de texto que contengan sustantivos comunes (NOUN) o sustantivos propios (PROPN) como núcleo.

Estas reglas personalizadas permiten la identificación de entidades basadas en patrones gramaticales, como sustantivos comunes, sustantivos propios y adjetivos. También se consideran sintagmas nominales compuestos.

**Ejemplos de sintagmas identificados con la lógica implementada:**

1. Sustantivo como núcleo: "ciudad", "España" (Estructura: [NOUN] [PROPN] )

2. Sustantivo acompañado de determinantes o adjetivos: "la gran ciudad", “casa pequeña”, "un pequeño pueblo" (Estructura: [DET] [ADJ] [NOUN])

3. Sustantivo con modificadores numéricos: "tres ciudades importantes" (Estructura: [NUM] [NOUN] [ADJ])

La lógica excluye elementos que:

* No contienen al menos un sustantivo como núcleo.
* Son frases comunes sin valor como entidad (definidas en las "stop phrases").
* Incluyen puntuaciones o palabras irrelevantes para el contexto.

**Ejemplos Excluidos**

1. **Frases generales sin significado semántico relevante**:
   * *"por lo tanto"*, *"a pesar de"*, *"de hecho"*
   * Razón: Estas frases están en la lista de "stop phrases" y se consideran irrelevantes ​EntityExtraction.
2. **Tokens no relacionados con sustantivos**:
   * Adjetivos o números sin un sustantivo posterior.
   * *"grande"* (si no está acompañado de un sustantivo como "ciudad").
3. **Puntuaciones**:
   * Comas, puntos o signos de interrogación no se consideran parte de las entidades.
4. **Verbos y formas verbales**

Se definió una lista de "stop phrases" para excluir frases comunes en español que no aportan valor como entidades significativas, evitando así la inclusión de términos genéricos o conectores que podrían distorsionar el grafo.

Este enfoque personalizado en la extracción de entidades garantiza que los nodos del grafo representen conceptos clave y específicos del texto analizado, proporcionando una base sólida para la construcción del grafo de conocimiento.

**Ejemplo**

Texto de Entrada:*"La gran ciudad de Barcelona es conocida por su arquitectura única y su vibrante vida cultural."*

Tokens procesados:

* 'la', [DET]
* 'gran', [ADJ]
* 'ciudad', [NOUN]
* 'de', [ADP]
* 'barcelona', [PROPN]
* 'es', [AUX]
* 'conocida', [VERB]
* 'por', [ADP]
* 'su', [DET]
* 'arquitectura', [NOUN]
* 'única', [ADJ]
* 'y', [CCONJ]
* 'su', [DET]
* 'vibrante', [ADJ]
* 'vida', [NOUN]
* 'cultural', [ADJ]
* '.', [PUNCT]

Entidades identificadas: *“gran ciudad de Barcelona”, “arquitectura única”, “vibrante vida cultural”.*

**Limitaciones Identificadas:**

* Requiere ajustes adicionales para dominios específicos, como textos técnicos o jergas.

### Identificación de relaciones

Para la identificación de relaciones, se utilizó el modelo **Llama2:7b** a través de **LangChain-Ollama**, (componente especializado dentro del ecosistema de LangChain para trabajar con servidores locales como Ollama), esto permite generar relaciones entre entidades basándose en el contexto del texto. Este proceso utiliza prompts cuidadosamente diseñados para garantizar que las relaciones derivadas sean precisas y relevantes.

En el desarrollo del proyecto fueron probados varios prompts diferentes, con el propósito de identificar la variante con la que el modelo devolvía los mejores resultados. El prompt seleccionado es el siguiente:

**“Es muy importante que tu respuesta este completamente en idioma español!**

**Dado el texto:  {context}**

**Y la lista de entidades extraídas:  {', '.join(entities)}**

**Tu tarea es identificar las conexiones entre pares de entidades de la lista y devolverlas como una lista de conexiones**

**Es muy importante que las conexiones tengan especificamente la estructura que te defino al final del prompt y cda conexion este completa, es decir, con 1.nodo origen, 2.relacion, 3.nodo destino.**

**Sigue estas instrucciones con precisión:**

**ES MUY IMPORTANTE QUE TODAS LAS ENTIDADES DE LA LISTA APAREZCAN EN AL MENOS UNA CONEXION DE LAS QUE IDENTIFIQUES, NO PUEDE FALTAR NINGUNA, SINO ESTARÁ INCORRECTO.**

**Es muy importante que no añadas informacion adicional a la del texto dado, las relaciones deben ser unicamente extraídas del texto.**

**Es muy importante que respetes la lista de entidades y estas explicitamente sean los nodos del grafo sin cambios ni que dos entidades esten en un mismo nodo**

**es imprescindible que la respuesta, cada conexion (trilpeta de dos nodos y relacion) esten una a continuacion de la otra, en el formato imprescindible que te especifico a continuación**

**Es imprescindible que tu respuesta sea completa en idioma español**

**Todas las entidades de la lista deben aparecer como nodo en alguna conexion**

**- Analiza todas las combinaciones posibles de entidades para determinar si existe una relación entre ellas en el texto proporcionado y elaborar la conexion con la estructura necesaria.**

**- Si existe una relación, inclúyela. Si no existe, omítela.**

**- No dejes ninguna entidad de la lista sin aparecer en ninguna conexion, una entidad se puede relacionar con más de una entidad.**

**que serán los nodos:**

**- Los nodos deben ser exclusivamente las entidades de la lista proporcionada, todas las entidades deben aparecer representadas como nodos independientes.**

**- No omitas ninguna entidad**

**- Cada nodo debe corresponder exactamente al nombre de las entidades listadas, sin modificaciones ni interpretaciones adicionales.**

**- El nodo origen debe ser el que "ejecuta la acción", es importante el orden de los nodos porque de esto depende el sentido de la relación**

**que será la relación:**

**- Las relaciones deben ser \*\*verbos o frases verbales\*\* que describan cómo las entidades interactúan, están conectadas o se influyen entre sí, no deben ser oraciones largas, sustantivos ni conceptos abstractos.**

**- Las relaciones deben unir dos entidades unicamente, son la conexión presente en el texto dado entre ambas entidades**

**- En la etiqueta de la relación no debe aparecer el nombre de los nodos que conecta, solo la frase verban de la conexion entre ambos**

**ES MUY IMPORTANTE, LO MAS IMPORTANTE DE TU TAREA, ES QUE LAS CONEXIONES DE TUS RESPUESTAS TENGAN ESTA ESTRUCTURA, DE LO CONTRARIO NO SIRVE PARA NADA:**

**Cada conexion contiene estos tres elementos es sumamente importante, de eso depende que cumplas con tu tarea, que cada conexion este escrita exactamente con exta estructura:**

**1. Nodo origen: [Entidad A]**

**2. Relación: [Acción o vínculo de la Entidad A sobre la Entidad B (no incluye el nombre de las entidades en la relación)]**

**3. Nodo destino: [Entidad B]**

**1. Nodo origen: [Entidad A]**

**2. Relación: [Acción o vínculo de la Entidad A sobre la Entidad C (no incluye el nombre de las entidades en la relación)]**

**3. Nodo destino: [Entidad C]**

**...y asi sucesivamente para todas las conexiones idenificadas entre todas las entidades. es IMPRESCINDIBLE que esten con esta estructura exacta ya que sino no podré procesarlo**

**Procesa internamente en inglés si es necesario, pero devuelve la respuesta final en español.”**

El proceso incia al enviarle al modelo el contexto y las entidades extraídas junto con el prompt. Este genera una respuesta a partir de lo recibido, en la cual identifica las relaciones entre las entidades, que luego debe ser procesada para llevarla al formato estructurado del grafo.

Por ejemplo, la respuesta del modelo, usando el prompt seleccionado, a partir de las entidades: ['cristóbal colón', 'américa en 1492', 'reyes católicos']. Extraídas del texto: " Cristóbal Colón llegó a América en 1492. Fue patrocinado por los Reyes Católicos." Es la siguiente:

**“Entendido, seguiré tus instrucciones y analizaré las entidades mencionadas en el texto para identificar las conexiones entre ellas. A continuación, proporcionaré la lista de conexiones completa según la estructura que has definido:**

**Conexiones entre las entidades mencionadas en el texto:**

**1. Nodo origen: Cristóbal Colón**

**2. Relación: fue patrocinado por**

**3. Nodo destino: los Reyes Católicos**

**1. Nodo origen: Cristóbal Colón**

**2. Relación: llegó a**

**3. Nodo destino: América**

**Espero que esta lista de conexiones sea adecuada y cumplida con tus requerimientos. Si necesitas más información o tienes alguna pregunta, no dudes en hacerme saber.”**

Y las tripletas generadas luego de procesar esta respuesta serían:

**[('Cristóbal Colón', 'América', {'label': 'llegó a'}),**

**('Cristóbal Colón', 'los Reyes Católicos', {'label': 'fue patrocinado por'})]**

## Interfaz de usuario

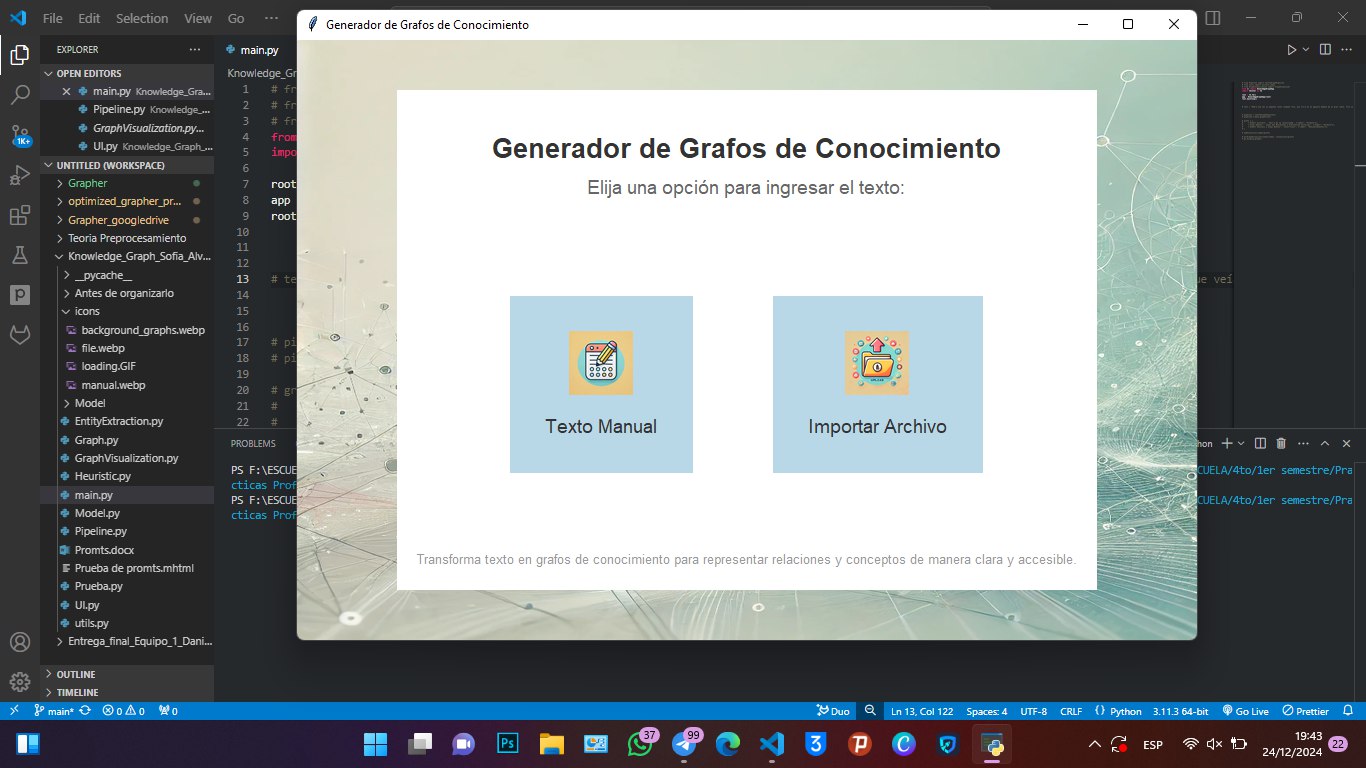
En esta sección se describirá el diseño de las interfaces del proyecto, destacando su estructura, organización y funcionalidad. Además, se demostrará el cumplimiento de las reglas de oro y los principios de diseño a través de ejemplos específicos tomados de las interfaces desarrolladas.

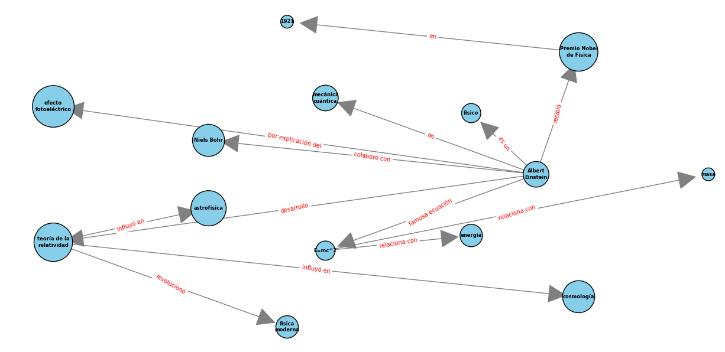
El diseño de las interfaces cumple con los estándares de usabilidad y estética necesarios para garantizar una experiencia de usuario óptima. Las reglas de oro y los principios de diseño se implementaron cuidadosamente. Esto asegura que el sistema sea funcional y accesible para todos los usuarios.

A continuación, se presenta cada una de las interfaces de manera organizada, clasificadas según su funcionalidad y relevancia dentro del sistema.

### Interfaces principales

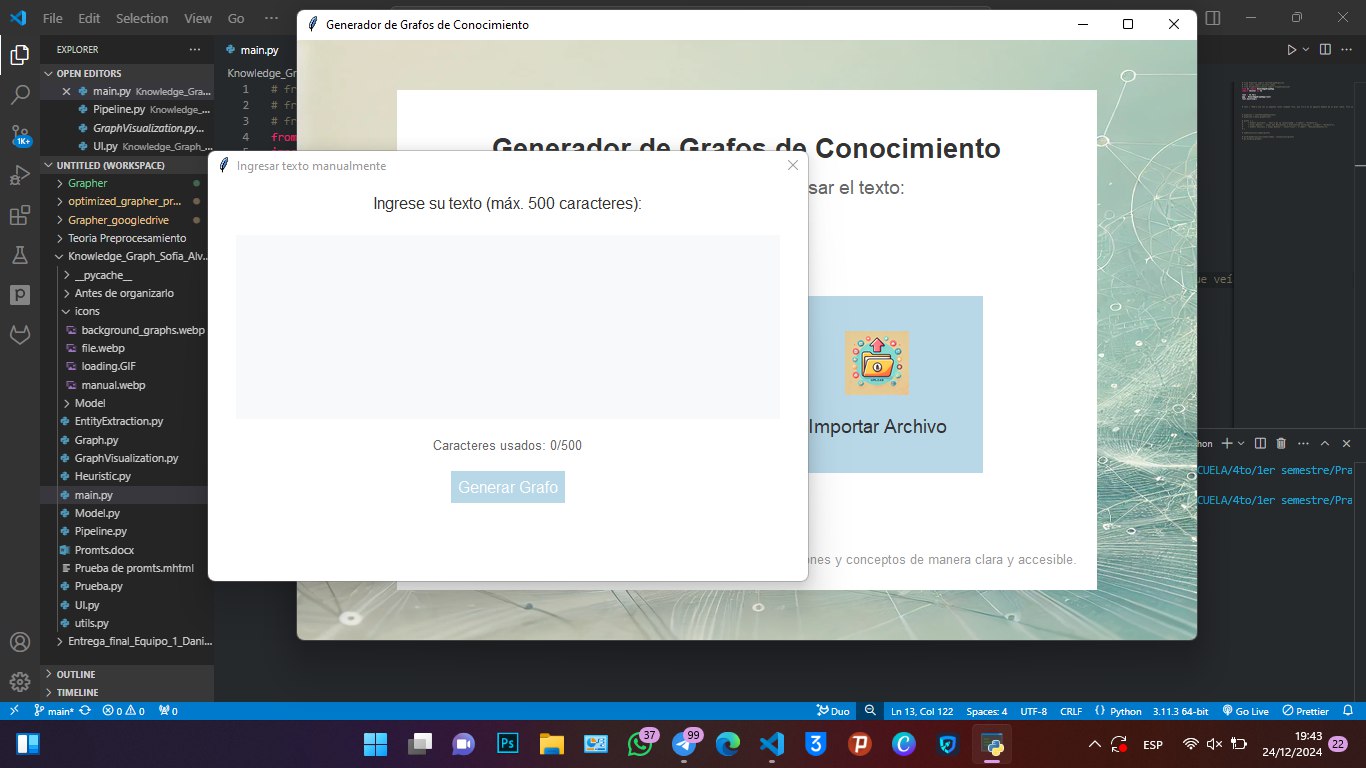
Estas interfaces representan los elementos centrales de interacción del usuario con el sistema.

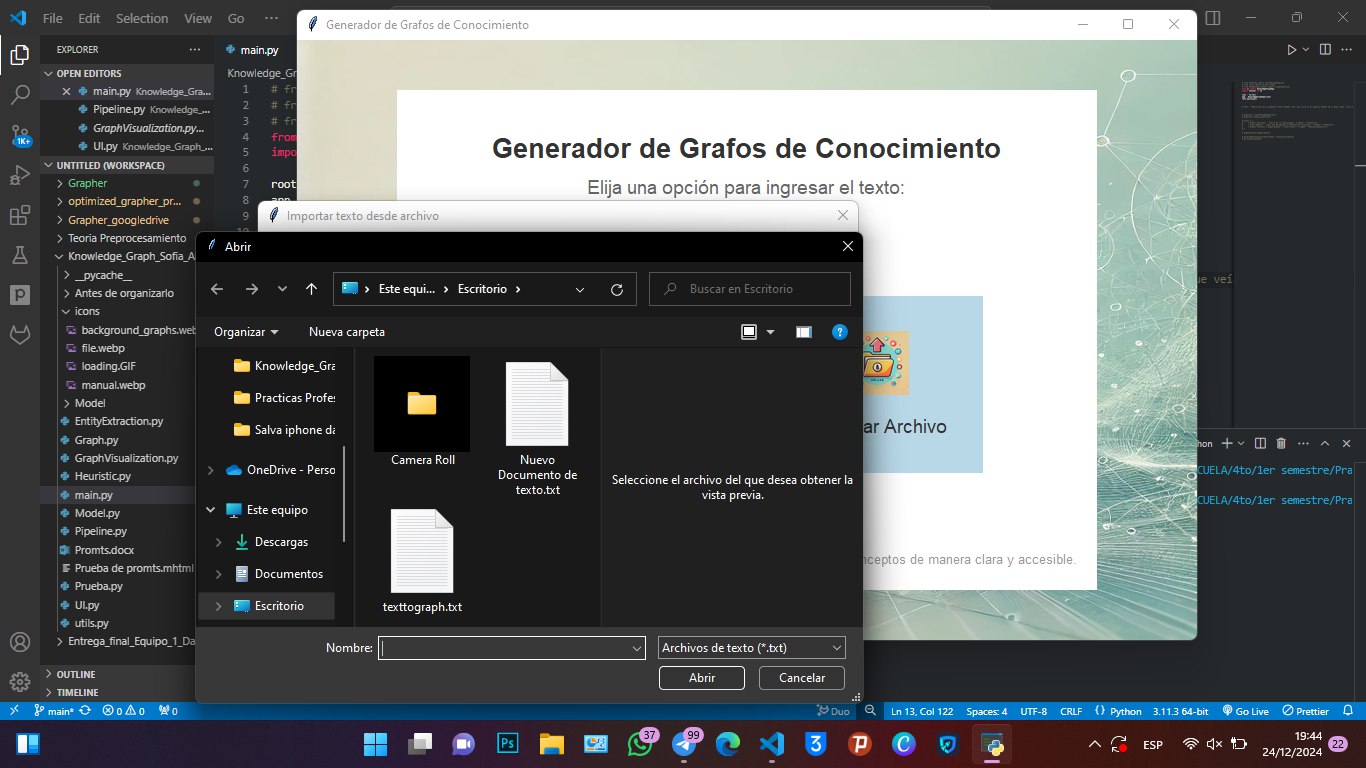
**Ventana Principal**: Permite al usuario acceder a las funcionalidades principales del sistema. Se cumple con los principios de consistencia y retroalimentación mediante una distribución clara de los elementos y retroalimentación visual al interactuar con ellos.

**Visualización del Grafo**: Muestra la representación gráfica de los datos procesados. Se promueve la estética y el diseño minimalista para facilitar la comprensión de los datos mostrados.

### Interfaces secundarias

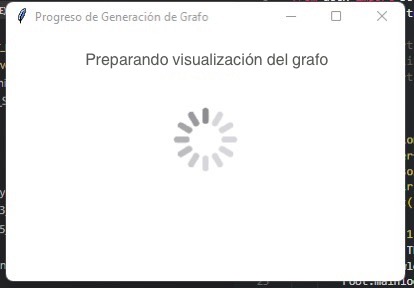
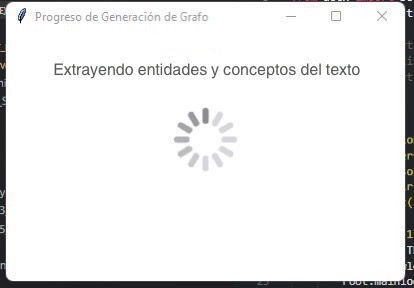
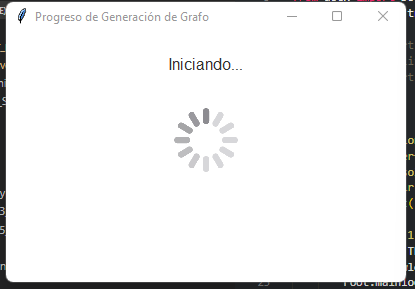
Estas interfaces complementan la funcionalidad del sistema y ofrecen soporte para tareas específicas.

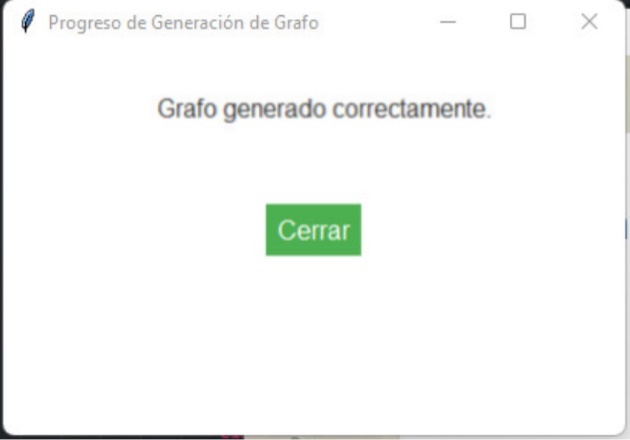
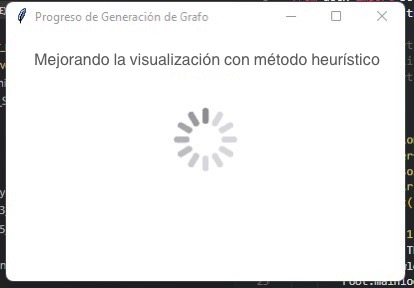
**Ventana de Entrada de Texto Manual**: Permite al usuario introducir el texto a procesar manualmente.

**Ventana de Selección de Archivo**: Facilita la carga de archivos de texto. El contenido luego de seleccionarse, puede editarse antes de procesarlo, en una ventana igual a la de ingresar texto manual. Cumple con el principio de control del usuario al permitir seleccionar archivos específicos de manera intuitiva.

### Retroalimentación al usuario

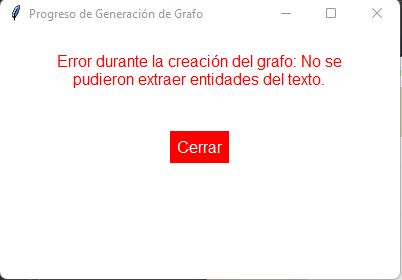
Estas interfaces aseguran que el sistema informe claramente al usuario sobre su estado y acciones. Se implementaron diversos mecanismos para informar al usuario de errores, progresos y resultados. Cumple con las reglas de retroalimentación inmediata y visibilidad del estado del sistema.

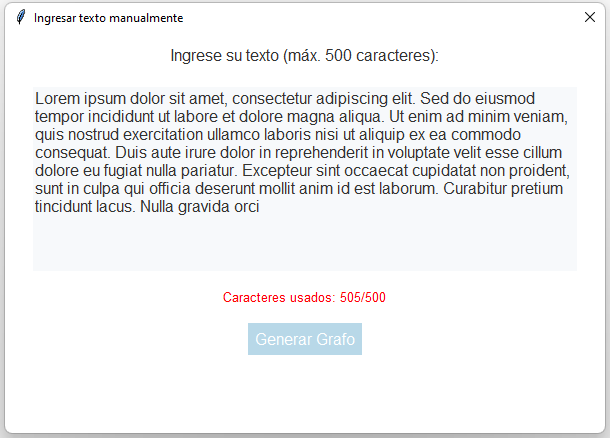




### Manejo de errores

Se diseñaron interfaces específicas para manejar errores y garantizar una experiencia de usuario positiva. Se cumplen los principios de ayuda al usuario y prevención de errores al proporcionar mensajes claros y opciones para resolver problemas. Además, se presentan de una forma intuitiva, fáciles de comprender



Se definió una cantidad máxima de 500 caracteres a introducir, ya en textos mas extensos el sistema desarrollado no presenta buenos resultados. Para mostrarle al usuario este límite de texto se eligió la variante presentada en la imagen, por sus características discretas y entendibles.

### Implementación técnica

La implementación de las interfaces del proyecto utiliza la biblioteca **tkinter** para la creación de ventanas y elementos de la GUI (interfaz gráfica de usuario).

**1. Estructura General**

El código define una clase principal, KnowledgeGraphApp, que contiene toda la lógica para crear y gestionar las interfaces. El constructor inicializa la ventana principal, configura su estilo y centra la ventana en la pantalla.

**2. Diseño y Estética**

* **Dimensiones y Posición**: Se definen dimensiones específicas para la ventana (900x600) y se centra dinámicamente en la pantalla.
* **Estilo Visual**: Se utiliza un fondo decorativo con una imagen no muy cargada para hacer la interfaz más atractiva. Además, los botones e iconos tienen bordes redondeados y colores suaves para mejorar la estética.

**3. Interactividad**

Se ofrecen dos opciones principales para interactuar con el sistema:

* **Texto Manual**: Abre una ventana secundaria donde el usuario puede escribir un texto de hasta el límite de caracteres definido.
* **Importar Archivo**: Permite al usuario seleccionar un archivo de texto desde su computadora. Este contenido puede editarse antes de procesarlo.

**4. Manejo de Ventanas Secundarias**

Cada opción abre una ventana secundaria (Toplevel) dependiente de la principal. Estas ventanas están diseñadas para ser claras y accesibles, con elementos como cuadros de texto, etiquetas de conteo de caracteres y botones.

**5. Validación de Entrada**

* En el caso de entrada manual, el sistema valida que el texto no exceda el límite definido de caracteres.
* Si se selecciona un archivo, verifica que sea válido y muestra un mensaje de error en caso contrario.

**6. Retroalimentación al Usuario**

El sistema proporciona retroalimentación visual en varias etapas:

* Mensajes de error en color rojo si hay problemas con el texto o el archivo.
* Indicaciones de progreso al generar un grafo mediante ventanas de progreso con mensajes dinámicos.
* Cambio de color en las etiquetas y botones según el estado (por ejemplo, verde para éxito y rojo para errores).

**7. Pipeline de Generación de Grafos**

La funcionalidad principal del sistema, generar grafos de conocimiento, se implementa a través del método generate\_graph. Este utiliza la clase TextToGraphPipeline para procesar el texto y generar el grafo. Durante este proceso:

* Se actualiza una ventana de progreso para mantener al usuario informado.
* Se gestiona el color del botón final según el resultado del proceso.

El diseño se enfoca en proporcionar una experiencia de usuario fluida, con elementos visualmente atractivos, retroalimentación clara y funcionalidades accesibles. Esta implementación combina estética, funcionalidad e interactividad para ofrecer un sistema completo de generación de grafos de conocimiento.

## Exportación del grafo

La exportación del grafo se realiza guardando las relaciones (aristas) presentes en el mismo en un archivo CSV. Cada fila del archivo representa una relación entre dos nodos, incluyendo la siguiente información:

1. **Nodo Origen**: El nodo desde el cual se origina la relación.
2. **Nodo Destino**: El nodo al cual está dirigida la relación.
3. **Relación**: La etiqueta que describe el vínculo entre los nodos.

El nombre del archivo se genera dinámicamente si no se especifica uno, utilizando el formato de nombre “graph\_fechaExportacion\_horaExportacion.csv”, garantizando que los nombres de los archivos sean únicos y reflejen el momento en el que se generaron.

El grafo se exporta en formato CSV con la estructura:

* Primera fila (encabezados): Nodo Orígen, Nodo Destino, Relación.
* Filas siguientes: Cada fila representa una relación en el grafo, con los datos organizados en columnas.

Ejemplo de salida en CSV:

**Nodo Origen, Nodo Destino, Relación**

**Cristóbal Colón, América, llegó a**

**Reyes Católicos, Cristóbal Colón, patrocinaron**

Se seleccionó este formato ya quees ampliamente reconocido y compatible con una variedad de herramientas y plataformas, como hojas de cálculo (Excel, Google Sheets) y librerías de análisis de datos (pandas en Python), su estructura simple basada en texto plano permite fácil lectura y edición.

Los datos exportados en CSV pueden transformarse fácilmente en otros formatos (como grafos visualizables en herramientas como Gephi o NetworkX).

También, el formato es intuitivo para los usuarios que quieran inspeccionar los datos manualmente, ya que las relaciones están organizadas de manera clara y tabular.

## Conclusiones parciales

El capítulo 2 aborda de manera detallada los aspectos técnicos y metodológicos implementados en el desarrollo del sistema, permitiendo una comprensión integral de los elementos que componen la solución propuesta. Las conclusiones que se presentan a continuación sintetizan los resultados y aprendizajes más relevantes obtenidos en esta fase, destacando los avances alcanzados y su contribución al objetivo general del proyecto.

* La solución propuesta logra integrar de manera efectiva herramientas y modelos ligeros, como spaCy y Llama2:7b, permitiendo la extracción de entidades y relaciones semánticas en textos en español, incluso en entornos de bajos recursos.
* El diseño modular del sistema basado en un patrón N-capas facilita la escalabilidad y reutilización de componentes, asegurando una implementación adaptada a las necesidades específicas del proyecto.
* La inclusión de algoritmos heurísticos para la visualización de grafos optimiza la claridad de los nodos y aristas, mejorando significativamente la legibilidad y utilidad de los grafos generados.
* La interfaz de usuario desarrollada con Tkinter simplifica la interacción con el sistema, haciendo accesibles las funcionalidades para usuarios con conocimientos técnicos limitados.
* El enfoque del sistema hacia la exportación en formatos estándar como CSV garantiza la interoperabilidad y reutilización de los grafos generados en aplicaciones educativas, de investigación o empresariales.
* La solución presenta aplicabilidad en el contexto dado, contribuyendo a resolver la problemática inicial relacionada con la generación de grafos de conocimiento en entornos de recursos limitados.

# Validación de los resultados

En este capítulo, se detallan los métodos empleados para evaluar el desempeño del modelo propuesto en la extracción de entidades y la generación de grafos. El propósito es evaluar su eficacia mediante métricas de evaluación como precisión, exhaustividad, F1-score y exactitud, así como analizar los recursos de hardware ocupados durante las pruebas.

## Marco de evaluación

Dado el contexto y la naturaleza del proyecto, las evaluaciones fueron realizadas de manera manual, utilizando como casos de prueba textos extraídos de Wikipedia en español, provenientes de diferentes dominios y categorizados según su tamaño. Esta metodología permite comprobar la capacidad de adaptación del modelo a distintos escenarios, que abarcan desde textos simples hasta textos más complejos. La evaluación se centró en tres aspectos fundamentales: la calidad de la extracción de entidades, la calidad de las relaciones generadas en los grafos, y el uso de los recursos computacionales durante la ejecución del modelo.

Una de las principales razones para la elección de este enfoque de evaluación manual radica en la carencia de datos etiquetados en español para tareas de extracción de entidades y análisis de relaciones. La disponibilidad de corpus etiquetados en otros idiomas, como el inglés, es significativamente mayor, sin embargo, en el ámbito del español, los recursos etiquetados son limitados, lo que hace necesario recurrir a evaluaciones manuales para asegurar una evaluación precisa y detallada. Además, la ambigüedad inherente al lenguaje español, que posee múltiples interpretaciones para muchas palabras o frases en función del contexto, complica la evaluación automatizada. Este desafío se ve intensificado por la flexibilidad que debe tener la evaluación, que no puede limitarse a un marco cerrado o rígido. En su lugar, debe permitir espacio para interpretaciones y valoraciones que consideren las diversas formas en que se pueden presentar las entidades y relaciones en un texto.

Para evaluar la calidad del modelo, se utilizaron métricas ampliamente aceptadas en el ámbito de la extracción de información, tales como la precisión, el recall y la puntuación F1. Estas métricas permiten ofrecer una visión cuantitativa y balanceada del rendimiento del modelo, midiendo la capacidad de este para identificar correctamente las entidades (precisión), su habilidad para encontrar todas las entidades relevantes (recall) y una combinación de ambos aspectos (F1 score). Para ello, se identificaron los falsos negativos, falsos positivos y verdaderos positivos en las pruebas realizadas, lo que facilitó un análisis detallado de la exactitud y cobertura del modelo.

## Datos para la evaluación

En esta sección se presentan y caracterizan los textos seleccionados para la evaluación de la solución propuesta.

### Textos simples

**Características**:

* Textos cortos y directos.
* Relaciones explícitas y sencillas, enfocadas en una acción, descripción o ubicación clara.
* Ideales para evaluar la extracción de relaciones básicas entre entidades sin ambigüedad.
* Los textos están diseñados para garantizar que el modelo identifique entidades y relaciones sin necesidad de análisis contextual complejo.

|  |  |
| --- | --- |
| Cant de oraciones | 1 a 2 |
| Cant de entidades | 2 a 3 |
| Cant de relaciones | 1 a 2 |

Los textos usados en esta categoría son:

* 1. "La Torre Eiffel está en París. Es uno de los monumentos más visitados del mundo."
  2. "El río Amazonas cruza Brasil."
  3. "La aspirina sirve para aliviar dolores de cabeza."
  4. "Cristóbal Colón llegó a América en 1492. Fue patrocinado por los Reyes Católicos."
  5. "El Sol es una estrella que ilumina la Tierra."
  6. "La penicilina combate infecciones bacterianas. Fue descubierta por Alexander Fleming."
  7. "El Monte Everest es la montaña más alta del mundo."
  8. "El fútbol es popular en Argentina."
  9. "El oro es un metal precioso usado en joyería."
  10. "La luna orbita alrededor de la Tierra."

### Textos intermedios

**Características**:

* Relaciones ligeramente más complejas que en los textos simples, algunas de las cuales requieren desambiguación.
* Contexto más elaborado, con múltiples entidades que interactúan o comparten características dentro de un marco común.
* Los textos evalúan la capacidad del modelo para manejar un mayor número de relaciones y entidades en un espacio más reducido.

|  |  |
| --- | --- |
| Cant de oraciones | 2 a 3 |
| Cant de entidades | 5 a 6 |
| Cant de relaciones | 3 a 5 |

Los textos usados en esta categoría son:

1. "El río Nilo atraviesa Egipto, Sudán y Uganda. Es una fuente vital para la agricultura."
2. "La Segunda Guerra Mundial terminó en 1945 con la rendición de Japón. Los Aliados ocuparon Alemania y establecieron nuevas políticas en Europa."
3. "La aspirina, fabricada por Bayer, se utiliza para tratar dolores, fiebre e inflamaciones."
4. "En la Revolución Industrial, Inglaterra, Francia y Alemania lideraron el desarrollo de maquinaria. Este avance marcó un cambio en la economía mundial."
5. "El Amazonas, el río más largo del mundo, cruza Brasil, Perú y Colombia. Alberga numerosas especies endémicas."
6. "El sistema inmunológico combate bacterias y virus mediante células como linfocitos y macrófagos."
7. "La fotosíntesis permite a las plantas producir oxígeno y glucosa a partir de dióxido de carbono y agua."
8. "Los Reyes Católicos financiaron la expedición de Cristóbal Colón, que descubrió América en 1492. Este evento marcó el inicio de la colonización."
9. "El Titanic, construido por ingenieros británicos, se hundió en el Atlántico tras chocar con un iceberg. Fue considerado uno de los barcos más modernos de su época."
10. "El cubismo, liderado por Picasso y Braque, revolucionó el arte del siglo XX."

### Textos avanzados

**Características**:

* Textos de longitud moderada.
* Relaciones más densas y variadas que abarcan diferentes dominios.
* Las interacciones entre entidades no siempre son explícitas, requiriendo un mayor análisis contextual y capacidad de desambiguación.
* Ideales para evaluar la habilidad del modelo de capturar conexiones cruzadas entre entidades en contextos con mayor complejidad semántica.

|  |  |
| --- | --- |
| Cant de oraciones | 3 a 5 |
| Cant de entidades | 7 a 12 |
| Cant de relaciones | 6 a 11 |

Los textos usados en esta categoría son:

1. "El ADN, compuesto por adenina, citosina, guanina y timina, se encuentra en el núcleo de las células. Fue descubierto por Watson y Crick, quienes recibieron el Premio Nobel en 1962. Esta molécula es fundamental para la herencia genética."
2. "El cambio climático, causado por el uso de combustibles fósiles como carbón y petróleo, afecta ecosistemas como el Ártico. Los osos polares y los corales son algunas de las especies más afectadas. Las políticas de mitigación buscan reducir estas emisiones."
3. "En 1492, Cristóbal Colón llegó a América bajo el patrocinio de Isabel y Fernando, los Reyes Católicos. Su expedición marcó el inicio de la colonización europea en el continente. Este evento transformó el comercio mundial."
4. "La revolución digital, impulsada por tecnologías como internet, teléfonos móviles y redes sociales, ha transformado sectores como la educación, el comercio y la comunicación. Empresas tecnológicas han liderado este cambio."
5. "La teoría de la evolución, desarrollada por Charles Darwin, explica la diversidad de especies en el planeta. Conceptos como selección natural y adaptación son claves en esta teoría. Esta idea cambió nuestra comprensión del mundo."
6. "El sistema solar está formado por el Sol, planetas como Marte y Júpiter, y cuerpos menores como asteroides y cometas. La Tierra es el único planeta conocido con vida. Otros planetas aún están siendo estudiados por la ciencia."
7. "En 1917, la Revolución Rusa derrocó al zar Nicolás II. Lenin, líder de los bolcheviques, estableció un régimen comunista que influyó en la política mundial. Esto marcó el inicio de la Unión Soviética."
8. "El Renacimiento, iniciado en Italia, destacó por avances en arte, ciencia y literatura. Artistas como Miguel Ángel y Da Vinci dejaron un legado importante. Ciudades como Florencia se convirtieron en centros culturales."
9. "La selva amazónica, conocida como el pulmón del planeta, alberga especies como el jaguar, el delfín rosado y el tucán. Además, es el hogar de muchas comunidades indígenas. La deforestación amenaza esta región única."
10. "La guerra fría enfrentó a Estados Unidos y la Unión Soviética en una lucha ideológica, política y económica. Conflictos como la crisis de los misiles en Cuba marcaron este periodo. El colapso soviético en 1991 puso fin a esta era."

### Textos extensos

**Características**:

* Es la categoría de las definidas que presenta los textos más largos
* Relaciones densas y cruzadas que abarcan múltiples dominios (historia, ciencia, geografía, etc.).
* Las conexiones entre entidades son variadas y pueden estar dispersas en el texto, lo que demanda un análisis más profundo para capturarlas todas.
* Los textos evalúan la capacidad del modelo para procesar y extraer información en escenarios de alta complejidad, donde las entidades y sus relaciones están más entrelazadas.

|  |  |
| --- | --- |
| Cant de oraciones | 4 a 6 |
| Cant de entidades | 15 a 28 |
| Cant de relaciones | 11 a 18 |

Los textos usados en esta categoría son:

1. "El cerebro humano está compuesto por el cerebro, el cerebelo y el tronco encefálico. Las neuronas se comunican mediante sinapsis, utilizando neurotransmisores como serotonina y dopamina. Alteraciones en estas sustancias pueden causar enfermedades como depresión y esquizofrenia. Estos trastornos afectan millones de personas en el mundo."
2. "La Revolución Industrial marcó el inicio de la mecanización en Inglaterra, Francia y Alemania. La invención de la máquina de vapor por James Watt transformó sectores como la minería y el transporte. Ciudades como Manchester y Birmingham se convirtieron en centros industriales. Esto impulsó la economía mundial en el siglo XIX."
3. "El cambio climático, impulsado por emisiones de dióxido de carbono y metano, afecta regiones como el Ártico y el Amazonas. Animales como el oso polar y el jaguar enfrentan amenazas crecientes debido a la pérdida de hábitats. Países como Estados Unidos y China lideran las emisiones globales. Las políticas internacionales buscan reducir estos impactos."
4. "La Unión Europea, formada por países como Alemania, Francia y España, implementó políticas para reducir el cambio climático. Iniciativas como el Acuerdo de París buscan limitar el aumento de la temperatura global. Sin embargo, tensiones políticas con Rusia y China han complicado la cooperación. Este desafío continúa siendo prioritario para las naciones."
5. "La Revolución Francesa comenzó en 1789, con la toma de la Bastilla como un evento clave. Líderes como Robespierre y Danton dirigieron el movimiento, que resultó en la abolición de la monarquía. El nuevo gobierno implementó reformas como la Declaración de los Derechos del Hombre y del Ciudadano. Estos cambios influyeron en revoluciones posteriores."
6. "El sistema cardiovascular humano está compuesto por el corazón, las arterias, las venas y los capilares. El corazón bombea sangre rica en oxígeno a través de las arterias hacia los tejidos del cuerpo. Las venas transportan la sangre de regreso al corazón para ser oxigenada nuevamente en los pulmones. Enfermedades como la hipertensión, la arteriosclerosis y el infarto afectan millones de personas en el mundo. La prevención incluye ejercicio, dieta equilibrada y evitar el consumo de tabaco."
7. "En 1969, Neil Armstrong se convirtió en el primer hombre en pisar la Luna durante la misión Apolo 11. Esta hazaña, liderada por la NASA, marcó un hito en la carrera espacial entre Estados Unidos y la Unión Soviética. Durante la Guerra Fría, ambos países invirtieron grandes recursos en tecnología aeroespacial para demostrar su supremacía. La misión también incluyó a Edwin Aldrin y Michael Collins, quienes formaron parte del equipo. Este evento inspiró generaciones de científicos y exploradores."
8. "La Gran Barrera de Coral, ubicada en Australia, es el sistema de arrecifes más grande del mundo. Alberga miles de especies marinas, como tortugas, tiburones y peces payaso. Sin embargo, el cambio climático y la contaminación han provocado el blanqueamiento del coral, una amenaza directa a su ecosistema. Además, actividades como la pesca excesiva y el turismo descontrolado contribuyen a su deterioro. Organizaciones ambientales trabajan en su conservación mediante proyectos de restauración y educación."
9. "El comercio electrónico, liderado por empresas como Amazon, Alibaba y Mercado Libre, ha transformado la economía global. Durante la pandemia, las ventas en línea crecieron exponencialmente debido a las restricciones de movilidad. Plataformas como PayPal y Stripe han facilitado los pagos digitales, mientras que servicios de logística como DHL y FedEx aseguran la entrega rápida de productos. Sin embargo, la competencia entre estas empresas ha generado debates sobre el impacto en pequeños negocios y la concentración del mercado."
10. "El barroco, un movimiento artístico surgido en Europa durante los siglos XVII y XVIII, destacó por su dramatismo y exuberancia. Pintores como Caravaggio, Rubens y Velázquez crearon obras icónicas llenas de contraste y movimiento. En la música, compositores como Bach, Handel y Vivaldi revolucionaron la composición con piezas complejas como 'El Mesías' y 'Las Cuatro Estaciones'. La arquitectura barroca se reflejó en iglesias como la Basílica de San Pedro, con diseños detallados y ornamentados. Este estilo influyó en la cultura de América Latina durante la colonización."

## Métricas utilizadas

Las métricas son indicadores numéricos que evalúan el desempeño de un modelo. La escala de rendimiento generalmente va de 0 a 1, donde 1 representa un modelo perfecto que clasifica correctamente.

Para comprender en que consiste cada métrica se deben tener en cuenta las siguientes definiciones las cuales se relacionan con la capacidad del modelo para identificar correctamente los aspectos presentes en una opinión.

* **Verdaderos Negativos (VN):** Casos en los que el modelo no identifica la categoría evaluada, y esta decisión es correcta porque dicha categoría no debía ser identificada.
* **Falsos Negativos (FN):** Casos en los que el modelo no identifica la categoría evaluada, pero debería haberla identificado. Es decir, la categoría estuvo presente, pero no fue reconocida por el modelo.
* **Verdaderos Positivos (VP):** Casos en los que el modelo identifica correctamente la categoría evaluada, es decir, la categoría fue identificada de forma precisa.
* **Falsos Positivos (FP):** Casos en los que el modelo identifica erróneamente la categoría evaluada, es decir, identifica algo que no correspondía o no debía ser identificado.

Por otra parte, las métricas analizadas son las siguientes:

* **Precisión** es la proporción entre los elementos correctamente identificados y el total de elementos identificados por el modelo. Esta métrica evalúa la proporción de identificaciones positivas que son verdaderamente correctas.
* **Recall** también conocido como Exhaustividad, es la relación entre los elementos correctamente identificados y la suma total de todos los elementos que se debían identificar. Esta métrica mide la capacidad del modelo para identificar correctamente todos los elementos relevantes .
* **F1-Score** combina las medidas de precisión y recall en un solo valor, facilitando la comparación del rendimiento global entre diferentes soluciones. Se calcula mediante la media armónica entre precisión y recall .

## Resultados obtenidos

### Evaluación de la extracción de entidades

Para la tarea de extracción de aspectos, se consideraron como **Falsos Negativos** aquellas entidades que el modelo debía extraer y no lo hizo, como **Falsos Positivos** las entidades que fueron extraídas de más, que no aportaban información relevante, y como **Verdaderos Positivos** aquellas entidades identificadas correctamente.

Se presenta un ejemplo de entidades extraídas en cada categoría de texto:

|  |  |  |
| --- | --- | --- |
| Categoría | Texto | Entidades |
| Sencillo | "El Sol es una estrella que ilumina la Tierra." | sol, estrella, tierra |
| Intermedio | "El Amazonas, el río más largo del mundo, cruza Brasil, Perú y Colombia. Alberga numerosas especies endémicas." | amazonas, río, mundo, brasil, perú, colombia, numerosas especies endémicas |
| Avanzado | "La guerra fría enfrentó a Estados Unidos y la Unión Soviética en una lucha ideológica, política y económica. Conflictos como la crisis de los misiles en Cuba marcaron este periodo. El colapso soviético en 1991 puso fin a esta era." | cambio social, modernización, valores, familia, tecnología, igualdad, derechos humanos, movimientos sociales, protestas, libertad |
| Extenso | "La Revolución Francesa comenzó en 1789, con la toma de la Bastilla como un evento clave. Líderes como Robespierre y Danton dirigieron el movimiento, que resultó en la abolición de la monarquía. El nuevo gobierno implementó reformas como la Declaración de los Derechos del Hombre y del Ciudadano. Estos cambios influyeron en revoluciones posteriores." | revolución francesa, 1789, toma de la bastilla, evento clave, líderes, robespierre, danton, movimiento, abolición de la monarquía, nuevo gobierno, reformas, declaración de los derechos del hombre, ciudadano, cambios, revoluciones posteriores |

Tabla : Ejemplo de extracción de entidades para cada categoría de texto

A continuación, se presentan los resultados de las pruebas de extracción de entidades sobre los textos definidos anteriormente, incluyendo también las métricas obtenidas para cada instancia, y las métricas finales para la extracción de entidades en cada categoría de texto La columna “Num” representa el número del texto, y “Cantidad” representa la cantidad de entidades que debían haber sido extraídas.

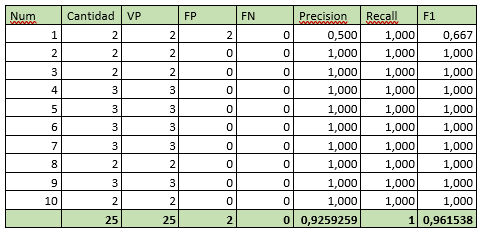


Ilustración : Resultado de la extracción de entidades en textos sencillos

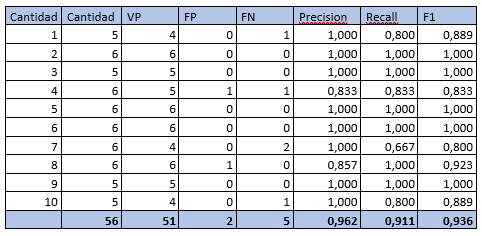


Ilustración : Resultado de la extracción de entidades en textos intermedios

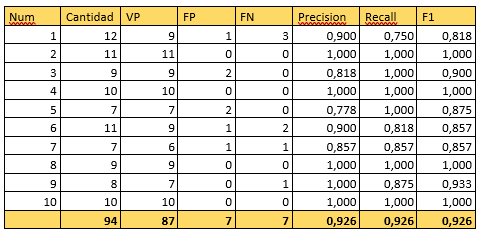


Ilustración : Resultado de la extracción de entidades en textos avanzados

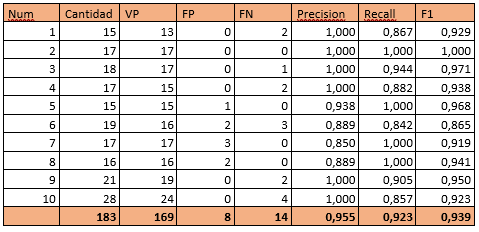


Ilustración : Resultado de la extracción de entidades en textos extensos

**Observaciones:**

Las métricas obtenidas para la extracción de entidades en todas las categorías de texto fueron satisfactorias y bastante estables, superiores a 0.91, no se aprecian diferencias significativas en la calidad de las extracciones en textos de mayor tamaño.

Se presentaron casos de fusión de entidades, pero, siempre y cuando tuviera sentido en el grafo la fusión, fueron considerados verdaderos positivos, por ejemplo, las entidades “Cristobal Colón” y “expedición” fueron consideradas correctas tanto separadas, como unidas en una sola entidad como “expedición de Cristobal Colón”

### Evaluación de la identificación de relaciones

Para la tarea de identificación de relaciones, se consideraron como **Falsos Negativos** aquellas relaciones que el modelo debía identificar y no lo hizo, como **Falsos Positivos** las relaciones identificadas que fueron alucinaciones del modelo, que no tenían sentido en el texto, o que estaban mal diseñadas, y como **Verdaderos Positivos** aquellas relaciones identificadas correctamente.

Se presenta un ejemplo de relaciones extraídas en cada categoría de texto:

|  |  |  |
| --- | --- | --- |
| Categoría | Texto | Entidades |
| Sencillo | "La penicilina combate infecciones bacterianas. Fue descubierta por Alexander Fleming." | (La penicilina, 'Combate', Infecciones bacterianas), (Alexander Fleming, 'Descubrió', La penicilina) |
| Intermedio | "El Amazonas, el río más largo del mundo, cruza Brasil, Perú y Colombia. Alberga numerosas especies endémicas." | (Amazonas, 'Cruza', Brasil, Perú, Colombia), (Numerosas especies endémicas, 'se albergan', Amazonas) |
| Avanzado | "El cambio climático, causado por el uso de combustibles fósiles como carbón y petróleo, afecta ecosistemas como el Ártico. Los osos polares y los corales son algunas de las especies más afectadas. Las políticas de mitigación buscan reducir estas emisiones." | (Los usos de combustibles fósiles, 'Afectan', Ecosistemas (como el Ártico)), (Carbón, 'Produce', Emisiones), (El cambio climático, 'Afecta', Ecosistemas (como el Ártico)), (Petróleo, 'Produce', Emisiones) |
| Extenso | "El sistema cardiovascular humano está compuesto por el corazón, las arterias, las venas y los capilares. El corazón bombea sangre rica en oxígeno a través de las arterias hacia los tejidos del cuerpo. Las venas transportan la sangre de regreso al corazón para ser oxigenada nuevamente en los pulmones. Enfermedades como la hipertensión, la arteriosclerosis y el infarto afectan millones de personas en el mundo. La prevención incluye ejercicio, dieta equilibrada y evitar el consumo de tabaco." | (Sistema cardiovascular humano, 'Bombea sangre rica', Corazón), (Capilares, 'Distribuyen el oxígeno y las glucosas a los tejidos del cuerpo', Tejidos del cuerpo), (Venas, 'Transportan la sangre de regreso al corazón', Pulmones) |

Tabla : Ejemplo de identificación de relaciones para cada categoría de texto

A continuación, se presentan los resultados de las pruebas sobre los textos definidos anteriormente, incluyendo también las métricas obtenidas para cada instancia, y las métricas finales para la identificación de relaciones en cada categoría de texto. La columna “Num” representa el número del texto, y “Cantidad” representa la cantidad de relaciones que debían haber sido identificadas.

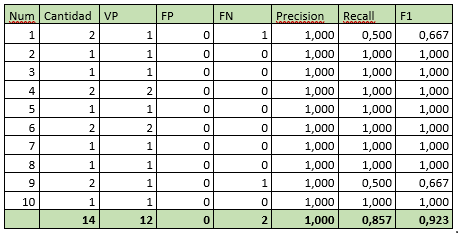


Ilustración : Resultado de la identificación de relaciones en textos sencillos

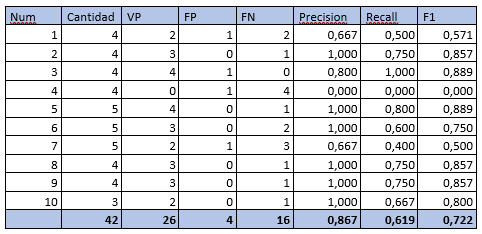


Ilustración : Resultado de la identificación de relaciones en textos intermedios

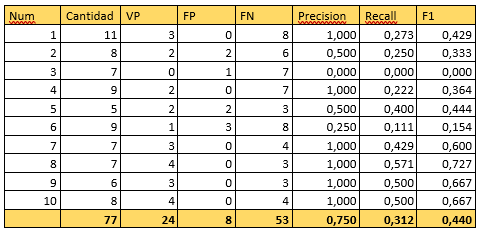


Ilustración : Resultado de la identificación de relaciones en textos avanzados

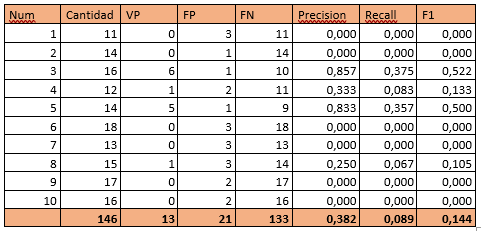


Ilustración : Resultado de la identificación de relaciones en textos extensos

**Observaciones:**

El desempeño del modelo en instancias pequeñas es muy bueno, con valor máximo de precisión, 0.85 de recall y 0.92 de F1-Score, sin embargo, a medida que aumenta el tamaño del texto y la cantidad de relaciones a identificar, empieza a presentar errores, como relaciones faltantes, alucinaciones y relaciones mal representadas, al punto de que, en la categoría mayor de textos, los resultados son bastante malos, abundando casos de conexiones en las que la etiqueta de relación es una oración demasiado extensa, y los nodos que conecta esta relación tampoco tienen sentido.

Para la evaluación de la calidad de los resultados, en aquellos casos que varias relaciones fueron resumidas en una sola, fue considerado correcto siempre y cuando tuviera sentido la fusión y no fueran excesivas la cantidad de relaciones fusionadas, por ejemplo, la relación “(Amazonas, Cruza, ‘Brasil, Perú, Colombia’)” fue considerada correcta al igual que “(Amazonas, cruza, Brasil), (Amazonas, cruza, Perú), (Amazonas, cruza, Colombia)”. La manera óptima se considera que es la segunda, pero, aun así, la primera variante no es incorrecta.

### Recursos computacionales utilizados

Además de la evaluación en cuanto a la calidad de la extracción de entidades y relaciones, se observó el impacto en los recursos computacionales durante la ejecución del modelo. Para ello se utilizó **psutil**, una biblioteca de Python que permite obtener información sobre el uso de la CPU, la memoria RAM y el disco duro (HDD). Esta herramienta permitió realizar un seguimiento en tiempo real del consumo de recursos del sistema y analizar la eficiencia computacional del modelo.

A continuación, se muestran los resultados promedios del uso del CPU, RAM, HDD y el tiempo de ejecución para las ejecuciones con cada categoría de texto.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Tipo de texto | CPU | RAM | HDD | Tiempo de ejecucion (min) |
| Pequeños | 15.19% | 80.00% | 51.43% | 18.32 |
| Intermedios | 11.54% | 81.82% | 51.44% | 32.11 |
| Avanzados | 12.98% | 84.70% | 51.60% | 17.49 |
| Extensos | 10.87% | 85.49% | 51.60% | 17.95 |

Tabla : Recursos computacionales de cada ejecución promedio por categoría de texto

Tras el análisis desarrollado, se puede apreciar que los recursos computacionales no muestran significativas variaciones con el aumento de texto presente en las pruebas realizadas. La memoria RAM en promedio se mantiene entre el 80 y 84 porciento de uso, el uso de disco se mantiene en el 51% en todos los casos, y el uso del CPU no muestra patrones de crecimiento a medida que aumenta el tamaño del texto. Por su parte, el tiempo de ejecución se mantiene bastante estable por debajo de 20 minutos, con la excepción del caso de los textos intermedios, esto puede ser ocasionado por sobrecarga del medio computo por causas externas, no parece estar asociado al tamaño del texto a procesar ya que en textos de mayor tamaño el tiempo de ejecución se mantiene sobre la normalidad.

## Conclusiones parciales

En este apartado se presentan las conclusiones derivadas del proceso de evaluación realizado, destacando los principales resultados obtenidos y los aprendizajes derivados del análisis desarrollado. Estas conclusiones ofrecen una visión general sobre el desempeño del modelo y su comportamiento en los diferentes escenarios evaluados.

* La metodología de evaluación manual permitió superar las limitaciones impuestas por la falta de datos etiquetados en español, demostrando ser un enfoque útil para garantizar una evaluación flexible y precisa, adaptada a las complejidades del lenguaje español.
* Las métricas utilizadas (precisión, recall y F1-Score) proporcionaron un análisis cuantitativo claro del desempeño del modelo tanto en la extracción de entidades como en la identificación de relaciones, permitiendo identificar fortalezas y áreas de mejora en ambos aspectos.
* En la extracción de entidades, el modelo mostró un rendimiento estable en todas las categorías de texto, sin diferencias significativas en la calidad, incluso en textos más complejos. Las fusiones de entidades fueron consideradas correctas cuando mantenían sentido semántico en el grafo, evidenciando la flexibilidad del enfoque evaluativo.
* En la identificación de relaciones, el modelo presentó un desempeño sólido en textos simples, pero a medida que los textos aumentaron en tamaño y complejidad, se observaron errores como relaciones faltantes, alucinaciones y etiquetas de relaciones mal representadas, destacando la necesidad de ajustes en la generación de relaciones en escenarios más complejos.
* El análisis del uso de recursos computacionales reveló que el modelo mantiene un consumo estable de CPU, RAM y HDD en todas las categorías de texto, sin un aumento significativo asociado al tamaño o complejidad de los textos procesados. El tiempo de ejecución también mostró estabilidad, salvo en un caso puntual con textos intermedios, probablemente causado por factores externos al modelo.
* La metodología aplicada sentó una base sólida para futuras mejoras y optimizaciones del modelo en tareas de extracción de información y generación de grafos.

# Conclusiones

El sistema desarrollado representa una solución práctica y eficiente para la construcción automática de grafos de conocimiento en entornos con recursos computacionales limitados. Su diseño modular y ligero, basado en herramientas como spaCy y Llama2:7b, permitió superar las barreras tradicionales en el procesamiento de texto en español, un idioma con retos adicionales debido a su complejidad lingüística.

El enfoque modular del sistema basado en un patrón N-capas garantiza escalabilidad, permitiendo futuras ampliaciones o adaptaciones para nuevos dominios o lenguajes. Esto asegura que el sistema pueda evolucionar para cumplir con necesidades específicas.

Las pruebas realizadas evidenciaron que el sistema es capaz de extraer entidades de manera estable y precisa en textos de diferentes tamaños y niveles de complejidad. En cuanto a la identificación de relaciones semánticas, tiene un buen desempeño en textos muy sencillos, pero es necesario valorar alternativas para mejorarla en textos de mayor complejidad.

El análisis de recursos computacionales demostró que el sistema mantiene un uso consistente de CPU, RAM y HDD, con tiempos de ejecución estables, incluso en escenarios de mayor carga. Esto confirma la usabilidad del sistema en entornos modestos, como laptops de gama baja. La RAM se vio un poco sobrecargada, pero aun con las limitantes, el programa funciona en el entorno definido.

La exportación de los grafos en formato CSV y la visualización heurística optimizada facilitan la interoperabilidad del sistema con otras herramientas y su uso en aplicaciones externas, lo que amplía significativamente su aplicabilidad.

A pesar de los logros alcanzados, se identificaron áreas de mejora, especialmente en la identificación de relaciones en textos más complejos, donde se observaron errores como relaciones faltantes o etiquetas poco precisas. Estas limitaciones resaltan la necesidad de ajustes en el diseño de prompts y algoritmos para estos escenarios.

# Recomendaciones

Luego del desarrollo del presente proyecto, se identificaron posibles áreas a mejorar:

Optimización del modelo para textos complejos:

* Refinar los prompts utilizados para la identificación de relaciones semánticas, especialmente en textos largos o con múltiples entidades interrelacionadas.
* Explorar otras alternativas para la extracción de relaciones buscando mejores resultados, manteniendo la compatibilidad con entornos de bajos recursos.

Ampliación del conjunto de evaluación:

* Incorporar textos de dominios más específicos o especializados para evaluar la capacidad del sistema en escenarios menos genéricos.

Optimización del uso de recursos:

* Investigar alternativas para reducir aún más el tiempo de ejecución.

Automatización y etiquetado:

* Desarrollar herramientas auxiliares que automaticen parcialmente el proceso de evaluación, generando un corpus etiquetado que facilite evaluaciones futuras.

# Referencias bibliográficas

1. Mihindukulasooriya N, Tiwari S, Enguix CF, Lata K, editors. Text2kgbench: A benchmark for ontology-driven knowledge graph generation from text2023: Springer.

2. Friedman S, Magnusson I, Sarathy V, Schmer-Galunder S. From unstructured text to causal knowledge graphs: A transformer-based approach. arXiv preprint arXiv:220211768. 2022.

3. Pan S, Luo L, Wang Y, Chen C, Wang J, Wu X. Unifying large language models and knowledge graphs: A roadmap. IEEE Transactions on Knowledge and Data Engineering %@ 1041-4347. 2024.

4. Trajanoska M, Stojanov R, Trajanov D. Enhancing knowledge graph construction using large language models. arXiv preprint arXiv:230504676. 2023.

5. Zhu Y, Wang X, Chen J, Qiao S, Ou Y, Yao Y, et al. Llms for knowledge graph construction and reasoning: Recent capabilities and future opportunities. World Wide Web. 2024;27(5):58 %@ 1386-145X.

6. Wu S, Tu Q, Liu H, Xu J, Liu Z, Zhang G, et al., editors. Unify Graph Learning with Text: Unleashing LLM Potentials for Session Search2024.

7. Osman AH, Barukub OM. Graph-based text representation and matching: A review of the state of the art and future challenges. IEEE Access. 2020;8:87562-83 %@ 2169-3536.

8. Wang H, Ren H, Leskovec J. Entity context and relational paths for knowledge graph completion. arXiv preprint arXiv:200206757. 2020;47.

9. Melnyk I, Dognin P, Das P. Knowledge graph generation from text. arXiv preprint arXiv:221110511. 2022.

10. Luan Y, He L, Ostendorf M, Hajishirzi H. Multi-task identification of entities, relations, and coreference for scientific knowledge graph construction. arXiv preprint arXiv:180809602. 2018.

11. Bosselut A, Rashkin H, Sap M, Malaviya C, Celikyilmaz A, Choi Y. COMET: Commonsense transformers for automatic knowledge graph construction. arXiv preprint arXiv:190605317. 2019.

12. Zhang D, Yuan B, Wang D, Liu R, editors. Joint semantic relevance learning with text data and graph knowledge2015.

13. Lu F, Cong P, Huang X. Utilizing textual information in knowledge graph embedding: A survey of methods and applications. IEEE Access. 2020;8:92072-88 %@ 2169-3536.

14. Ni P, Okhrati R, Guan S, Chang V. Knowledge graph and deep learning-based text-to-GraphQL model for intelligent medical consultation chatbot. Information Systems Frontiers. 2024;26(1):137-56 %@ 1387-3326.

15. Kim T, Yun Y, Kim N. Deep learning-based knowledge graph generation for COVID-19. Sustainability. 2021;13(4):2276 %@ 071-1050.

16. Abedini F, Mahmoudi F, Jadidinejad AH. From text to knowledge: Semantic entity extraction using yago ontology. International Journal of Machine Learning and Computing. 2011;1(2):113 %@ 2010-3700.

17. Wang Y, Zhang C, Li K. A review on method entities in the academic literature: Extraction, evaluation, and application. Scientometrics. 2022;127(5):2479-520 %@ 0138-9130.

18. Dalvi BB, Cohen WW, Callan J, editors. Websets: Extracting sets of entities from the web using unsupervised information extraction2012.

19. Ghannay S, Caubrière A, Estève Y, Camelin N, Simonnet E, Laurent A, et al., editors. End-to-end named entity and semantic concept extraction from speech2018: IEEE.

20. Consoli S, Reforgiato Recupero D, Saisana M. Data science for economics and finance: Methodologies and applications: Springer Nature; 2021.

21. Specia L, Motta E, editors. A hybrid approach for extracting semantic relations from texts2006.

22. Zhou G, Qian L, Fan J. Tree kernel-based semantic relation extraction with rich syntactic and semantic information. Information Sciences. 2010;180(8):1313-25 %@ 0020-255.

23. Akbik A, Broß J, editors. Wanderlust: Extracting semantic relations from natural language text using dependency grammar patterns2009.

24. Ben Abacha A, Zweigenbaum P. Automatic extraction of semantic relations between medical entities: a rule based approach. Journal of biomedical semantics. 2011;2:1-11.

25. González EB. Extracción de relaciones semánticas y entidades en documentos del dominio de salud: Benemérita Universidad Autónoma de Puebla; 2022.

26. Gigerenzer G, Gaissmaier W. Heuristic decision making. Annual review of psychology. 2011;62(1):451-82 %@ 0066-4308.

27. Doğrusöz U, Madden B, Madden P, editors. Circular layout in the graph layout toolkit1996: Springer.

28. Palacios CPO. Métodos basados en patrones léxicos para la extracción de información. Instituto Nacional de Astrofísica, Óptica y Electrónica. 2008.

29. Li Y, Liang X, Hu Z, Chen Y, Xing EP. Graph transformer. 2019.